

Microsoft SQL Server

SQL Server 2014 Upgrade Technical Guide

Writers: Ron Talmage, Richard Waymire, James Miller, Vivek Tiwari, Ken Spencer, Paul Turley, Danilo Dominici, Dejan Sarka, Johan Åhlén, Nigel Sammy, Allan Hirt, Herbert Albert, Antonio Soto, Régis Baccaro, Milos Radivojevic, Jesús Gil, Simran Jindal, Craig Utley, Larry Barnes, Pablo Ahumada

Published: December 2014

Applies to: SQL Server 2014

Summary: This technical guide takes you through the essentials for upgrading SQL Server 2005, SQL Server 2008, and SQL Server 2008 R2 instances to SQL Server 2014.

Copyright

This document is provided “as-is”. Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

© 2014 Microsoft. All rights reserved.

Contents

SQL Server 2014 Upgrade Technical Guide	1
Copyright	2
Introduction	17
Executive Summary	18
Planning the Upgrade	18
The Relational Database Environment	20
The Business Intelligence Environment	21
Chapter 1: Upgrade Planning and Deployment	22
Introduction	22
Upgrading SQL Server 2012 to SQL Server 2014	22
Preparing to Upgrade.....	22
Upgrade Strategies.....	23
Upgrade Tools.....	38
SQL Server 2014 Setup.....	44
Allowable Upgrade Paths.....	50
Application and Connection Requirements.....	56
Upgrading Applications that Use the .NET Framework.....	57
Plan for Backups	59
Upgrading Both Windows and SQL Server	60
Upgrading Multiple Instances	63
Upgrading Very Large Databases	63
Upgrading High Availability Servers	64
Minimizing Upgrade Downtime	64
Developing an Upgrade Plan	66
Treat the Upgrade as an IT Project.....	66
Minimize Variables Involved in the Upgrade.....	69
Create Upgrade Checklists	73

Test the Upgrade Plan.....	74
Develop Acceptance Criteria and Rollback Steps.....	77
Post-Upgrade Tasks	78
Integrate the New Instance into Its New Environment.....	78
Determine Application Acceptance.....	79
Run the SQL Server 2012 Best Practices Analyzer	79
Troubleshooting an Upgrade	79
Decommission and Uninstall After a Side-by-Side or New Hardware Upgrade.....	80
Considerations for Upgrading Without a DBA	80
Conclusion.....	83
Additional References	83
Chapter 2: Management Tools.....	84
Introduction	84
Upgrading to SQL Server 2014 and SQL Server’s Management Tools	84
Upgrading In-Place.....	84
Tools Connectivity	85
SQL Server Management Studio (SSMS).....	85
Upgrading Master and Target Servers.....	86
Project Files	86
Post-Upgrade SSMS Tasks.....	86
Azure-Facing Features in SSMS 2014.....	87
SQL Server Data Tools.....	88
Conclusion.....	88
Chapter 3: Relational Databases	89
Introduction	89
Relational Database Configurations.....	89
Upgrade Considerations.....	90
Full-Text Search	91
What Can Be Upgraded?.....	92

What Cannot Be Upgraded?	93
In-Place Upgrade vs. Side-by-Side Upgrade	95
In-Place Upgrade.....	95
Side-by-Side Upgrade.....	95
Evaluating Potential Upgrade Issues.....	102
Deprecated Features.....	102
Discontinued Functionality.....	102
Breaking Changes	103
Behavior Changes	105
Preparing for an Upgrade.....	105
Preparing for an In-Place Upgrade.....	106
Preparing for a Side-by-Side Upgrade	110
Performing an Upgrade.....	111
Performing an In-Place Upgrade	111
Performing a Side-By-Side Upgrade	112
Post-Upgrade Tasks	114
In-Place Upgrade.....	114
Side-by-Side Upgrade.....	115
General Post-Upgrade Tasks	115
Review Use Plan Hints.....	116
Important Information about Query Plans.....	117
Connecting Client Applications to SQL Server 2014.....	117
Conclusion.....	118
Additional References	118
Chapter 4: High Availability	119
Introduction	119
Preparing to Upgrade.....	119
In-Place Upgrade.....	119
Side-by-Side Upgrade on the Same Standalone Server or Cluster	120

Side-by-Side Upgrade to a Separate Standalone Server or Cluster	122
Decommissioning and Disabling the Original Instance or Database in a Side-by-Side Upgrade.....	123
Methods for Side-by-Side Upgrades to a Separate Server or Cluster.....	123
Which SQL Server Upgrade Method Should You Use?	126
Minimizing Downtime During the Upgrade	126
Prepare for SQL Server 2014.....	126
Devise an Upgrade Plan	128
Test the Upgrade Plan.....	129
Prepare Servers and Instances for SQL Server 2014	130
Upgrading Failover Cluster Instances	134
Feature Changes in SQL Server 2014 Failover Clustering.....	134
Operating System Versions and SQL Server 2014 Failover Clustering	135
Considerations for Upgrading a SQL Server 2005 Failover Cluster to SQL Server 2014...	139
Considerations for Upgrading SQL Server 2008, SQL Server 2008 R2, or SQL Server 2012 to SQL Server 2014	139
Considerations for Upgrading a Pre-Release Version of SQL Server 2014 to SQL Server 2014 RTM	139
Upgrading a Failover Clustering Instance to SQL Server 2012.....	139
Additional References for Clustering Upgrades.....	156
Upgrading AlwaysOn Availability Groups.....	157
Feature Changes in SQL Server 2014 AlwaysOn Availability Groups	157
Considerations for Upgrading a SQL Server 2012 or SQL Server 2014 CTP2 Server with AlwaysOn Availability Groups.....	157
Considerations for Upgrading an Availability Group with a Single Remote Secondary Replica	158
Considerations for Upgrading SQL Server Instances with Multiple Availability Groups...	159
Upgrading Mirrored Databases	160
Feature Changes in SQL Server 2008 R2 Database Mirroring.....	160
In-Place Upgrade.....	160
Side-by-Side Upgrade to a New Server	164
Additional References for Upgrading with Mirrored Databases.....	165

Upgrading Log Shipped Databases.....	165
Feature Changes in SQL Server 2014 Log Shipping	165
Log Shipping Upgrade Scenarios.....	165
Upgrading with Multiple Secondaries.....	166
Steps to Upgrade Log Shipping without a Role Change.....	166
Steps to Upgrade Log Shipping with a Role Change.....	168
Steps to Upgrade Side-by-Side to a New Server or Cluster.....	170
Additional References for Upgrading Log Shipping.....	172
Upgrading Replicated Databases.....	172
Feature Changes in SQL Server 2014 Replication.....	172
Planning an Upgrade with Replication	172
In-Place Upgrade.....	174
Side-by-Side Upgrade to a New Server or Cluster.....	177
Additional Information for Upgrading Replicated Databases	179
Conclusion.....	179
Additional References	179
Chapter 5: Database Security	180
Introduction.....	180
New Security Features.....	180
New Configuration Tools.....	181
Configuring Services and Connections.....	181
Service Account Security.....	183
Configuring Features	184
SQL Server Audit	185
Preparing to Upgrade.....	185
Deprecated Features.....	185
Discontinued Features.....	187
Breaking Changes	188
Behavior Changes	189
Pre-Upgrade Security Tasks.....	189

Upgrading to SQL Server 2014.....	191
In-Place Upgrade.....	191
Side-by-Side Upgrade.....	191
Post-Upgrade Security Tasks.....	192
Post-Upgrade Security Testing	192
Conclusion.....	193
Additional References	193
Chapter 6: Full-Text Search	194
Introduction	194
Preparing to Upgrade.....	194
Deprecated Features.....	195
Breaking Changes	195
Behavior Changes	196
Running Upgrade Advisor	196
Preparing for a Possible Rollback	196
Upgrading a Full-Text-Enabled Database.....	198
In-Place Upgrade.....	198
Side-by-Side Upgrade.....	199
Post-Upgrade Tasks	202
Using Customized Noise-Word Files from a Previous SQL Server Version.....	203
Conclusion.....	203
Additional References	204
Chapter 7: Service Broker	205
Introduction	205
Feature Changes	205
Preparing to Upgrade.....	206
Disk Space Requirements.....	206
Upgrade Tools.....	207
64-bit Considerations.....	208

Upgrading from previous version of SQL Server	208
In-Place Upgrade.....	208
Side-by-Side Upgrade.....	209
Post-Upgrade Tasks	210
Restoring Settings.....	210
Routing Changes.....	210
Implementing Conversation Priorities.....	211
Conclusion.....	211
Additional References	212
Chapter 8: SQL Server Express	213
Introduction	213
LocalDB	214
Feature Changes	214
Preparing to Upgrade.....	214
Deprecated Features.....	217
Discontinued Functionality.....	217
Breaking Changes	217
Behavior Changes	217
Upgrade Tools.....	218
64-Bit Considerations	219
SQL Server 2014 Express Packages	219
System Requirements for SQL Server 2014 Express	220
Upgrading from SQL Server 2000 (MSDE).....	225
Upgrading to SQL Server 2014 Express.....	225
In-Place Upgrade.....	225
Side-by-Side Upgrade.....	225
Post-Upgrade Tasks	227
Upgrading to LocalDB.....	227
Upgrading to Other Editions of SQL Server 2014	229

Upgrading to the SQL Server 2014 Web Edition	230
Upgrading to SQL Server 2014 Standard Edition	231
Conclusion.....	231
Additional References	232
Chapter 9: SQL Server Data Tools.....	233
Introduction	233
Preparing to Upgrade.....	233
Upgrading Visual Studio 2010/SSDT 2012/Visual Studio 2013 Database Projects to SSDT 2014	234
Breaking and Behavior Changes	235
SQL Server Database Project Template	235
Schema Definition Files.....	235
Schema Comparison Files	236
SQLCMD Variable Definitions.....	236
Server Logins	237
Other Breaking and Behavior Changes.....	237
Pre-Upgrade Tasks.....	238
Convert .dbschema Files to .dacpac Files.....	238
Convert SQLCMD Variable Definitions.....	239
Post-Upgrade tasks	240
Change the Debugging Instance for Full-Text Search.....	240
Correct Linked Server Definitions That Use SQLCMD Variables	240
Additional References	241
Chapter 10: Transact-SQL Queries.....	242
Introduction	242
Preparing to Upgrade.....	242
Backup and Rollback Plan.....	243
Deprecated Features.....	243
Discontinued Features.....	244

Breaking Changes	244
Behavior Changes	245
Reserved Keywords	245
Upgrade Tools.....	251
64-Bit Considerations	252
Known Issues and Workarounds.....	252
Upgrading from SQL Server 2005, SQL Server 2008, SQL Server 2008 R2, or SQL Server 2012.....	253
In-Place Upgrade.....	253
Side-by-Side Upgrade.....	253
Post-Upgrade Tasks	254
Conclusion.....	254
Additional References	255
Chapter 11: Spatial Data	256
Introduction	256
Preparing to Upgrade.....	256
Deprecated Features.....	256
Discontinued Functionality.....	257
Breaking Changes	257
Behavior Changes	262
Post-Upgrade Tasks	264
Conclusion.....	264
Additional References	264
Chapter 12: XML and XQuery.....	266
Introduction	266
Preparing to Upgrade.....	266
Deprecated Features.....	266
Discontinued Functionality.....	267
Breaking Changes	267

Behavior Changes	270
Post-Upgrade Tasks	271
Features Added in SQL Server 2012.....	271
Features Added in SQL Server 2008.....	272
Conclusion.....	274
Additional References	275
Chapter 13: CLR.....	276
Introduction	276
Preparing to Upgrade.....	276
Breaking Changes	276
Behavior Changes	277
Dynamic Management View (DMV) Changes.....	282
Visual Studio 2010 Compatibility.....	283
Additional References	283
Chapter 14: SQL Server Management Objects.....	284
Introduction	284
Preparing to Upgrade.....	284
Deprecated Features.....	284
Discontinued Functionality.....	285
Upgrading from previous versions	285
SMO and PowerShell	287
Conclusion.....	288
Additional References	288
Chapter 15: Business Intelligence Tools	289
Introduction	289
BI Tools Users.....	289
Business Information Workers	289
Software Developers.....	290
System Administrators.....	290

BI Tools Overview	290
Import and Export Wizard	292
SQL Server Data Tools for Business Intelligence.....	293
SQL Server Management Studio	297
Analysis Services Deployment Wizard	298
Reporting Services Configuration Manager.....	298
Integration Services Deployment Wizard	298
Integration Services Project Conversion Wizard	299
SQL Server Profiler.....	299
Report Builder	300
Power Pivot for Excel 2013 Add-In.....	301
Power View.....	303
Data Mining Add-ins for Office	304
Preparing to Upgrade BIDS to SSDT Projects	305
Post Upgrade Tasks and Breaking Changes	308
Additional References	309
Chapter 16: Analysis Services.....	310
Introduction	310
Preparing to Upgrade: In-Place Upgrade vs. Side-by-Side Upgrade.....	312
In-Place Upgrade.....	312
Side-by-Side Upgrade.....	312
Preparing to Upgrade: Determining and Evaluating Potential Upgrade Issues	313
Deprecated Features.....	313
Discontinued Functionality.....	315
Breaking Changes	315
Behavior Changes	316
64-bit Considerations	317
Upgrading from SSAS 2005, 2008, 2008 R2, or 2012	317
Side-By-Side Upgrade.....	318

In-Place Upgrade.....	318
Post-Upgrade Tasks	320
Conclusion.....	321
Additional References	322
Chapter 17: Integration Services	323
Introduction	323
Preparing to Upgrade to SSIS 2014.....	323
SSIS Backward Compatibility	324
64-Bit Considerations	325
SSIS Sample Package Overview	326
ETL Frameworks.....	326
Master Package.....	327
Execution Packages.....	329
Running Upgrade Advisor	333
Installing SSIS 2014.....	337
SSIS 2014 Server Overview	338
In-Place Upgrade.....	339
Side-by-Side Upgrade.....	339
New Installations.....	340
Project Conversion Wizard	341
Convert to the Project Deployment Model.....	352
Additional References	364
Chapter 18: Reporting Services	365
Introduction	365
Reporting Services Editions.....	366
Upgrade Considerations.....	366
In-Place Upgrade vs. Side-by-Side Upgrade.....	369
Preparing to Upgrade.....	370
Important Reporting Services Configuration Files	371

Storing Configuration Settings	371
Deprecated Features.....	371
Discontinued Functionality.....	372
Breaking Changes	372
Behavior Changes	373
Updating Report Projects and Definitions for Use in SSDT-BI.....	373
Upgrade Tools.....	374
64-Bit Considerations	374
Known Issues and Workarounds.....	374
Backup and Rollback Plan.....	375
Upgrading from SQL Server 2005	376
In-Place Upgrade.....	376
Side-by-Side Upgrade.....	381
Upgrading from SQL Server 2008	385
In-Place Upgrade.....	385
Side-by-Side Upgrade.....	385
Upgrading from SQL Server 2008 R2 or SQL Server 2012.....	386
In-Place Upgrade.....	386
Side-by-Side Upgrade.....	387
Troubleshooting a Failed Upgrade	387
Post-Upgrade Tasks	389
Moving Reports Between SSRS 2005 and SSRS 2014.....	391
Moving Reports Between SSRS 2008, SSRS 2008 R2, SSRS 2012, and SSRS 2014.....	391
Deploying Custom Extensions and Assemblies.....	392
Verifying Configuration Files	392
Uninstalling SSRS 2005/2008/2008 R2/2012.....	393
Conclusion.....	393
Additional References	394
Chapter 19: Data Mining.....	395
Introduction	395

Data Mining Features in the Various SQL Server Versions and Editions	396
Preparing to Upgrade.....	399
Deprecated Features.....	399
Discontinued Functionality.....	399
Breaking Changes	400
Behavior Changes	401
Running Upgrade Advisor	401
Upgrading from SQL Server 2005	401
In-Place Upgrade.....	404
Side-by-Side Upgrade.....	405
Post-Upgrade Tasks	407
Upgrading from SQL Server 2008, SQL Server 2008 R2, and SQL Server 2012	416
Conclusion.....	418
Additional References	419
Chapter 20: Other Microsoft Applications and Platforms.....	420
Introduction	420
Microsoft Lync Server 2013.....	420
Microsoft Office SharePoint Server 2013	421
Microsoft System Center.....	421
Microsoft Dynamics.....	421
Conclusion.....	422
Additional References	422
Appendix 1: Version and Edition Upgrade Paths	423
Appendix 2: SQL Server 2014: Upgrade Planning Checklist	425

Introduction

To attain a smooth and trouble-free upgrade to SQL Server 2014, you must plan for the upgrade and address the complexities of your application. 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 takes you through the essential technical details for planning and testing an upgrade of existing SQL Server 2005, 2008, 2008 R2, and 2012 instances to SQL Server 2014. You will be presented with best practices for preparation, planning, pre-upgrade tasks, and post-upgrade tasks. All the SQL Server components are covered, each in its own chapter.

This document is a supplement to SQL Server 2014 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.

Executive Summary

The SQL Server 2014 Upgrade Technical Guide assists you with planning to upgrade to SQL Server 2014 by explaining the common upgrade strategies and the key decisions that you will have to make. This guide contains detailed technical information on both breaking and behavioral changes that you can expect for each SQL Server component. You will need to accommodate some breaking and behavioral changes when you upgrade from SQL Server 2005, SQL Server 2008, or SQL Server 2008 R2 to SQL Server 2014. There are no breaking changes between SQL Server 2012 and SQL Server 2014.

The guide contains chapters on general planning and strategizing, and individual chapters detailing upgrade-related issues for each SQL Server component. The remainder of this summary will point you to various chapters of the guide that discuss these topics and components.

Note: You can explore the online documentation for upgrading all SQL Server 2014 components at [Upgrade to SQL Server 2014](http://technet.microsoft.com/en-us/library/bb677622.aspx) (<http://technet.microsoft.com/en-us/library/bb677622.aspx>).

Planning the Upgrade

The degree of effort you put into planning an upgrade to SQL Server 2014 should reflect the complexity of your environment and the number of upgrades involved. Here are the general steps of the upgrade process, as explained in Chapter 1, “Upgrade Planning and Deployment”:

Research your current environment. Find out what SQL Server components are used most often on your SQL Server instances and determine whether each instance serves primarily online transaction processing (OLTP), data warehouse, or business intelligence purposes. Use the SQL Server Best Practices Analyzer to remove any bad practices in the database applications. In addition, assess the criticality of the SQL Server instance and monitor its resource and performance requirements. Finally, build a test environment for testing your upgrade plan.

Choose your target environment. The target environment, whether the same or a new server, must be able to support the original instance’s requirements. You may use the same or different servers, Windows version, or type of host. You can explore Microsoft’s various host environments for SQL Server, including the cloud, virtual machines, and bare metal at [SQL Server 2014 Server and Cloud Platform](http://www.microsoft.com/en-us/server-cloud/products/sql-server/) (<http://www.microsoft.com/en-us/server-cloud/products/sql-server/>).

Choose an edition of SQL Server 2014. Ensure that the edition of SQL Server 2014 you are upgrading to is an allowable path. For allowed paths, see Appendix 1 of the guide or go to [Supported Version and Edition Upgrades](http://msdn.microsoft.com/en-us/library/ms143393(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms143393\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143393(v=sql.120).aspx)).

Run the SQL Server Upgrade Advisor report on your server. The main tool for SQL Server upgrading is the [SQL Server Upgrade Advisor](http://msdn.microsoft.com/en-us/library/ee210467.aspx) (<http://msdn.microsoft.com/en-us/library/ee210467.aspx>). It will diagnose your SQL Server's configuration and Transact-SQL (T-SQL) code, looking for any items that might block the upgrade process or that will need to be modified. To upgrade SQL Server 2005/2008/2008 R2/2012 instances to SQL Server 2014, use the SQL Server 2014 Upgrade Advisor.

Develop an upgrade plan. The best approach is to treat your upgrade like you would any IT project. You should organize an upgrade team that has the database administration, network, extraction, transformation, and loading (ETL), and other skills required for the upgrade. The team needs to:

- Choose an upgrade strategy. There are two general strategies for upgrading:
 - In-place, where you directly replace an older instance of SQL Server with a newer one.
 - Side-by-side, where both instances remain intact and you switch operations from one to the other.

Note: In most cases, the side-by-side strategy is the best. It allows you to roll back much more easily in case a particular upgrade encounters an error. It does, however, require more resources.

- Develop a plan to back up your data before the upgrade. You can use the backup to restore the original environment if you have to roll back.
- Determine acceptance criteria (e.g., tests to ensure that the upgrade was successful).
- Develop checklists that detail the handoff points between various roles.

Test the upgrade plan. Using your test environment, make sure your upgrade and rollback steps are complete and revise the plan whenever you find issues.

Perform the upgrade. When it comes time to put your upgrade plan into action, take time to coordinate with all the teams involved. As you step through the process, make sure you follow your checklists. Note all deviations and make revisions as needed. As you repeat and refine your upgrade plan, you and your teams will become more efficient and confident.

Apply post-upgrade tasks. Apply those changes required after the upgrade. Apply the acceptance criteria you previously developed in order to determine the success of the upgrade. Depending on the complexity of your environment, you will want to anticipate what will be needed for troubleshooting if any issues are found and how to roll back to the original state if the end result is not acceptable.

For a general flowchart of this entire upgrade process, see the flowchart in Figure 4 in Chapter 1, “SQL Server 2014 Upgrade Flowchart.” For a sample checklist to use in your upgrade plan, see Appendix 2.

The Relational Database Environment

In an OLTP or data warehouse environment, you need to focus on SQL Server’s relational database components. Chapter 3, “Relational Databases,” walks you through how to implement the in-place and side-by-side strategies for the relational database components.

The components you might need to upgrade in an OLTP or data warehouse environment include:

High availability components. Often in an OLTP or data warehouse environment, the most important SQL Server instances are using some high availability technology. Chapter 4, “High Availability,” takes you through the upgrade process and illustrates a number of common scenarios, including AlwaysOn Availability Groups, Windows failover clustering, database mirroring, log shipping, and SQL Server replication.

Other relational database components. The remaining SQL Server relational database components are covered in Chapter 5 through Chapter 14. These chapters contain a wealth of technical details and specific advice. They cover, for example, database security, full-text search, T-SQL code, spatial data, Extensible Markup Language (XML), XQuery, Common Language Runtime (CLR), and SQL Server Management Objects (SMO).

The Business Intelligence Environment

A business intelligence environment has different requirements than the relational database OLTP and data warehouse environments. The SQL Server components and tools are different, and different servers are often involved. Nevertheless, there are still some breaking changes and behavioral changes when you upgrade from SQL Server 2005/2008/2008 R2 to SQL Server 2014. As with the relational components, there are no breaking or behavioral changes between SQL Server 2012 and SQL Server 2014, so upgrading from SQL Server 2012 to SQL Server 2014 is seamless.

The components you might need to upgrade in a business intelligence environment include:

SQL Server Analysis Services (SSAS). Chapter 16, “Analysis Services,” tells you how to handle deprecated objects and links you may encounter when upgrading earlier versions of SSAS to SSAS 2014. You can also find more information at [Upgrade Analysis Services](http://msdn.microsoft.com/en-us/library/ms143686(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms143686\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143686(v=sql.120).aspx)).

SQL Server Integration Services (SSIS). Significant changes to SSIS projects were made in SQL Server 2012 that directly affect the upgrade from SQL Server 2005/2008/2008 R2 to SQL Server 2014. Chapter 17, “Integration Services,” shows you how to upgrade your packages after an upgrade. Also see [Considerations for Upgrading Data Transformation Services](http://msdn.microsoft.com/en-us/library/ms143716.aspx) (<http://msdn.microsoft.com/en-us/library/ms143716.aspx>).

SQL Server Reporting Services (SSRS). Chapter 18, “Reporting Services,” details when items such as configurations, report projects, and report definitions must be modified in order to upgrade from earlier versions of SSRS to SSRS 2014. Also see [Upgrade and Migrate Reporting Services](http://msdn.microsoft.com/en-us/library/ms143747(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms143747\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143747(v=sql.120).aspx)).

Other business intelligence components. Chapter 15 covers business intelligence tools, and Chapter 19 covers upgrade issues related to data mining.

Chapter 1: Upgrade Planning and Deployment

Introduction

This first chapter lays out the general guidelines for planning a successful upgrade to SQL Server 2014. These guidelines include upgrade strategies, test and rollback considerations, and upgrade tools. This chapter also introduces you to the SQL Server 2014 Upgrade Advisor, a tool that analyzes legacy instances of SQL Server 2005, 2008, 2008 R2, and 2012, 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.

Upgrading SQL Server 2012 to SQL Server 2014

SQL Server 2014 adds compelling features to the SQL Server 2012 database engine, such as In-Memory OLTP, updateable clustered columnstore indexes, and delayed durability. Any one of these new features can be a compelling case for upgrading, depending on your need for high availability, performance, and additional functionality. In addition, 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. For SQL Server 2014 features that make upgrading helpful, see the [SQL Server](http://www.microsoft.com/sqlserver/en/us/default.aspx) home page (<http://www.microsoft.com/sqlserver/en/us/default.aspx>).

When upgrading from SQL Server 2012 to SQL Server 2014, note that there are no breaking changes and few behavioral changes. For information on behavioral changes, see the chapters in this guide. Perhaps the most significant changes from SQL Server 2012 to SQL Server 2014 concern high availability, most notably adapting AlwaysOn features to Windows Server 2012 and Windows Server 2012 R2. For information about how these enhancements affect upgrading, see Chapter 4, "High Availability," in this guide.

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, whereas other

scenarios might be complex. Start to plan by analyzing the upgrade requirements, including reviewing upgrade strategies, understanding SQL Server 2014 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 2005, 2008, 2008 R2, or SQL Server 2012 to SQL Server 2014. There are two fundamental strategies for upgrading, with two main variations in the second strategy:

- **In-place upgrade:** Use the SQL Server 2014 Setup program to directly upgrade an instance of SQL Server 2005, 2008, 2008 R2, or 2012. The older instance of SQL Server is replaced.
- **Side-by-side upgrade:** Perform steps to move all or some data from an instance of SQL Server 2005, 2008, 2008 R2, or 2012 to a separate instance of SQL Server 2014. 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 2014 Setup program directly replaces an instance of SQL Server 2005, 2008, 2008 R2 or 2012 with a new instance of SQL Server 2014 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 the "Extended System Support (WOW64)" section later in this chapter.) This kind of upgrade is called "in-place" because the upgraded instance of SQL Server is actually replaced by the new instance of SQL Server 2014. You do not have to copy database-related data from the older instance to SQL Server 2014 because the old data files are automatically converted to the new format. When the process is complete, the old instance of SQL Server is removed from the server, with only the backups that you retained being able to restore it to its previous state. Older client tools such as SQL Server Management Studio (SSMS) have to be removed manually.

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

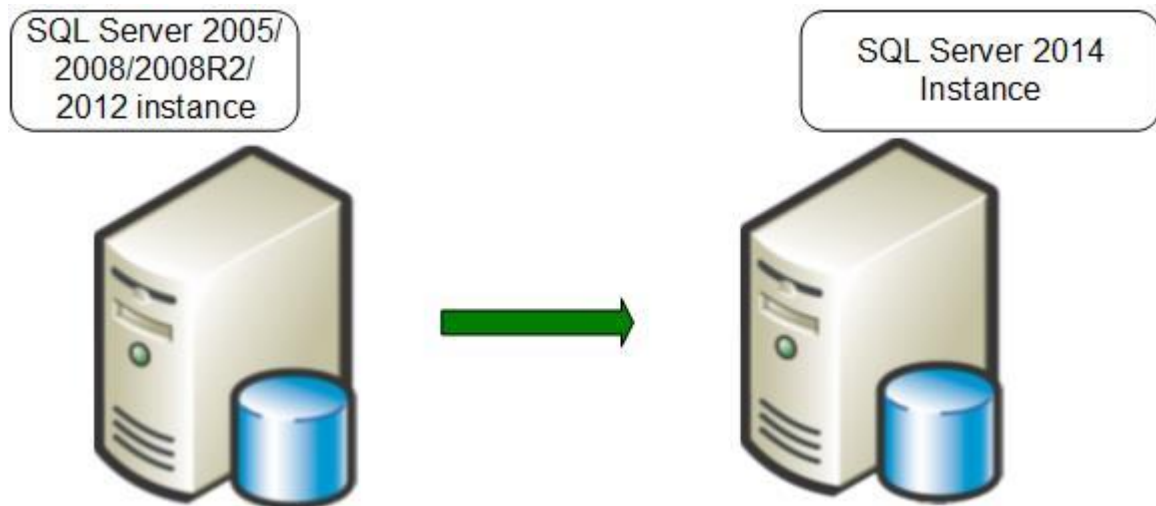


Figure 1: In an in-place upgrade, SQL Server 2014 Setup replaces the legacy instance of SQL Server

Note the following restrictions on an in-place upgrade:

- SQL Server 2014 Setup requires that all SQL Server components be upgraded together. 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 2008 R2 Database Engine without also upgrading the Analysis Services component.
- An in-place upgrade from a 32-bit instance of SQL Server 2005/2008/2008 R2 (x86) to a 64-bit instance of SQL Server 2008 R2 (x64), or vice versa, is not supported. For more information, see the note in the "Extended System Support (WOW64)" section later in this chapter.

Here are the major steps that the SQL Server 2014 Setup program takes when you perform an in-place upgrade:

1. The SQL Server 2014 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. The Setup program checks for upgrade blocking issues, a small set of issues that will completely block an upgrade. If the Setup program finds any, it will list them. You must fix them and restart the upgrade process.
3. If a pending restart exists, you will have to restart the computer.
4. The Setup program installs the required SQL Server 2014 executables and support files.
5. The Setup program stops the legacy SQL Server service. At this point, the legacy instance is no longer available.
6. SQL Server 2014 updates the selected component data and objects.
7. The Setup program removes the legacy executables and support files. Legacy SQL Server 2005/2008/2008 R2 client tools are not removed. (See Chapter 2, "Management and Development Tools," for more information.)

The new instance of SQL Server 2014 is now fully available. The legacy instance of SQL Server has been replaced and must be reinstalled 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 a legacy instance of SQL Server 2005/2008/2008 R2/2012 to a separate instance of SQL Server 2014. It is called a "side-by-side" method because the new instance of SQL Server 2014 runs alongside the legacy instance of SQL Server, either 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 2014 that is located on a different physical server or on a different virtual machine.
- You can transfer data and components to an instance of SQL Server 2014 on the same physical server.

Both options let you run the new instance of SQL Server 2014 alongside the legacy instance of SQL Server. Typically, after the upgraded instance is accepted and moved into production, you can remove the older instance.

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

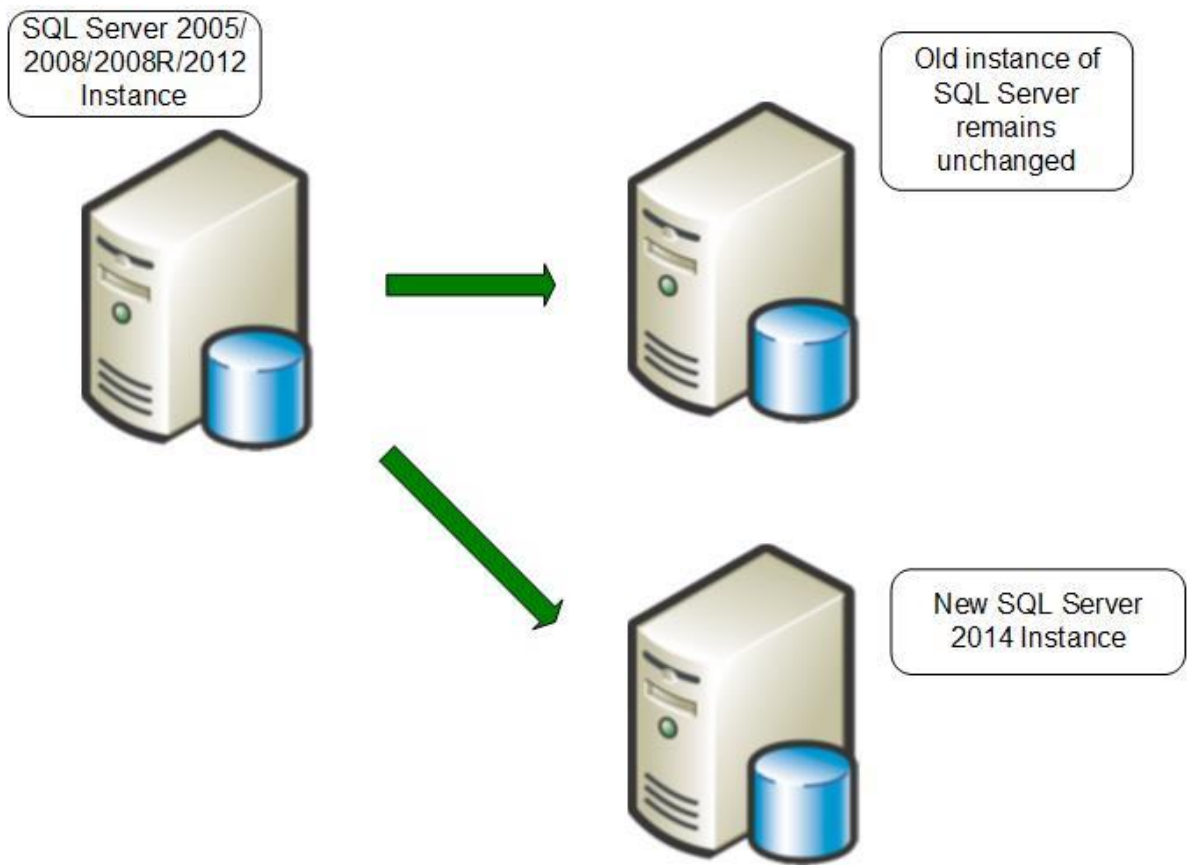


Figure 2: A side-by-side upgrade to another server leaves the legacy instance of SQL Server unchanged

You can also use the side-by-side method to upgrade to SQL Server 2014 on the same server as the legacy instance of SQL Server. Figure 3 shows a side-by-side upgrade on the same server.

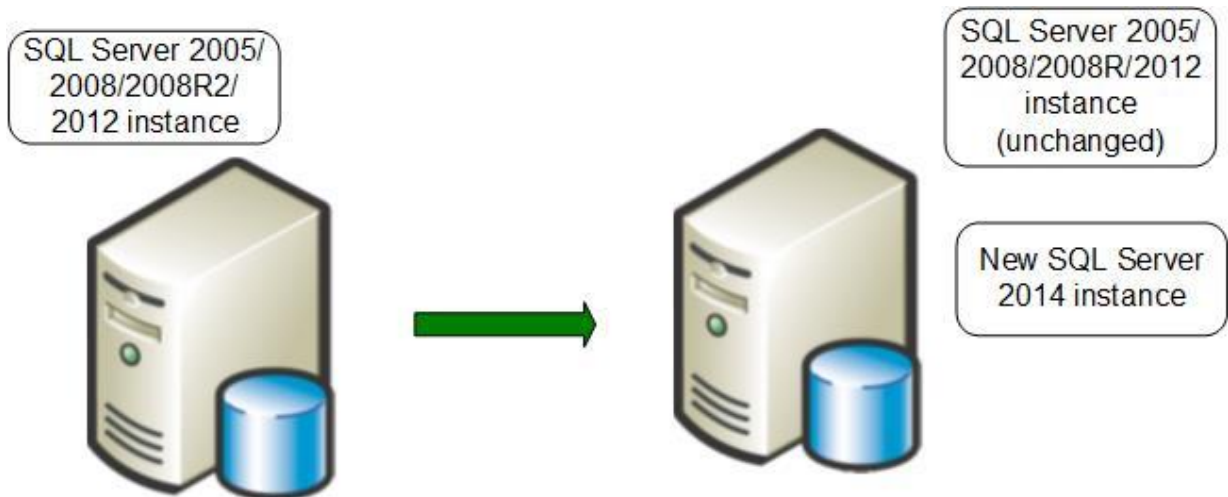


Figure 3: Performing a side-by-side upgrade on the same server, leaving both instances running

No matter whether a side-by-side upgrade is performed using a separate instance on the same server or 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 2014. The SQL Server 2014 Setup program will not perform this task. The objects that you must transfer include the following:

- Data files
- Database objects
- SQL Server Analysis Services (SSAS) cubes
- Configuration settings
- Security settings
- SQL Server Agent jobs
- SQL Server Integration Services (SSIS) packages

Here are the main steps that you must perform when doing a side-by-side upgrade of SQL Server 2005/2008/2008 R2/2012 to SQL Server 2014:

1. Install a separate instance of SQL Server 2014 on the legacy server or on a separate server. The legacy instance continues to be available.
2. Run the SQL Server 2014 Upgrade Advisor against the legacy instance, and remove any upgrade blocking 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 2014.
5. Apply supporting objects such as SQL Server Agent jobs, security settings, and configuration settings to the new instance of SQL Server 2014.
6. Upgrade SSIS (and potentially Data Transformation Services—DTS) packages to SSIS. (See Chapter 17, "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 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 any 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 performing 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 often be worth the additional effort.

Upgrading SQL Server 2000 to SQL Server 2014

You cannot upgrade a SQL Server 2000 instance or database directly to SQL Server 2014.

- For an in-place upgrade, upgrade the SQL Server 2000 instance to SQL Server 2005 SP4, SQL Server 2008, or SQL Server 2008 R2. Then apply SQL Server 2014 Setup.
- For a side-by-side upgrade, first restore the SQL Server 2000 databases to SQL Server 2005, 2008, or 2008 R2 and then backup and restore the resulting database to SQL Server 2014.

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

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

Table 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	Two
Number of physical servers involved	One	One or more
Data file transfer	Automatic	Manual
SQL Server instance configuration	Automatic	Manual
Supporting tools	SQL Server Setup	Several data transfer methods

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 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 to meet your needs.

Table 2: Upgrade Strategies and Components

Component Level	Single Resulting Instance of SQL Server 2014	
	In-place	Two Resulting Instances
All components	In-place	Side-by-side
Single component	In-place	Side-by-side
Single database	Not available	Side-by-side

The 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 the Setup program. (See "Setup Requirements for an In-Place Upgrade" in the "SQL Server 2014 Setup" section later in this chapter.)
- Because it is mostly automated, it takes the least amount of 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 2014 Setup can continue.
- Upgrading in place is not recommended for all SQL Server components, such as some DTS packages. See Chapter 17, "Integration Services," for more information about how to upgrade DTS packages.
- Because the new instance of SQL Server 2014 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" later in this section for more information.

The 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.

The 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 3 is intended to give you some guidelines for 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 3: Summary of Factors Affecting the Upgrade Strategy Decision

Consideration	In-Place Upgrade Advantages	Side-by-Side Upgrade Advantages
Require the fewest number of hours for the upgrade deployment team to plan and prepare the upgrade effort	<p>Setup automatically upgrades data and settings in place, without the need for a manual transfer of data or settings.</p> <p>The resulting upgraded instance has the same name as the original.</p> <p>Applications continue to connect to the same instance name.</p>	<p>It gives more granular control over which database objects are migrated.</p> <p>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.</p> <p>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.</p> <p>Applications 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 (see Murphy's Law). If a problem does occur, you will be able to roll back to the legacy system.</p>
Require minimal database administrator (DBA) skill set expertise (or no DBA available to implement the upgrade)	A junior DBA, system engineer, or person with similar basic knowledge and adherence to good IT practices can complete upgrades without special scripts or manual interventions. The Setup	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 the test or quality assurance (QA) department will the users be migrated to the new server.

Consideration	In-Place Upgrade Advantages	Side-by-Side Upgrade Advantages
	program automates the upgrade process.	
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, without user downtime. Some user downtime is still required to update the instance version.
Require the fastest possible revert/rollback in case an issue is encountered		The legacy instance of SQL Server is still present at the end of the upgrade and may be used as a rollback option.
Need the ability to schedule different downtime windows for different user databases within the same instance		You can separately control when each user database is upgraded. After the last user database is upgraded, the legacy instance can be removed.
Need to preserve the server and instance name	The same server and instance names are preserved.	
Have a server consolidation project		The upgraded instance can be placed on the consolidation server, after which the old server can be taken out of service.
Require applications to run in parallel with the original and new instances of SQL Server		You can maintain both systems with production transactions until ready to turn off the original system.
Have new server hardware or operating system		The upgraded instance can be put on the new server, after which the old server can be taken out of service.
Have a shortage of disk space in production	Because there is only one resulting instance, there is less additional data space required than is possible with two complete resulting instances running side by side.	
Have a legacy server that does not meet the SQL Server 2014 Setup requirements for installation	None—not supported.	Only possible with a side-by-side upgrade to a new server that meets the Setup requirements.
Want to change to an earlier edition of SQL Server 2014	None—not supported.	Only possible with a side-by-side upgrade.
Want to upgrade a 32-bit version of SQL	None—not supported.	Only possible with a side-by-side upgrade.

Consideration	In-Place Upgrade Advantages	Side-by-Side Upgrade Advantages
Server 2005/2008/2008 R2 to a 64-bit version of SQL Server 2014		
Have a target instance that is running SQL Server 2000 SP3a and only one downtime window is available		<p>Transferring data to a new instance may be faster than the three steps required:</p> <ul style="list-style-type: none"> • Apply the required SQL Server 2000 service pack. • Upgrade to SQL Server 2005/2008/2008 R2. • Upgrade to SQL Server 2014.
Want to upgrade from a nonclustered legacy instance of SQL Server to a clustered instance of SQL Server 2014	None—not supported.	Only possible with a side-by-side upgrade.
Want to upgrade from SQL Server 7.0	None—not supported. First upgrade to SQL Server 2005 and then to SQL Server 2014.	Can be done in one direct upgrade by using manual data transfer.
Want to upgrade multiple instances	Generally faster because data transfer and configuration steps are handled by Setup.	
Want to upgrade very large databases (VLDBs)	Setup converts existing data files automatically; no data transfer steps are required.	Can control the timing of several steps and the rollback if it is necessary.
Want to perform upgrade testing		<p>Retesting might be easier because the legacy instance of SQL Server does not have to be rebuilt.</p> <p>Testing can be done while the legacy system still supports production applications. SQL Server Profiler can be used to capture SQL commands against the legacy system and to play back against the new instance of SQL Server to verify that everything is working well.</p>
Need to have the data transformed during the upgrade		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 2014.
Need to change SQL Server's localized language		You can upgrade to SQL Server 2014 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.

Consideration	In-Place Upgrade Advantages	Side-by-Side Upgrade Advantages
Need to upgrade Notification Services	None—not supported.	Upgrading Notification Services is only possible with a side-by-side upgrade. It requires the Notification Services backward compatibility add-in.
Need to integrate applications	Integrating applications is simpler because the same server name, instance name, and database security settings are preserved by the Setup program, without manual intervention.	Applications 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. If a problem does occur, you will be able to roll back to the legacy system.
Need to integrate a server	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 2014 are incompatible with legacy instances of SQL Server 2005/2008/2008 R2/2012. Because the new instance of SQL Server 2014 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 2014, remove the data files and other components, reinstall the legacy instance of SQL Server 2005/2008/2008 R2/2012, 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 2014 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 2014 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 2014 to the legacy instance of SQL Server 2005/2008/2008 R2/2012 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 "Test the Upgrade 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 17, "Integration Services." Also, we recommend that you transfer most SSAS components if the source is SQL Server 2000. For more information, see Chapter 16, "Analysis Services."
- **Editions.** The in-place upgrade strategy does not support all paths between editions. For example, to upgrade a SQL Server 2005 Enterprise instance to SQL Server 2014 Standard Edition, you must perform a side-by-side upgrade because the Setup program 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 2014 and leave the rest on the legacy version, you must use a side-by-side upgrade.
- **Upgrading over time.** To transition databases gradually (i.e., several databases at a time) from a legacy instance to SQL Server 2014, 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 2014. In a side-by-side upgrade, SQL Server 2014 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 the "Test the Upgrade Plan" section later in this chapter.

Backward Compatibility

When planning for an upgrade to SQL Server 2014, 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 2014 is backward compatible with SQL Server 2005/2008/2008R2/2012. 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 2014. If the SQL Server 2014 Setup program detects these issues during an in-place upgrade, it will exit the installation, leaving the legacy instance unchanged. The SQL Server 2014 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://technet.microsoft.com/en-us/library/cc707787(v=sql.120).aspx) ([http://technet.microsoft.com/en-us/library/cc707787\(v=sql.120\).aspx](http://technet.microsoft.com/en-us/library/cc707787(v=sql.120).aspx)).

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 2014. If the SQL Server 2012 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 2014 still operate the same as in the legacy versions. However, they may 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 2014. For more information, see [Deprecated SQL Server Features in SQL Server 2014](http://msdn.microsoft.com/en-us/library/cc707789(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/cc707789\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/cc707789(v=sql.120).aspx)). Also see "System Monitor—SQL Server: Deprecated Features Object" 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 sufficient time to resolve the issues before they are discontinued in a future SQL Server release.

Discontinued Features

In any component of SQL Server 2014, some features of earlier SQL Server versions may have been discontinued. These features functioned in earlier versions of SQL Server but may have been removed from SQL Server 2014 or SQL Server 2012. 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 the SQL Server 2014 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 2014](http://msdn.microsoft.com/en-us/library/cc707782(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/cc707782\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/cc707782(v=sql.120).aspx)).

Breaking Changes

Breaking changes to SQL Server 2014 are those that might require changes to the applications because the features in question now have a different behavior. There are no breaking changes between SQL Server 2012 and SQL Server 2014, but if you are upgrading from a version earlier than SQL Server 2012, you may be affected. The best tool for discovering this kind of issue is the SQL Server 2014 Upgrade Advisor, which

analyzes a legacy system and reports on all potential breaking changes and how to address them. For documentation about breaking changes in SQL Server 2014, see [Breaking Changes to SQL Server Features in SQL Server 2014](http://msdn.microsoft.com/en-us/library/cc707784(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/cc707784\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/cc707784(v=sql.120).aspx)). For documentation about breaking changes in SQL Server 2012, see [Breaking Changes to SQL Server Features in SQL Server 2012](http://msdn.microsoft.com/en-us/library/cc707784(v=sql.110).aspx) ([http://msdn.microsoft.com/en-us/library/cc707784\(v=sql.110\).aspx](http://msdn.microsoft.com/en-us/library/cc707784(v=sql.110).aspx)).

Behavior Changes

Behavior changes might not visibly affect your database code or applications. However, you have to be aware of them because the interpretation might be different. For example, the behavior of the SQL Server Native Client changes from SQL Server 2005 to SQL Server 2012. Currently, there are no documented behavior changes to SQL Server features in SQL Server 2014. For more information about the behavior changes for SQL Server 2012, see [Behavior Changes to SQL Server Features in SQL Server 2012](http://msdn.microsoft.com/en-us/library/cc707785(v=sql.110).aspx) ([http://msdn.microsoft.com/en-us/library/cc707785\(v=sql.110\).aspx](http://msdn.microsoft.com/en-us/library/cc707785(v=sql.110).aspx)).

Upgrade Tools

We have talked about the value of the SQL Server 2014 Upgrade Advisor several times already in this chapter. Some other tools are also available to help automate the upgrade process to SQL Server 2014. Each tool has its own purpose and timing, so it is best to become familiar with all the tools and then use those most appropriate to each upgrade project.

SQL Server 2014 Upgrade Advisor

Perhaps the most important tool of the tools typically used for upgrade planning is Upgrade Advisor. Upgrade Advisor eases the transition to SQL Server 2014 by predicting issues in your legacy instances of SQL Server. 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 (T-SQL) scripts and SQL Server Profiler/SQL Trace traces. Upgrade Advisor examines SQL code for syntax that is no longer valid in SQL Server 2014. 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 T-SQL queries, stored procedures, scripts, and application code, see Chapter 10, "Transact-SQL Queries."

Requirements for running Upgrade Advisor are as follows:

- Windows Vista SP1, Windows 7, or Windows Server 2008 R2/2012/2012 R2
- The Microsoft .NET Framework 4 (the same version of the .NET Framework included with SQL Server 2014 and Visual Studio 2010)
- Windows Installer 4.5
- 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 filename as an input parameter.

Note: You can run the SQL Server 2014 Upgrade Advisor with instances of SQL Server SQL Server 2005, 2008, 2008 R2, and 2012. You cannot run it with instances of SQL Server 2000, or 7.0.

The Upgrade Advisor is a separate download. The most recent downloadable version is available as part of the [Microsoft SQL Server 2014 Feature Pack](http://www.microsoft.com/en-us/download/details.aspx?id=42295) (<http://www.microsoft.com/en-us/download/details.aspx?id=42295>). You can find more information about this valuable tool in [Use Upgrade Advisor to Prepare for Upgrades](http://msdn.microsoft.com/en-us/library/ms144256(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms144256\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms144256(v=sql.120).aspx)).

Best Practices Analyzer for SQL Server 2005, 2008, 2008 R2, and 2012

You should run the SQL Server 2012 Best Practices Analyzer (BPA) against your 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 2005 version of BPA from the [SQL Server 2005 Best Practices Analyzer \(August 2008\)](http://www.microsoft.com/en-us/library/ms144256(v=sql.120).aspx) ([http://www.microsoft.com/en-](http://www.microsoft.com/en-us/library/ms144256(v=sql.120).aspx)

us/download/details.aspx?displaylang=en&id=23864) download page.

You can download the SQL Server 2008 R2 BPA from the [SQL Server 2008 R2 Best Practices Analyzer](http://www.microsoft.com/en-us/download/details.aspx?id=15289) (http://www.microsoft.com/en-us/download/details.aspx?id=15289) download page. Use this tool for both SQL Server 2008 and SQL Server 2008 R2.

You can download the SQL Server 2012 BPA from the [SQL Server 2012 Best Practices Analyzer](http://www.microsoft.com/en-us/download/details.aspx?id=29302) (http://www.microsoft.com/en-us/download/details.aspx?id=29302) download page.

SQL Server 2014 Setup: System Configuration Checker

An in-place upgrade uses SQL Server 2014 Setup to directly upgrade SQL Server 2005/2008/2008 R2/2012. The SQL Server 2014 Setup program installs prerequisites such as the .NET Framework and Windows PowerShell 2.0. 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 2012 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://technet.microsoft.com/en-us/library/ms143753(v=sql.120).aspx) (http://technet.microsoft.com/en-us/library/ms143753(v=sql.120).aspx).

SCC looks for conditions that will prevent a successful SQL Server installation or upgrade. These checks occur before the Setup program starts the SQL Server 2014 Installation Wizard. SCC reports any issues that would block an installation along with advice about how to address the blocking issues. It uses rules from the following categories; for more information about any of these categories, see the related links:

- [Installation Rules](http://technet.microsoft.com/en-us/library/cc646015(v=sql.120).aspx)
(http://technet.microsoft.com/en-us/library/cc646015(v=sql.120).aspx)
- [Feature Rules \(Upgrade\)](http://technet.microsoft.com/en-us/library/cc281843(v=sql.120).aspx)
(http://technet.microsoft.com/en-us/library/cc281843(v=sql.120).aspx)
- [Edition Upgrade Rules](http://technet.microsoft.com/en-us/library/cc645998(v=sql.120).aspx)
(http://technet.microsoft.com/en-us/library/cc645998(v=sql.120).aspx)
- [Uninstallation Rules](http://technet.microsoft.com/en-us/library/cc645979(v=sql.120).aspx)
(http://technet.microsoft.com/en-us/library/cc645979(v=sql.120).aspx)

The common, relevant rules—across all four categories—for an in-place upgrade and a side-by-side upgrade, are as follows. 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 destination computer cannot be a domain controller.
- 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 2.0 must be installed. (The Setup program 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 2012/2014 is not supported on the same server with 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 as well as Chapter 4, "High Availability," later in this guide).

For more information about the Setup program, see "SQL Server 2014 Setup" later in this chapter.

Upgrade Assistant for SQL Server 2012 (UAFS)

The Upgrade Assistant for SQL Server 2012 (UAFS) is an external tool that lets you determine in a test environment how an application currently running on SQL Server 2005, 2008, or 2008 R2 will run on SQL Server 2012. This tool uses the SQL Server 2012 Management Tools Distributed Replay (DReplay) utility, together with baseline and trace replays in a test environment, to help identify compatibility issues.

The requirements for using the Upgrade Assistant are as follows:

- Four servers are needed while doing replays: a Production Server, Baseline Server, Test Server, and Report Server.
 - The Production Server contains the databases that will be migrated and the activity that will be replayed.
 - The Baseline Server has the same version of SQL Server as the Production Server with the databases to be migrated on it. It is isolated in order to retrieve baseline information without the effect of other applications or server activity. The Upgrade Assistant for SQL Server 2012 will run its replay against this server, resulting in the baseline trace file.
 - The Test Server has the new SQL Server 2012 instance. If testing an in-place upgrade, this will be the baseline server after the SQL Server 2012 upgrade has been applied. If testing a side-by-side upgrade, this should be a separate server.
 - The Report Server, which has SQL Server 2012 Distributed Replay installed, is used to save trace file comparison results. The Upgrade Assistant for SQL Server 2012 contains step-by-step instructions on how to configure the Distributed Replay utility.
- 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

Using the reports on the Report Server, you can compare the results of the baseline test workload on SQL Server 2012 against the original baseline. If there are any material differences between the two results, you need to work to resolve them in advance of the production upgrade.

For more information and download instructions, see [Upgrade Assistant for SQL Server 2012 \(UAFS\)](http://www.scalabilityexperts.com/tools/downloads.html) (<http://www.scalabilityexperts.com/tools/downloads.html>) on the

Scalability Experts Tools Downloads page.

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 2014 Profiler to trace database activity on a SQL Server 2005/2008/2008 R2/2012 instance under load and save the trace. You can then restore the legacy SQL Server database to two instances on equivalent hardware: an instance of the legacy SQL Server and an instance of SQL Server 2014. Run the replay on each (but at different times if on the same server). While you are running the replay, also run a Profiler trace on each run, capturing errors and query durations. By comparing the results, you can determine whether the upgrade behaves correctly (without error) and performs well.

Note: When using Profiler 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 [Replay Traces](http://technet.microsoft.com/en-us/library/ms190995(v=sql.120).aspx) ([http://technet.microsoft.com/en-us/library/ms190995\(v=sql.120\).aspx](http://technet.microsoft.com/en-us/library/ms190995(v=sql.120).aspx)).

System Monitor—SQL Server: Deprecated Features Object

You can use the SQL Server 2014 System Monitor (Perfmon) counter called SQLServer:Deprecated Features to monitor whether your application is submitting commands to the SQL Server 2014 Database Engine that are scheduled for removal from SQL Server in future releases. You should remove such deprecated commands from SQL Server 2014 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 2014, 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 2012 was last started. For more information about how to use this tool, see [SQL Server, Deprecated Features Object](http://msdn.microsoft.com/en-us/library/bb510662(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/bb510662\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/bb510662(v=sql.120).aspx)).

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 16, "Analysis Services."

Notification Services

You cannot perform an in-place upgrade of SQL Server Notification Services because it is not installed by SQL Server 2014. You must use the SQL Server 2008 R2 Notification Services backward compatibility add-in. See Chapter 9 in the [SQL Server 2008 R2 Upgrade Technical Reference Guide](#)

[Upgrade Technical Reference Guide](#)

(http://download.microsoft.com/download/3/0/D/30DB8D46-8ACF-442A-99A2-0F4CE74AE14D/SQL_Server_2008_R2_Upgrade_Technical_Reference_Guide.docx).

SQL Server 2014 Setup

When planning an upgrade to SQL Server 2014, you first have to make sure that the target servers meet the necessary hardware and software requirements for the setup to be completed.

Note: If your production instance of SQL Server 2005/2008/2008 R2 is installed on Windows Server 2003, you must upgrade to a new server by using a side-by-side upgrade. SQL Server 2012 and 2014 are not supported on Windows Server 2003 and an in-place upgrade is not possible.

Setup Requirements for an In-Place Upgrade

SQL Server 2014 Setup has important version-level requirements for upgrading instances of SQL Server 2005, 2008, 2008 R2, and 2012 in-place. The basic requirements are as follows:

- SQL Server 2005: SP4 is required.
- SQL Server 2008: SP2 is required.
- SQL Server 2008 R2: SP1 is required.
- SQL Server 2012: SP1 is required.

For an in-place upgrade, the target SQL Server 2014 server and the legacy instance of SQL Server 2005/2008/2008 R2/2012 must satisfy some additional requirements:

- Cross-version instances of SQL Server 2014 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 2014. Therefore, you must upgrade all these components together during an in-place upgrade.

- Make sure that sufficient disk space is available for SQL Server 2014 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 [Hardware and Software Requirements for Installing SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms143506(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms143506\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143506(v=sql.120).aspx)).
- Before upgrading SQL Server, enable Windows Authentication for SQL Server Agent and verify the default configuration. (The SQL Server Agent service account must be a member of the SQL Server sysadmin group.)
- Before upgrading from one edition of SQL Server 2014 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 in [Planning a SQL Server Installation](http://msdn.microsoft.com/en-us/library/bb500442(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/bb500442\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/bb500442(v=sql.120).aspx)).
- Cross-platform in-place upgrades from 32-bit to 64-bit (x86 to x64) versions 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 standalone 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. (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 2005/2008/2008 R2/2012 system.

For more information, see the "Unsupported Scenarios" section in [Supported Version and Edition Upgrades](http://msdn.microsoft.com/en-us/library/ms143393(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms143393\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143393(v=sql.120).aspx)).

SQL Server Prerequisites Installed by Setup

The SQL Server 2014 Installation Wizard installs the following prerequisites if they are not already present on the computer:

- .NET Framework 4.0
- SQL Server Native Client
- SQL Server support files

SQL Server 2014 also requires that the .NET Framework 3.5 SP1 components already be installed on the host. If the production system cannot be disturbed in any way before the scheduled downtime for the upgrade process, the SQL Server 2014 Setup program will automatically install the prerequisites as part of the upgrade process. However, this increases the time that is required for the upgrade.

PowerShell 2.0 is required by SQL Server 2014. If the target SQL Server is running Windows Server 2008 SP2, install and enable PowerShell 2.0. If it is running Windows Server 2008 R2 SP1, enable PowerShell 2.0. SQL Server 2014 Setup also installs the SQL Server PowerShell snap-ins.

Note: The `sqlps.exe` command prompt utility for running SQL Server 2008 R2 PowerShell snap-ins has been deprecated.

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 unique identifier specified during the upgrade (or install) to identify that specific SQL Server 2014 installation. The Instance ID behaves similarly to an instance 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 2014 program files, which are located by default at `X:\Program Files\Microsoft SQL Server\MSSQL12.InstanceID`, where X is your system drive, such as drive C.

For a named instance, 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 2014

In this section, we describe the minimum hardware and software requirements for

running SQL Server 2014. For detailed information about the minimum hardware and software requirements for all editions of SQL Server 2014, see [Hardware and Software Requirements for Installing SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms143506(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms143506\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143506(v=sql.120).aspx)). 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 2014 editions:

- SQL Server 2014 Setup will install .NET 4.0, the SQL Server Native Client, and the required SQL Server 2014 Setup support files.
- .NET requirements:
 - If you are installing SQL Server 2014 on a Windows Server 2008 SP2 server, you must download and install .NET Framework 3.5.
 - On Windows 7 and Windows Server 2008 R2 SP1, you must enable .NET Framework 3.5.
 - The .NET 4.0 is a requirement of SQL Server 2014, and SQL Server 2014 Setup will install it during the feature installation step.
 - If you are installing SQL Server 2014 Express on Windows 2008 R1 SP1 Core, you must first install .NET 4.0.
- Windows PowerShell 2.0 is not installed by SQL Server 2014, but it is required. You can download and install Windows PowerShell 2.0 from the [Windows Management Framework](http://support.microsoft.com/kb/968929) site (<http://support.microsoft.com/kb/968929>).

Processor, Memory, and Operating System Memory Requirements

The processor and memory minimum requirements are the same for all the SQL Server 2014 editions. The [Hardware and Software Requirements for Installing SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms143506(v=sql.120).aspx) web page ([http://msdn.microsoft.com/en-us/library/ms143506\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143506(v=sql.120).aspx)) specifies those requirements.

The operating system minimum requirements vary based on the SQL Server 2014 edition you select. For information about the operating system requirements, see the [Hardware and Software Requirements for Installing SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms143506(v=sql.120).aspx) web page ([http://msdn.microsoft.com/en-us/library/ms143506\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143506(v=sql.120).aspx)).

Note the following restrictions or constraints:

- SQL Server 2014 is supported on Windows Server 2008 R2 SP1 and later Server Core installations.
- The Setup program's System Configuration Checker will block the setup if the requirements for processor type and minimum operating system, in addition to other conditions, are not met.

Hard Disk Space Requirements (32-Bit and 64-Bit)

You need to consider the user data in your disk space calculations. If you intend to keep the target and new instances online in parallel for any length of time, you will need double the disk space to store the same data twice.

In addition, there are disk space needs for the SQL Server system files and objects that must be considered in your calculations. Installing SQL Server 2014 requires that the Windows Installer create temporary files on the system drive. Before you run the Setup program to install or upgrade SQL Server, verify that you have at least 4 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 4, from SQL Server 2014 Books Online, shows the minimum disk space requirements for the major SQL Server 2014 components.

Table 4: SQL Server 2014 Disk Space Minimum Requirements

Feature	Disk Space Minimum Requirement
Database Engine and data files, Replication, Full-Text Search, and Data Quality Services	811 MB
Analysis Services and data files	345 MB
Reporting Services and Report Manager	304 MB
Integration Services	591 MB
Master Data Services	243 MB
Client Components	1823 MB
SQL Server 2014 Books Online (downloaded)	375 MB

Extended System Support (WOW64)

SQL Server 2014 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 2014 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 2014 edition on Windows x86, a 64-bit SQL Server 2014 edition on Windows x64, or a 32-bit SQL Server 2014 edition on Windows x64 using WOW64. For more information, see the "Extended System Support" section in [Hardware and Software Requirements for Installing SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms143506(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms143506\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143506(v=sql.120).aspx)).

Installing SQL Server 2014 on a Domain Controller

Although you can install SQL Server 2014 on a domain controller, it is not recommended for the following reasons:

- SQL Server services cannot run under a local service account on a domain controller.
- After SQL Server 2014 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.
- After SQL Server 2014 is installed on a domain controller, it cannot be changed to a domain member unless SQL Server is first uninstalled.
- SQL Server 2014 AlwaysOn failover cluster instances are not supported where the cluster nodes are domain controllers.
- SQL Server 2014 Setup cannot create security groups or provision services accounts on a read-only domain controller, so the setup will fail.

For more information, see the "Installing SQL Server on a Domain Controller" section in [Hardware and Software Requirements for Installing SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms143506(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms143506\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143506(v=sql.120).aspx)).

Product Updates (Slipstreaming) in SQL Server 2014 Setup

SQL Server 2014 uses the slipstreaming capability first developed with SQL Server 2012 Product Updates. Product Updates allows you to include the latest product updates with the main product installation. For example, you can combine the latest SQL Server 2014 Service Pack (SP) or Cumulative Update (CU) in the same file as the Release to Manufacturing (RTM) binaries. This can save you a lot of time and steps. Instead of having to install (or upgrade) to SQL Server 2014 and install patches post-installation, these steps can be combined into one. SQL Server does not ship slipstreamed, but you can create the updated installation media on your own. For instructions on using Product Updates, see [Product Updates in SQL Server 2012 Installation](#)

([http://msdn.microsoft.com/en-us/library/hh231670\(v=SQL.110\).aspx](http://msdn.microsoft.com/en-us/library/hh231670(v=SQL.110).aspx)). For SQL Server 2012 restrictions on SQL Server 2008-style slipstreaming, see "Slipstream Functionality" in [Deprecated SQL Server Features in SQL Server 2012](http://msdn.microsoft.com/en-us/library/cc707789(v=sql.110).aspx) ([http://msdn.microsoft.com/en-us/library/cc707789\(v=sql.110\).aspx](http://msdn.microsoft.com/en-us/library/cc707789(v=sql.110).aspx)).

Allowable Upgrade Paths

We have mentioned that certain versions and components of SQL Server can be upgraded to SQL Server 2014. Now let's be more specific about what versions, components, and upgrade paths are available.

Minimum Versions of SQL Server 2005, 2008, 2008 R2, and 2012

For an in-place upgrade, SQL Server 2014's Setup program requires you to have certain versions of the legacy SQL Server instance. (For more information, see "SQL Server 2014 Setup: System Configuration Checker" earlier in this chapter.) Specifically, the in-place method that is provided by SQL Server 2014 Setup can be used to perform an upgrade for the following versions:

- SQL Server 2005: SP4 is required.
- SQL Server 2008: SP2 is required.
- SQL Server 2008 R2: SP1 is required.
- SQL Server 2012: SP1 is required.

If you have to upgrade earlier versions of SQL Server 2005/2008/2008 R2/2012 and cannot apply the necessary patches, use the side-by-side method.

Upgrading from SQL Server 7.0 to SQL Server 2014

SQL Server 2014 cannot directly upgrade a SQL Server 7.0 instance in-place and cannot restore backups from versions older than SQL Server 2005 SP4. The available options for upgrading from SQL Server 7.0 to SQL Server 2014 include the following:

- Upgrade the instance of SQL Server 7.0 in-place to SQL Server 2005 SP4, and then upgrade the resulting instance in-place to SQL Server 2014. As you can see, this in-place upgrade path requires two steps.
- Upgrade the instance of SQL Server 7.0 to a separate instance of SQL Server 2005 SP4, and then perform an in-place upgrade to SQL Server 2014.

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://www.microsoft.com/en-us/download/details.aspx?DisplayLang=en&id=19471) (<http://www.microsoft.com/en-us/download/details.aspx?DisplayLang=en&id=19471>).

Upgrading SQL Server Components

SQL Server is a complex product, featuring many components that are fairly independent. Table 5 shows the SQL Server 2005 components that you can upgrade and what paths are available. Table 6 and Table 7 show the same information for SQL Server 2008 and SQL Server 2008 R2, respectively. In these tables, the possible strategies from the earlier versions to SQL Server 2014 are the same as to SQL Server 2012.

Table 5: Components and Upgrade Strategies: SQL Server 2005 SP4 to SQL Server 2014

SQL Server 2014 Component	In-Place Upgrade (SQL Server 2005 SP4)	Side-by-Side Upgrade (SQL Server 2005 SP4)
Database Engine	SQL Server Setup (upgrades all databases and preserves server configurations)	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 16, "Analysis Services")
Integration Services	SQL Server Setup	See Chapter 17, "Integration Services"
Reporting Services	SQL Server Setup (for a default configuration only)	Use manual data transfer steps (see Chapter 18, "Reporting Services")
Notification Services	Not Available	See the SQL Server 2005 Upgrade Technical Reference Guide (http://www.microsoft.com/en-us/download/details.aspx?DisplayLang=en&id=19471)

Table 6: Components and Upgrade Strategies: SQL Server 2008 SP2 to SQL Server 2014

SQL Server 2014 Component	In-Place Upgrade (SQL Server 2008 SP2)	Side-by-Side Upgrade (SQL Server 2008 SP2)
Database Engine	SQL Server Setup (upgrades all databases and preserves server configurations when possible)	One or two servers (see Chapter 3, "Relational Databases")

SQL Server 2014 Component	In-Place Upgrade (SQL Server 2008 SP2)	Side-by-Side Upgrade (SQL Server 2008 SP2)
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 16, "Analysis Services")
Integration Services	SQL Server Setup	Manual data transfer (see Chapter 17, "Integration Services")
Reporting Services	SQL Server Setup	Manual transfer of reports (see Chapter 18, "Reporting Services")

Table 7: Components and Upgrade Strategies: SQL Server 2008 R2 SP1 to SQL Server 2012 and 2014

SQL Server 2014 Component	In-Place Upgrade (SQL Server 2008 R2 SP1)	Side-by-Side Upgrade (SQL Server 2008 R2 SP1)
Database Engine	SQL Server Setup (upgrades all databases and preserves server configurations when 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 16, "Analysis Services")
Integration Services	SQL Server Setup	Manual data transfer (see Chapter 17, "Integration Services")
Reporting Services	SQL Server Setup	Manual transfer of reports (see Chapter 18, "Reporting 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 Editions of SQL Server 2005/2008/2008 R2/2012. So, a legacy SQL Server Enterprise database that contains partitioned tables cannot be restored to SQL Server 2014 Standard.

However, there are restrictions on the upgrade path when you upgrade directly. When

you use SQL Server 2014'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://technet.microsoft.com/en-us/library/cc645998(v=sql.120).aspx) (http://technet.microsoft.com/en-us/library/cc645998(v=sql.120).aspx).

Note: The different components of SQL Server 2014 must be kept at the same edition level when they are on the same instance. For example, if you are running the SQL Server 2014 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 Table 1 in Appendix 1, "Version and Edition Upgrade Paths," in this guide.

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 operate 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.

SQL Server 2008 R2 introduced, and SQL Server 2014 continues, collations that are in full alignment with the collations provided by Windows Server 2008 R2 and later. 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 that are linguistically correct for surrogates, support for Chinese minority scripts, the Unicode 5.0 case table, and weights to previously non-weighted characters that would have compared equal.

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:

- Enables consistency across data types in SQL Server because non-Unicode data is converted to Unicode to perform comparison operations.
 - Enables 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 that have 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 to a collation that does not have them. 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 collation 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 belongs 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, you can download the following white papers: [The Impact of Changing Collations and of Changing Data Types from Non-Unicode to Unicode](http://go.microsoft.com/fwlink/?LinkID=113891) (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) (http://go.microsoft.com/fwlink/?LinkID=113890).

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 2005/2008/2008 R2/2012 to localized versions of SQL Server 2014 of the same language.
- You cannot upgrade non-English localized versions of SQL Server to the English-language version of SQL Server 2014.
- You cannot upgrade the English version of SQL Server to any non-English-language localized version of SQL Server 2014. Note: This is a change from SQL Server 2005.
- You cannot upgrade localized versions of legacy SQL Servers to a localized SQL Server 2014 version if the upgrade is 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 2014. For more information, see [Local Language Versions in SQL Server](http://msdn.microsoft.com/en-us/library/ee210665(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ee210665\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ee210665(v=sql.120).aspx)).

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 2014 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 2014 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 2014 version.

SQL Server 2012 and SQL Server 2014 are closely coordinated with the .NET Framework. To use many of the features in SQL Server 2012, SQL Server Native Client (version 11.0) and .NET Framework 3.5 SP1 must be installed on the server and SQL Server 2014 setup will install .NET Framework 4.0 on the server. SQL Server 2012 will automatically install the SQL Server Native Client 11.0, but software developers must upgrade their applications to use .NET Framework 3.5 and 4.0 to take full advantage of SQL Server 2014's new driver and many of its new features. However, it is not required that all applications use .NET Framework 3.5 or 4.0.

SQL Server Native Client 11.0

SQL Server 2014 uses the same native client version that was introduced in SQL Server 2012: SQL Server Native Client, version 11.0. The behavior changes in this version that software developers should be aware of are documented in [Updating an Application](#).

[from SQL Server 2005 Native Client](http://msdn.microsoft.com/en-us/library/bb964722(v=sql.110).aspx) ([http://msdn.microsoft.com/en-us/library/bb964722\(v=sql.110\).aspx](http://msdn.microsoft.com/en-us/library/bb964722(v=sql.110).aspx)).

Applications developed by using OLE DB or ODBC will still work when they connect to SQL Server 2014. Although the overall recommended method for connecting to SQL Server 2014 for non-.NET Framework applications is through the current SQL Server Native Client, you can use the current OLE DB and ODBC drivers until you upgrade the clients. However, features in SQL Server 2005/2008/2008 R2/2012 and SQL Server 2014, 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 Native Client](http://msdn.microsoft.com/en-us/library/ms130828(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms130828\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms130828(v=sql.120).aspx)).

Although DB-Library client tools (such as isql.exe) are no longer supported in SQL Server 2012 and SQL Server 2014, you can still make connections to SQL Server 2008 R2 by using the DB-Library API. However, this feature is deprecated and will be removed in a future version of SQL Server. You should not develop any new application using DB-Library, and you should remove any dependencies on DB-Library from your current applications. For more information, see the note about DB-Library in [Deprecated Database Engine Features in SQL Server 2012](http://msdn.microsoft.com/en-us/library/ms143729(v=sql.110).aspx) ([http://msdn.microsoft.com/en-us/library/ms143729\(v=sql.110\).aspx](http://msdn.microsoft.com/en-us/library/ms143729(v=sql.110).aspx)) in SQL Server 2012 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 2014. However, these applications cannot take advantage of new SQL Server 2014 (or SQL Server 2005/2008/2008 R2/2012) capabilities unless they are upgraded to ADO.NET 3.5 and ADO.NET 4.02, and the SQL Server .NET Managed Data Provider. For example, some 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 SQL Server 2014 data types such as the XML data type, the client applications must 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 2005/2008/2008 R2 to SQL Server 2014. However, the clients cannot take full advantage of features introduced in SQL Server 2014. Some features such as snapshot isolation are available only through T-SQL commands. Also, SQL Server 2005 Notification Services is not available for .NET 1.1 clients and has been removed from

SQL Server starting with SQL Server 2008 R2.

If you upgrade clients from .NET 1.1, 2.0, or 3.0 applications to .NET 3.5 SP1 and .NET 4.0 or later versions, the client applications will be able to take full advantage of features introduced in SQL Server 2014.

For information about SQL Server Native Client, see the following links:

- [SQL Server Native Client Programming](http://msdn.microsoft.com/en-us/library/ms130892(v=sql.120).aspx)
([http://msdn.microsoft.com/en-us/library/ms130892\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms130892(v=sql.120).aspx))
- [Updating an Application from SQL Server 2005 Native Client](http://msdn.microsoft.com/en-us/library/bb964722(v=sql.120).aspx)
([http://msdn.microsoft.com/en-us/library/bb964722\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/bb964722(v=sql.120).aspx))
- [When to Use SQL Server Native Client](http://msdn.microsoft.com/en-us/library/ms130828(v=sql.120).aspx)
([http://msdn.microsoft.com/en-us/library/ms130828\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms130828(v=sql.120).aspx))
- [SQL Server Native Client Features](http://msdn.microsoft.com/en-us/library/ms131456(v=sql.120).aspx)
([http://msdn.microsoft.com/en-us/library/ms131456\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms131456(v=sql.120).aspx))
- [Building Applications with SQL Server Native Client](http://msdn.microsoft.com/en-us/library/ms130904(v=sql.120).aspx)
([http://msdn.microsoft.com/en-us/library/ms130904\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms130904(v=sql.120).aspx))
- [System Requirements for SQL Server Native Client](http://msdn.microsoft.com/en-us/library/ms131002(v=sql.120).aspx)
([http://msdn.microsoft.com/en-us/library/ms131002\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms131002(v=sql.120).aspx))
- [SQL Server Native Client \(OLE DB\)](http://msdn.microsoft.com/en-us/library/ms131687(v=sql.120).aspx)
([http://msdn.microsoft.com/en-us/library/ms131687\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms131687(v=sql.120).aspx))
- [SQL Server Native Client \(ODBC\)](http://msdn.microsoft.com/en-us/library/ms131415(v=sql.120).aspx)
([http://msdn.microsoft.com/en-us/library/ms131415\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms131415(v=sql.120).aspx))
- [Finding More SQL Server Native Client Information](http://msdn.microsoft.com/en-us/library/ms130969(v=sql.120).aspx)
([http://msdn.microsoft.com/en-us/library/ms130969\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms130969(v=sql.120).aspx))

For developer information about SQL Server 2014 data access, see the [Microsoft Data Developer Center](http://msdn.microsoft.com/en-us/data) (<http://msdn.microsoft.com/en-us/data>), which is the primary data access site for Microsoft developer technologies.

Installing .NET Framework on Windows Server 2008 R2

Installing .NET Framework 3.51 on Windows Server 2008 R2 is different than any previous version. If you try to use the standalone installation package that ships with the DVD or that you can download, you will see an error message that you must use the Role Management Tool to install or configure .NET Framework 3.5 SP1. The only way to install .NET Framework 3.51 in Windows Server 2008 R2 is using one of the mechanisms provided by the operating system: the GUI-based Server Manager, a command prompt, or PowerShell. The good thing about this is that installing .NET Framework 3.51 will not require a reboot, maximizing your uptime.

Installing .NET Using Server Manager

If you are not comfortable using a command prompt or PowerShell, you can use Server Manager to configure .NET 3.51:

1. From the Start menu, select Administrative Tools and then Server Manager.
2. Select Features.

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, 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 necessary.
- When the upgrade is complete, but before you do any configurations or changes, perform backups under SQL Server 2014. This lets you easily roll back to a point where the SQL Server 2005/2008/2008 R2/2012 upgrade completed successfully but where an error was introduced after that point.
- After you make any changes to the SQL Server 2014 databases and configuration, but before allowing acceptance testing on the new SQL Server 2014 instance, take full database 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 back up 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 2014 is completed. It is a good practice to perform all of them, just in case an unexpected issue requires a restore at any point in time.

Upgrading Both Windows and SQL Server

Because SQL Server 2014 requires at least Windows Server 2008 R2 SP1, you might be tempted to combine a SQL Server 2014 upgrade with an upgrade from Windows Server 2003/2008 to Windows Server 2008 R2 SP1 or later to reduce the downtime of two major upgrades into one downtime event. However, most Windows Server upgrades require that you reinstall SQL Server. Therefore, it is not possible to upgrade both in-place at the same time.

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 2005/2008/2008 R2 are on Windows Server 2003 or Windows Server 2008, an upgrade of Windows is required because SQL Server 2014 is only supported on Windows Server 2008 R2 SP1 and later. 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 2014.
- If the legacy instances of SQL Server 2000/2005/2008/2008 R2 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 2005/2008/2008 R2 before upgrading to Windows Server 2008 R2 SP1. This process will translate into two outages, but if you upgrade to Windows Server 2008 R2 SP1 first, SQL Server 2000 instances will not be supported.
- If a legacy instance of SQL Server 2000 is running on Windows Server 2000, consider using a multistep process. For example, you might upgrade Windows Server 2000 to Windows Server 2003, then SQL Server 2000 to SQL Server

2005/2008/2008 R2, then Windows Server 2003 to Windows Server 2008 R2 SP1, and then the legacy SQL Server to SQL Server 2014.

- If you are upgrading to Windows Server 2008 R2 SP1 and the server is currently running SQL Server 2005, make sure that you apply SQL Server 2005 SP4 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 and later have an installation option called Server Core. Server Core is basically a locked-down, minimal version of Windows Server that does not have an interface other than a command line. SQL Server 2014 supports Windows Server 2008 R2 and Windows Server 2012 and 2012 R2 Server Core.
- There are two kinds of virtualization with a Windows Server 2008/2008 R2/2012/2012 R2 installation: with Hyper-V and without Hyper-V. SQL Server is supported on both types. As a rule, a server that is used for production database data should be dedicated to SQL Server to reduce the risk of another process bringing the SQL Server down. Unless a server must run virtual machines that will host SQL Server, install Windows Server 2008/2008 R2/2012/2012 R2 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 2008/2008 R2/2012/2012 R2 is installed with Hyper-V, apply the final Hyper-V RTM patch. You can find more information about Hyper-V for Windows Server 2012 R2 in the following resources:
 - [What's New in Hyper-V for Windows Server 2012 R2](http://technet.microsoft.com/en-us/library/dn282278.aspx)
<http://technet.microsoft.com/en-us/library/dn282278.aspx>
 - Windows Server 2012 R2, [Server Virtualization](http://www.microsoft.com/en-us/server-cloud/products/windows-server-2012-r2/#s1section2)
<http://www.microsoft.com/en-us/server-cloud/products/windows-server-2012-r2/#s1section2>
- You cannot use an old Windows NT-style domain for a Windows Server 2008 or later. You must be using Active Directory. If your organization is using older-style domains, you cannot deploy Windows Server 2008/2008 R2/2012 until your Active Directory infrastructure is upgraded.

- By default, Windows Server 2008 and later versions are more secure than Windows Server 2000 and Windows Server 2003. Many of the features in Windows Server 2000 and Windows Server 2003 are not configured, so there will be additional work involved in the upgrade.
- Windows Server 2008 and later versions support only Serial Attached SCSI (SAS), Fibre Channel, and iSCSI. If you are using older parallel SCSI, you have to upgrade your disk solutions to support Windows Server 2008 or later.

For more Windows-specific information about how to deploy and upgrade to Windows Server 2008/2008 R2/2012/2012 R2, see:

- [Upgrading to Windows Server 2008](http://technet.microsoft.com/en-us/library/cc754728(WS.10).aspx)
([http://technet.microsoft.com/en-us/library/cc754728\(WS.10\).aspx](http://technet.microsoft.com/en-us/library/cc754728(WS.10).aspx))
- [Windows Server 2008 R2 Upgrade Paths](http://technet.microsoft.com/en-us/library/dd979563(WS.10).aspx)
([http://technet.microsoft.com/en-us/library/dd979563\(WS.10\).aspx](http://technet.microsoft.com/en-us/library/dd979563(WS.10).aspx))
- [Upgrade Options for Windows Server 2012 R2](http://technet.microsoft.com/en-us/library/dn303416.aspx)
(<http://technet.microsoft.com/en-us/library/dn303416.aspx>)

For information about how to upgrade to Windows Server 2008/2008 R2/2012/2012 R2 failover clustering with SQL Server 2014, see Chapter 4, "High Availability."

Upgrading Multiple Instances

You might have multiple instances of SQL Server 2005/2008/2008 R2/2012 on a standalone 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 T-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 need to review Chapter 4's 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 in 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 2005/2008/2008 R2/2012 are at the correct service-pack level for an in-place upgrade. If you are upgrading side-by-side, make sure that the backups from the legacy instance will be restorable on SQL Server 2014. If they are not at the correct level, plan to update them before the upgrade.
- **Make sure that installation requirements are met.** SQL Server 2014 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, install .NET Framework 3.5 SP1 or a later version and 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 2014 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 2014 common components.** Install the SQL Server 2014 common components (such as the SQL Server 2012 SQL Native Client) before the upgrade. Also consider pre-installing the Management Tools if SSMS 2012 is not required to manage SQL Server 2000 instances. (For more information, see Chapter 2, "Management Tools.")
- **Select the optimal side-by-side upgrade strategy.** In a side-by-side upgrade, SQL Server databases must be transferred during the upgrade. 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 16, "Analysis Services").
- **Use new service accounts.** Create and use new service accounts and groups, if it is necessary, with your SQL Server 2014 implementations. This guarantees account separation from the legacy instances of SQL Server 2005/2008/2008 R2/2012. 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 2014 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 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 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.
- Take 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.
- Take 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 upgrade, 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 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 2014 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 2014, make sure that those people administering or deploying SQL Server 2014 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 2014 environment so that everyone who has to update his or her skill set can become familiar with the new version. If the DBAs are comfortable with SQL Server 2014 by the time of production deployment, the transition from SQL Server 2005/2008/2008 R2/2012 will go much more smoothly.

There are many resources that are available to update skills. To start, see the [SQL Server Developer Center](http://msdn.microsoft.com/en-us/sqlserver) (<http://msdn.microsoft.com/en-us/sqlserver>). Also you can get initial experience with SQL Server 2014 using the training material in the [TechNet Virtual Labs](http://technet.microsoft.com/en-us/virtuallabs?id=J+avMPtWIXk) (<http://technet.microsoft.com/en-us/virtuallabs?id=J+avMPtWIXk>).

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 also 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.
- **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 "Create 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 2014 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 2014 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 2005/2008/2008 R2/2012, and then restoring the SQL Server legacy 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 Analysis Services. 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 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. Conversely, the more complex, the more likely you will want to make the additional effort to ensure a successful upgrade outcome.

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 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. If you choose a side-by-side upgrade, you can change edition level in any direction. 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 2005/2008/2008 R2/2012, the result will be a default instance of SQL Server 2014. However, if you choose a side-by-side upgrade on 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 2014 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 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 2014. (For more information, see the "Upgrading Both Windows and SQL Server" section.)

Note: If your legacy instance of SQL Server 2005/2008/2008 R2/2012 is installed on Windows Server 2003 or 2008, you must upgrade to a new server by using a side-by-side upgrade because SQL Server 2014 is only supported on Windows Server 2008 R2 SP2 or later.

- **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 that the fewer changes 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 2014 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, "SQL Server 2014 Upgrade Planning 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, the checklist for upgrading the lowest-level instances of SQL Server would be much simpler.
- 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 contain many topics that contain valuable information that can help in developing upgrade checklists:

- Database Engine: See [Upgrade Database Engine](http://msdn.microsoft.com/en-us/library/bb933942(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/bb933942\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/bb933942(v=sql.120).aspx)) and Chapter 3, "Relational Databases."
- Management tools: See Chapter 2, "Management Tools."
- Clustered environments: See [Upgrade a SQL Server Failover Cluster](http://msdn.microsoft.com/en-us/library/ms191009(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms191009\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms191009(v=sql.120).aspx)) and Chapter 4, "High Availability."

- Replication: See [Upgrade Replicated Databases](http://msdn.microsoft.com/en-us/library/ms143699(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms143699\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143699(v=sql.120).aspx)) and the section on replication in Chapter 4, "High Availability."
- Analysis Services: See [Upgrade Analysis Services](http://msdn.microsoft.com/en-us/library/ms143686(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms143686\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143686(v=sql.120).aspx)) and Chapter 16, "Analysis Services."
- Integration Services and DTS: See Chapter 17, "Integration Services." For instructions on upgrading DTS to SQL Server 2008 Integration Services, see [Considerations for Upgrading Data Transformation Services](http://msdn.microsoft.com/en-us/library/ms143716.aspx) (<http://msdn.microsoft.com/en-us/library/ms143716.aspx>).
- Reporting Services: See [Upgrade and Migrate Reporting Services](http://msdn.microsoft.com/en-us/library/ms143747(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms143747\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143747(v=sql.120).aspx)) and Chapter 18, "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. Make a disk image of the database server, in addition to copies of all data files. 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 2014 drivers on the production server by using SQL Server 2014 Setup before the upgrade process, if that is possible and it matches the upgrade plan for production. This saves time during the actual upgrade.
- Execute Upgrade Advisor remotely against the test server that runs the legacy instance of SQL Server 2005/2008/2008 R2/2012 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. For more information, see the "SQL Server 2014 Upgrade Advisor" section 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 90 if you upgraded from SQL Server 2005, or 100 if you upgraded from SQL Server 2008/2008 R2, and 110 if you upgraded from SQL Server 2012. 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 2014.
- 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 2005/2008/2008 R2/2012 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 2014. After you install the parallel instance of SQL Server 2014, 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 2014 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 2014 and verify that the legacy version of SQL Server 2005/2008/2008 R2/2012 is working correctly.
- In a side-by-side upgrade, the rollback will most likely be to the original instance of SQL Server 2005/2008/2008 R2/2012. 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 2014. 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 2014 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 2014. If this is the case, test the effect that SQL Server 2014 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 2014 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 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:

- Integrate the new SQL Server instance into the application and database server environment.
- Determine 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 2014 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 2014. 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 of 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 2014 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 2014 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.

Run the SQL Server 2012 Best Practices Analyzer

To ensure that your newly upgraded SQL Server 2014 components satisfy known SQL Server best practices, download and run the [SQL Server 2012 Best Practices Analyzer](http://www.microsoft.com/en-us/download/details.aspx?id=29302) (<http://www.microsoft.com/en-us/download/details.aspx?id=29302>). This will ensure that your databases and components comply with current SQL Server best practices.

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, as well as 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 2014 installation. The SQL Server 2014 Setup program logs a summary of its actions in the Summary.txt file in the Program Files\Microsoft SQL Server\120\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 under 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://msdn.microsoft.com/en-us/library/ms144287.aspx) (<http://msdn.microsoft.com/en-us/library/ms144287.aspx>).

- 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 2014 Setup.
- The steps for troubleshooting a side-by-side upgrade are 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, so 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 2014 environment. This ensures 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 2005/2008/2008 R2/2012, 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 2014 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 2014 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 2014 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.

The flowchart in Figure 4 describes the different stages in upgrading your instance of SQL Server 2005/2008/2008 R2/2012 to SQL Server 2014. The chart also describes basic tools available to help you upgrade to SQL Server 2014.

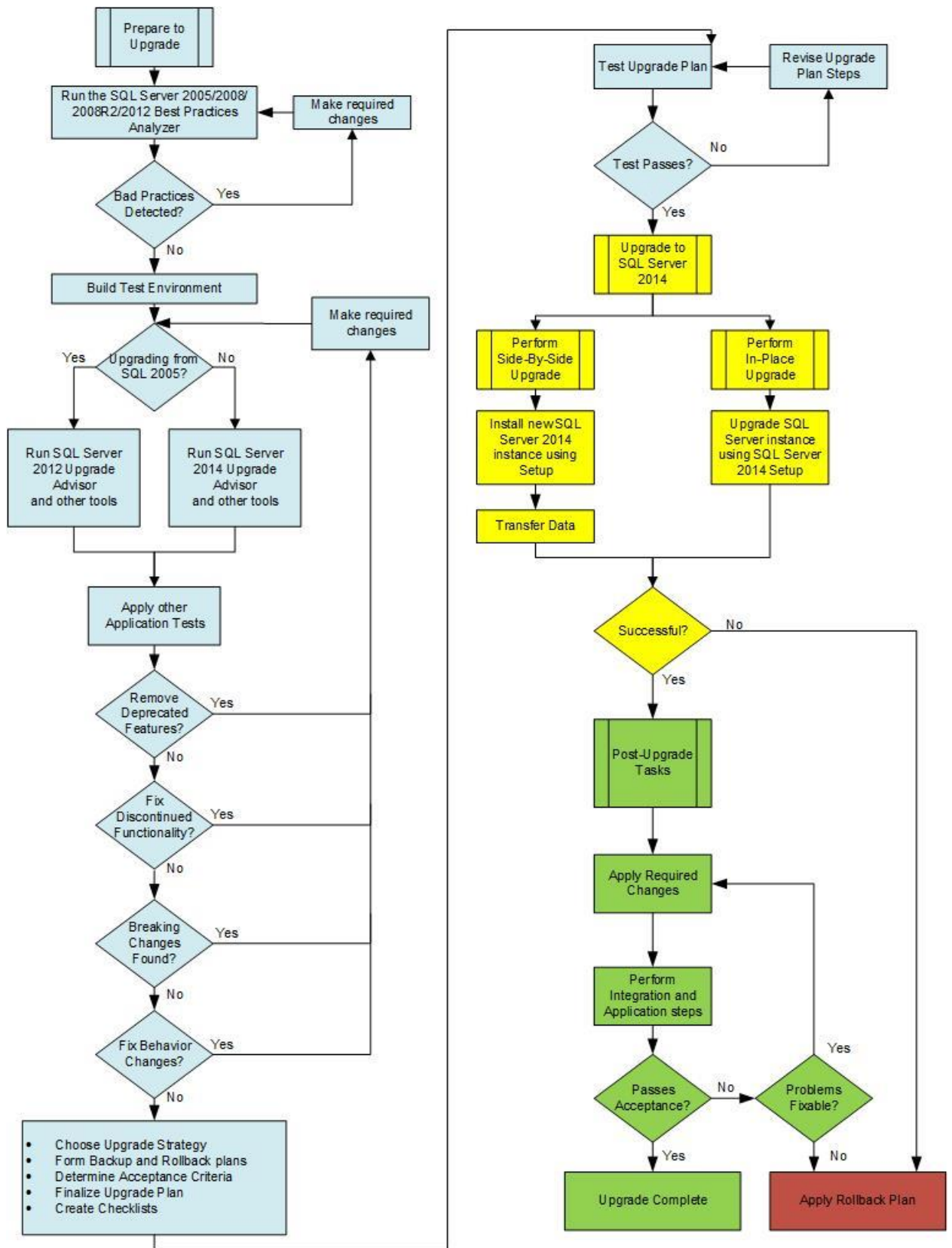


Figure 4: SQL Server 2014 upgrade flowchart

Conclusion

The key steps in planning for a successful SQL Server 2014 upgrade are as follows:

- Research the upgrade requirements.
- Establish pre-upgrade tasks.
- Establish upgrade implementation tasks.
- Establish 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 2014 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 2014 upgrade planning and deployment references, see the following links:

- [Upgrade to SQL Server 2014](http://technet.microsoft.com/en-us/library/bb677622.aspx)
(<http://technet.microsoft.com/en-us/library/bb677622.aspx>)
- [SQL Server 2014 Server and Cloud Platform](http://www.microsoft.com/en-us/server-cloud/products/sql-server/)
(<http://www.microsoft.com/en-us/server-cloud/products/sql-server/>)
- [Books Online for SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms130214(v=sql.120).aspx)
([http://msdn.microsoft.com/en-us/library/ms130214\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms130214(v=sql.120).aspx))
- [SQL Server Developer Center](http://msdn.microsoft.com/en-us/sqlserver/)
(<http://msdn.microsoft.com/en-us/sqlserver/>)
- [SQL Server TechNet Resources](http://technet.microsoft.com/en-us/sqlserver/)
(<http://technet.microsoft.com/en-us/sqlserver/>)

For more information about how to upgrade each SQL Server component (e.g., Analysis Services, Integration Services) see the remaining chapters in this guide.

Chapter 2: Management Tools

Introduction

This chapter covers how to handle SQL Server management tools when you upgrade to SQL Server 2014.

For information about the effect of upgrading to other SQL Server tools, see in this Guide:

- For the business intelligence tools, see Chapter 15, “Business Intelligence Tools,” in this Guide.
- For the Reporting Services tools, see Chapter 18, “Reporting Services,” in this Guide and [Upgrade and Migrate Reporting Services](http://msdn.microsoft.com/en-us/library/ms143747(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms143747\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143747(v=sql.120).aspx)).

Upgrading to SQL Server 2014 and SQL Server’s Management Tools

There are no feature changes for the SQL Server 2014 management tools that will affect your upgrade process. Transitioning to using the new tools is therefore seamless.

These tools include:

- SQL Server Management Studio (SSMS)
- SQL Server Configuration Manager
- Database Engine Tuning Advisor
- SQL Server Profiler

Upgrading In-Place

When you are upgrading in-place from SQL Server 2005 to SQL Server 2012, Setup will require that you upgrade all SQL Server components (e.g., Database Engine, Analysis Services, Integration Services, etc.)

SQL Server 2014 Setup will *not* replace the SQL Server 2005/2008/2008R2/2012 tools with SQL Server 2014 tools:

- When you upgrade to SQL Server 2014 from SQL Server 2005/2008/2008R2/2012, the older versions of these tools will remain on the server. The upgrade has no impact on these tools and they remain usable.

However, to manage SQL Server 2014 instances, be sure that you use the SQL 2014 versions of these tools.

- The legacy SQL Server management tools on the upgraded instance must be removed separately by using Add/Remove Programs. This applies whether there are one or many legacy instances of SQL Server on the server.
- SQL Server 2014 Setup will detect the registered servers and other settings of the legacy SQL Server tools and attempt to preserve them when upgrading to SQL Server 2014.
- Exporting and importing SSMS registered servers between SQL Server 2005 and SQL Server 2014 is not supported because the underlying .regsvr files do not have the same structure.

Tools Connectivity

Legacy SQL Server 2005/2008/2008R2/2012 tools should not, and in some cases cannot be used to manage SQL Server 2014 instances. You should use only SQL Server 2014 tools and utilities to manage SQL Server 2014 instances and any legacy SQL Server 2005/2008/2008R2/2012 instances. To manage SQL Server instances remotely, install the SQL Server 2014 tools on a workstation to ensure that you can manage SQL Server 2014 after it is installed.

SQL Server Management Studio (SSMS)

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

You should always use SSMS 2014 to manage SQL Server 2014 instances:

- SSMS 2014 can manage your legacy SQL Server 2005, 2008, 2008R2, and 2012 instances.
- Earlier versions of SSMS, such as SSMS 2012, may connect to SQL Server 2014 instances, but they may not correctly display any new SQL Server 2014 features
- Therefore your best option is to upgrade your client software to SSMS2014 even before you start the server-based upgrade process.

Upgrading Master and Target Servers

DBAs upgrading multi-server environments need to upgrade all target servers (TSX) before upgrading master servers (MSX). You can use SSMS to re-enlist target servers by following these steps:

3. In Object Explorer, connect to an instance of the Database Engine and expand that instance.
4. 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.

Two features introduced in SQL Server 2008 R2 make working in a distributed environment more cost-effective than using master and target servers. Those features are:

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

For information about these features, see the following SQL Server 2014 topics:

- [Administer Servers by Using Policy-Based Management](http://msdn.microsoft.com/en-us/library/bb510667(v=sql.120).aspx)
([http://msdn.microsoft.com/en-us/library/bb510667\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/bb510667(v=sql.120).aspx))
- [Administer Multiple Servers Using Central Management Servers](http://msdn.microsoft.com/en-us/library/bb895144(v=sql.120).aspx)
([http://msdn.microsoft.com/en-us/library/bb895144\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/bb895144(v=sql.120).aspx))

Project Files

SSMS 2014 can open and work with project files created in previous versions of SSMS. When the project is opened with SSMS 2014, it must be upgraded with a Visual Studio conversion process. This upgrade process is a function of the Visual Studio Shell upon which the SSMS 2014 is built. After the project is upgraded, it cannot be opened by any of the previous versions of SSMS. For more information, see [Projects \(SQL Server Management Studio\)](http://msdn.microsoft.com/en-us/library/hh231442(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/hh231442\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/hh231442(v=sql.120).aspx)).

Post-Upgrade SSMS Tasks

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

Database Maintenance Plans

Maintenance plans create a workflow of the tasks that are required to ensure that your databases are optimized, backed up, and free of inconsistencies. These plans can be created either manually or by using the Maintenance Plan Wizard. The wizard lets you create plans for almost any SQL Server-related maintenance task while taking advantage of the Integration Services designer to extend the functionality of the maintenance plans. To gain the advantages of the new maintenance plan functionality, you need to transfer your maintenance plans to SQL Server 2014. This transfer can be done automatically or manually.

For details about the Maintenance Plan Wizard, see [Use the Maintenance Plan Wizard](http://msdn.microsoft.com/en-us/library/ms191002.aspx) (<http://msdn.microsoft.com/en-us/library/ms191002.aspx>) in SQL Server 2012 Books Online.

Database Diagrams

During an in-place upgrade to SQL Server 2014, database diagrams created in earlier releases of SQL Server will automatically be upgraded.

For details about upgrading database diagrams, see [Upgrade Database Diagrams from Previous Editions \(Visual Database Tools\)](http://msdn.microsoft.com/en-us/library/ms190628(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms190628\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms190628(v=sql.120).aspx)).

After an in-place upgrade, SQL Server 2014 will preserve your settings in SSMS. However, in a side-by-side upgrade, you will need to customize the tools to have the same functionality as before. This includes:

- Reregistering your servers after the upgrade.
- Customizing the layout of SSMS 2014.

Azure-Facing Features in SSMS 2014

There are a number of new Azure-facing features in SSMS 2014. For more information, see the following:

- [Deploy a SQL Server Database to a Windows Azure Virtual Machine](http://msdn.microsoft.com/en-us/library/dn195938.aspx) (<http://msdn.microsoft.com/en-us/library/dn195938.aspx>)
- [SQL Server Backup and Restore with Microsoft Azure Blob Storage Service](http://msdn.microsoft.com/en-us/library/jj919148.aspx) (<http://msdn.microsoft.com/en-us/library/jj919148.aspx>)

SQL Server Data Tools

There are no changes to the SQL Server 2014 version of the SQL Server Data Tools that will affect the upgrade to SQL Server 2014. However, you should download the new plug-in to create SQL Server 2014 projects inside SSDT:

- [Microsoft SQL Server Data Tools](http://msdn.microsoft.com/en-us/data/tools.aspx) (http://msdn.microsoft.com/en-us/data/tools.aspx)

Conclusion

You can use the SQL Server 2014 management tools to manage your legacy SQL Server 2005/2008/2008R2/2012 instances. Because there are no breaking or behavioral changes from the SQL Server 2012 versions of those tools, transitioning to the new tools is seamless.

However, you should always use the SQL Server 2014 management tools to manage your SQL Server 2014 instances. As a result, consider installing and using SQL Server 2014 management tools on your desktop even before you begin the server upgrades to SQL Server 2014.

Chapter 3: Relational Databases

Introduction

Customers currently using SQL Server 2005/2008/2008 R2/2012 have several options for upgrading their relational databases to SQL Server 2014. The best approach depends on how the legacy SQL Server 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 2005/2008/2008 R2/2012 can be deployed in a variety of 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 many 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:

- [Upgrade Database Engine](http://msdn.microsoft.com/en-us/library/bb933942(v=sql.120).aspx) (http://msdn.microsoft.com/en-us/library/bb933942(v=sql.120).aspx) provides a valuable overview of the upgrade process and covers many high-level details to consider before you start the upgrade process.
- For upgrading 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 [Upgrade a SQL Server Failover Cluster Instance \(Setup\)](http://msdn.microsoft.com/en-us/library/ms191295(v=sql.120).aspx) (http://msdn.microsoft.com/en-us/library/ms191295(v=sql.120).aspx).
 - For log shipping upgrade information, see [About Log Shipping \(SQL Server\)](http://msdn.microsoft.com/en-us/library/ms187103(v=sql.120).aspx) (http://msdn.microsoft.com/en-us/library/ms187103(v=sql.120).aspx).

- For mirroring upgrade information, see [Minimize Downtime for Mirrored Databases When Upgrading Server Instances](http://msdn.microsoft.com/en-us/library/bb677181(v=sql.120).aspx) (http://msdn.microsoft.com/en-us/library/bb677181(v=sql.120).aspx).
- For replication upgrade information, see [Upgrade Replicated Databases](http://msdn.microsoft.com/en-us/library/ms143699(v=sql.120).aspx) (http://msdn.microsoft.com/en-us/library/ms143699(v=sql.120).aspx).
- For full-text upgrade information from SQL Server 2008/2008 R2/2012, see Chapter 6, "Full-Text Search."
- For SQL Server Express upgrades, see Chapter 8, "SQL Server Express."

Upgrade Considerations

Before upgrading from SQL Server 2005/2008/2008 R2/2012 to SQL Server 2014, you need 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 2005 Service Pack 4 (SP4) or later, SQL Server 2008 SP3 or later, SQL Server 2008 R2 SP2 or later, and SQL Server 2012 SP1 or later can be upgraded to SQL Server 2014. For comprehensive information about which versions and editions can be upgraded, see [Supported Version and Edition Upgrades](http://msdn.microsoft.com/en-us/library/ms143393(v=sql.120).aspx) (http://msdn.microsoft.com/en-us/library/ms143393(v=sql.120).aspx).
- To minimize potential problems when a database is upgraded to SQL Server 2014, the database keeps its existing compatibility level (except for system databases, which have a 120 compatibility level). If you upgrade a database from SQL Server 2005, it will have an initial compatibility level of 100. If you upgrade from SQL Server 2008 or 2008 R2, the database will have an initial compatibility level of 100. If you upgrade from SQL Server 2012, the database will have an initial compatibility level of 110. Before you change the compatibility level of an upgraded database to 120, you must assess how the change might affect your applications. For specific compatibility-level guidance, see [ALTER DATABASE Compatibility Level \(Transact-SQL\)](http://msdn.microsoft.com/en-us/library/bb510680(v=sql.120).aspx) (http://msdn.microsoft.com/en-us/library/bb510680(v=sql.120).aspx).
- On a server that contains multiple legacy instances of the SQL Server Database Engine, you must upgrade each instance individually. Upgrading one instance has no effect on other instances on the same server. From the SQL Server 2014

point of view, you can upgrade multiple instances in any order and at any time.

- You cannot perform a direct, in-place upgrade from a 32-bit edition of SQL Server 2005/2008/2008 R2/2012 to any 64-bit edition of SQL Server 2014. However, databases from a legacy 32-bit edition can be restored on or attached to a 64-bit edition of SQL Server 2014, 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 2014. For comprehensive information about how to upgrade SQL Server tools, see Chapter 2, "Management Tools."

SQL Server 2014 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 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 2014](http://msdn.microsoft.com/en-us/library/ms143506(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms143506\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143506(v=sql.120).aspx)).

Full-Text Search

SQL Server 2008 added numerous improvements to full-text search, including the following:

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

SQL Server 2012 added:

- Significantly greater scalability
- Improvements to CONTAINS and NEAR
- Significant performance improvements on larger data sets

For information on how to upgrade full-text search to SQL Server 2014, see Chapter 6, "Full-Text Search."

What Can Be Upgraded?

There are a series of paths for upgrading legacy SQL Server 2005/2008/2008 R2/2012 relational databases to SQL Server 2014.

Versions

As mentioned previously, you can upgrade the following versions to SQL Server 2014:

- SQL Server 2005 SP4 or later
- SQL Server 2008 SP3 or later
- SQL Server 2008 R2 SP2 or later
- SQL Server 2012 SP1 or later

You can also upgrade SQL Server 2014 Community Technology Preview 2 (CTP2) to SQL Server 2014. For more information about the 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 2014. You can also upgrade the management tools, full-text search, Data Quality Services, Analysis Services, and Reporting Services components. For more information about each of these topics, see the following chapters in this guide:

- Chapter 2, "Management Tools"
- Chapter 6, "Full-Text Search"
- Chapter 16, "Analysis Services"
- Chapter 18, "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 to or later than your current edition. For example, you can upgrade from SQL Server 2008 Standard to SQL Server 2014 Standard, Enterprise, or Business Intelligence Edition.

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. 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 [Supported Version and Edition Upgrades](http://msdn.microsoft.com/en-us/library/ms143393(v=sql.120).aspx) (http://msdn.microsoft.com/en-us/library/ms143393(v=sql.120).aspx).

Note also that SQL Server 2014 is not supported on Windows 2008 and earlier operating systems. For a complete list of supported operating systems, see [Hardware and Software Requirements for Installing SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms143506(v=sql.120).aspx) (http://msdn.microsoft.com/en-us/library/ms143506(v=sql.120).aspx).

Cross-Language Support

There is broad support for localization within SQL Server 2014. The US English version of SQL Server 2014 is supported on all localized versions of supported operating systems. If you are running a localized version of SQL Server 2005/2008/2008 R2/2012, you can perform an in-place upgrade to SQL Server 2014 as long as the target instance and operating system are of the same language as your original SQL Server instance. You can run a localized instance of SQL Server 2014 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 [Supported Version and Edition Upgrades](http://msdn.microsoft.com/en-us/library/ms143393(v=sql.120).aspx) (http://msdn.microsoft.com/en-us/library/ms143393(v=sql.120).aspx).

What Cannot Be Upgraded?

Certain paths for upgrading legacy SQL Server 2005/2008/2008 R2/2012 relational databases to SQL Server 2014 are not allowed.

Versions

Although SQL Server 2000 and earlier versions are not directly upgradeable to SQL Server 2014, there are options available to upgrade these databases indirectly. For an in-place upgrade, you can upgrade the SQL Server 2000 instance to SQL Server 2005/2008/2008 R2, and then to SQL Server 2014.

Instances of SQL Server 2005 with service pack levels less than SP4 cannot be upgraded in-place. However, individual SQL Server 2005 databases can be upgraded by using the

side-by-side upgrade model, even if SQL Server 2005 is at the release to manufacturing (RTM) level.

Components

Some components have either been deprecated or removed in SQL Server 2014. For example, Notification Services was removed from SQL Server 2008 R2 and therefore is not available in SQL Server 2014. In addition, Data Transformation Services (DTS) was replaced by SQL Server Integration Services (SSIS). For more information about DTS and SSIS, see Chapter 17, "Integration Services."

Evaluation Editions

Although Microsoft has made a great effort to support as many in-place upgrade paths as possible, you cannot upgrade SQL Server 2005/2008/2008 R2/2012 Evaluation Editions to SQL Server 2014. For more information, see [Supported Version and Edition Upgrades](http://msdn.microsoft.com/en-us/library/ms143393(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms143393\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143393(v=sql.120).aspx)).

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 [Supported Version and Edition Upgrades](http://msdn.microsoft.com/en-us/library/ms143393(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms143393\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143393(v=sql.120).aspx)).

Additional Limitations on Upgrades

Be aware that when you perform an in-place upgrade, you cannot add features or make configuration changes. If you need 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 2005/2008/2008 R2/2012 instance to SQL Server 2014, 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 2005/2008/2008 R2/2012.

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 2014 instance that was created during the failed upgrade attempt.
3. Restart the server.
4. Reinstall the earlier version of SQL Server (SQL Server 2005/2008/2008 R2/2012).
5. Reinstall any required SQL Server service packs to your legacy SQL Server 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 will be required if you must roll back an in-place upgrade.

Side-by-Side Upgrade

With a side-by-side upgrade, the SQL Server 2014 relational Database Engine is installed as a second instance and the original SQL Server 2005/2008/2008 R2/2012 relational Database Engine remains in place. Then you move or copy one or more legacy user databases to the SQL Server 2014 instance (each moved database is automatically upgraded).

During the side-by-side upgrade process, users can continue to access the legacy SQL Server relational Database Engine and its databases (which are unaffected by the upgrade process) while the new SQL Server 2014 instance 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 2014 instance, or you may lose data. After the side-by-side upgrade is complete, the SQL Server 2014 relational Database Engine and the legacy SQL Server relational Database Engine co-exist. After you verify the SQL Server 2014 relational Database Engine, you can let applications access the new server. After the SQL Server 2014 relational Database Engine is in production, you can uninstall SQL Server 2005/2008/2008 R2/2012 from the old server, or else just decommission the server (e.g., take it off the network and rebuild the operating system).

A side-by-side upgrade can require much more effort than an in-place upgrade. In an in-place upgrade, SQL Server 2014 Setup makes sure that the new SQL Server 2014 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 2014 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 2014 on a separate computer or as a separate instance on the same computer as part of the upgrade to SQL Server 2014, 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 2014 cannot be restored or attached to an earlier version of SQL Server. You could extract the data out of the SQL Server 2014 instance and import it to the legacy instance. Whatever the case, if an in-place upgrade fails (perhaps 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 re-created manually. For further information, see [Use the Copy Database Wizard](http://msdn.microsoft.com/en-us/library/ms188664(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms188664\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms188664(v=sql.120).aspx)).

Important: If you update any data in the legacy instance of SQL Server 2005/2008/2008 R2/2012 while you are testing an upgraded database in an instance of SQL Server 2014, 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 2014 forward in time, you must bring the new data from the legacy SQL Server instance forward by using some 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 2014 relational Database Engine and then move one or more user databases to this instance, be aware that the following SQL Server 2005/2008/2008 R2/2012 relational database features are disabled by default in new installations:

- Ad hoc distributed queries
- SQL Mail
- Named pipes
- xp_cmdshell

To enable some or all of these features, configure the server properties in SQL Server Management Studio (SSMS) or use the `sp_configure` stored procedure.

Important: After upgrading to SQL Server 2014, always run `sp_updatestats` in the upgraded databases to ensure optimal use of improved statistics features in SQL Server 2014.

If you opt to use the side-by-side method of upgrading to the SQL Server 2014 relational Database Engine, you have several methods to choose from, including the following:

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

Let's take a look at each of these methods.

Backup/Restore

You can upgrade a SQL Server 2005/2008/2008 R2/2012 relational database by performing a database backup of that database and then restoring it to a SQL Server 2014 instance. You can create a database backup by using SSMS or by executing a Transact-SQL (T-SQL) or PowerShell script. You can then restore this database backup on the SQL Server 2014 instance by using SSMS or a T-SQL script. The database metadata will automatically be upgraded as it is restored. You may then change the compatibility level to 120, the level for SQL Server 2014.

This side-by-side upgrade method lets you leave the legacy SQL Server 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 T-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 the local disk instead of offline to other storage, the destination SQL Server will need additional disk space to accommodate the backup file in addition to the original database files.

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 2014 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, stop all activity on the legacy databases, back up the transaction logs on the legacy SQL Server 2005/2008/2008 R2/2012 instance, and restore the transaction logs on the SQL Server 2014 instance with RECOVERY. After testing and acceptance, you can allow updating to the new instance.

Detach/Attach

You can upgrade a SQL Server 2005/2008/2008 R2/2012 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 2014 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 (e.g., connection string changes, modifications required by the Database Engine upgrade, T-SQL changes).

Note: In many SQL Server databases, a significant amount of unused space may be reserved in the data and transaction log files. This is by design to minimize how often 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: During a side-by-side upgrade, if you allow data updates to the SQL Server 2005/2008/2008 R2/2012 instance while you are testing an upgraded database in a SQL Server 2014 instance, the databases will not remain synchronized. Data changes that you make to the existing database will not be made to the upgraded database. (This is true not only for the detach/attach method but also any of the other methods discussed in the “Side-by-Side Upgrade” section.) You must perform another detach/attach upgrade of the database when you are ready to switch to SQL Server 2014. 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 them from these backups.

When you move a very large database (VLDB) from one instance to another on the same server, the detach/attach method 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 Storage Area Network (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 2014. These options will save disk space and might save database administrators (DBAs) 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 recreate that clone on another disk array, which you can then attach to the SQL Server 2014 relational database instance for upgrade. DBAs 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 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 2014, you cannot reattach it to an earlier version of the SQL Server relational Database Engine.

Manual Schema Rebuild and Data Export/Import

Another way you can perform a side-by-side upgrade of a SQL Server 2005/2008/2008 R2/2012 database is by using SSMS to generate a database creation script for the database and executing the script in the desired SQL Server 2014 instance. You can then manually copy the data from the original relational database to the new relational database by using T-SQL scripts, DTS packages, SSIS packages, BCP commands, or any number of other methods that are available to SQL Server DBAs for copying data from one database to another.

Most DBAs 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 2014 and then fail over the log shipping as the final upgrade step. For more information about this alternative, see Chapter 4, "High Availability."

Copy Database Wizard

You can upgrade a SQL Server 2005/2008/2008 R2/2012 database by using the Copy Database Wizard. This wizard supports two upgrade approaches:

- **Detach/attach.** This approach 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, automatically reattaching the detached files after a copy (and optionally after a failed move).
- **SQL Server Management Objects (SMO).** This approach is similar in concept to the manual schema rebuild and data export/import upgrade method discussed earlier. It 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. It then 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 approach within the Copy Database Wizard method lets you automate and schedule the upgrade process at a convenient time. Each approach 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 method, 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 of 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 2014 Upgrade Advisor, which you can download from the [Microsoft SQL Server 2014 Feature Pack](http://www.microsoft.com/en-in/download/details.aspx?id=42295) (<http://www.microsoft.com/en-in/download/details.aspx?id=42295>). You should run Upgrade Advisor to analyze all the SQL Server 2005/2008/2008 R2/2012 databases that you want to upgrade. For information about how to install and run 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.

You should look at the most important SQL Server 2014 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://msdn.microsoft.com/en-us/library/cc280407(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/cc280407\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/cc280407(v=sql.120).aspx)). 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 2014 Upgrade Advisor Help file.

Deprecated Features

There are some features in SQL Server 2014 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 the SQL Server 2014 relational engine, see [Deprecated SQL Server Features in SQL Server 2014](http://msdn.microsoft.com/en-us/library/cc707789(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/cc707789\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/cc707789(v=sql.120).aspx)).

Discontinued Functionality

Several features from earlier versions of the SQL Server Database Engine are not supported in SQL Server 2014, so you must use replacement features for these. The following link lists the discontinued features that you will most likely encounter as well as the recommended replacement features: [Discontinued Database Engine Functionality in SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms144262(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms144262\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms144262(v=sql.120).aspx)).

For a complete list of discontinued features in SQL Server 2005/2008/2008 R2/2012, see the following topics in SQL Server Books Online:

- [Discontinued Database Engine Functionality in SQL Server 2005](http://msdn.microsoft.com/en-us/library/ms144262(v=sql.90).aspx)
(http://msdn.microsoft.com/en-us/library/ms144262(v=sql.90).aspx)
- [Discontinued Database Engine Functionality in SQL Server 2008](http://msdn.microsoft.com/en-us/library/ms144262(v=sql.100).aspx)
(http://msdn.microsoft.com/en-us/library/ms144262(v=sql.100).aspx)
- [Discontinued Database Engine Functionality in SQL Server 2008 R2](http://msdn.microsoft.com/en-us/library/ms144262(v=sql.105).aspx)
(http://msdn.microsoft.com/en-us/library/ms144262(v=sql.105).aspx)
- [Discontinued Database Engine Functionality in SQL Server 2012](http://msdn.microsoft.com/en-us/library/ms144262(v=sql.110).aspx)
(http://msdn.microsoft.com/en-us/library/ms144262(v=sql.110).aspx)

Breaking Changes

Some settings will prevent the SQL Server 2014 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 1 lists.

Table 1: Issues That Will Prevent an Upgrade

Issue	Corrective Action
Username of sys in a database. SQL Server 2014 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 2014 does not permit duplicate login SIDs for SQL Server authentication.	Drop and recreate the duplicate SQL Server logins on the legacy system.
Login names matching fixed server role names. SQL Server 2014 does not allow login names to match fixed server role names.	Rename the logins on the legacy system.
Database ID 32767. SQL Server 2014 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 2014 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 2014 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 2005/2008/2008 R2/2012 database to SQL Server 2014. You must resolve these issues before you start an upgrade, or the upgrade will fail. Table 2 lists the most likely issues of this kind together with the recommended corrective action.

Table 2: Issues to Resolve Before You Start the Upgrade Process

Issue	Corrective Action
Compressed drives. SQL Server 2014 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 filegroups. SQL Server will not upgrade filegroups in relational databases set to READ_ONLY because non-writeable files will not be upgraded.	Make sure that all filegroups 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 (for upgrades before SQL Server 2012), 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 filegroup 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 2014 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.

For a complete list of breaking changes in SQL Server 2005/2008/2008 R2/2012, see the following:

- [Breaking Changes to Database Engine Features in SQL Server 2005](http://msdn.microsoft.com/en-us/library/ms143179(v=sql.90).aspx)
(http://msdn.microsoft.com/en-us/library/ms143179(v=sql.90).aspx)
- [Breaking Changes to Database Engine Features in SQL Server 2008](http://msdn.microsoft.com/en-us/library/ms143179(v=sql.100).aspx)
(http://msdn.microsoft.com/en-us/library/ms143179(v=sql.100).aspx)
- [Breaking Changes to Database Engine Features in SQL Server 2008 R2](http://msdn.microsoft.com/en-us/library/ms143179(v=sql.105).aspx)
(http://msdn.microsoft.com/en-us/library/ms143179(v=sql.105).aspx)
- [Breaking Changes to Database Engine Features in SQL Server 2012](http://msdn.microsoft.com/en-us/library/ms143179(v=sql.110).aspx)
(http://msdn.microsoft.com/en-us/library/ms143179(v=sql.110).aspx)

Behavior Changes

SQL Server 2014 has some behavior changes that might require you to take corrective action after the upgrade is complete. For more information, see [Behavior Changes to Database Engine Features in SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms143359(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms143359\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143359(v=sql.120).aspx)).

Important: Backward compatibility with earlier versions was a high priority in SQL Server 2014, so in most cases, applications will behave as they did before the upgrade.

For a complete list of behavior changes in SQL Server 2005/2008/2008 R2/2012, please review the following topics:

- [Behavior Changes to Database Engine Features in SQL Server 2005](http://msdn.microsoft.com/en-us/library/ms143359(v=sql.90).aspx) ([http://msdn.microsoft.com/en-us/library/ms143359\(v=sql.90\).aspx](http://msdn.microsoft.com/en-us/library/ms143359(v=sql.90).aspx))
- [Behavior Changes to Database Engine Features in SQL Server 2008](http://msdn.microsoft.com/en-us/library/ms143359(v=sql.100).aspx) ([http://msdn.microsoft.com/en-us/library/ms143359\(v=sql.100\).aspx](http://msdn.microsoft.com/en-us/library/ms143359(v=sql.100).aspx))
- [Behavior Changes to Database Engine Features in SQL Server 2008 R2](http://msdn.microsoft.com/en-us/library/ms143359(v=sql.105).aspx) ([http://msdn.microsoft.com/en-us/library/ms143359\(v=sql.105\).aspx](http://msdn.microsoft.com/en-us/library/ms143359(v=sql.105).aspx))
- [Behavior Changes to Database Engine Features in SQL Server 2012](http://msdn.microsoft.com/en-us/library/ms143359(v=sql.110).aspx) ([http://msdn.microsoft.com/en-us/library/ms143359\(v=sql.110\).aspx](http://msdn.microsoft.com/en-us/library/ms143359(v=sql.110).aspx))

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 covers 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 2014 hardware and software requirements at [Hardware and Software Requirements for Installing SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms143506.aspx) (<http://msdn.microsoft.com/en-us/library/ms143506.aspx>). If you do not meet these requirements, the System Configuration Checker (SCC) part of the SQL Server Setup program will detect it and not permit Setup to continue.
2. Run SQL Server 2014 Upgrade Advisor to analyze legacy SQL Server 2005/2008/2008 R2/2012 relational engine components. Here are the steps involved:
 - a. Specify the name of the legacy server to analyze and select the SQL Server component, as Figure 1 shows. If you want to verify the compatibility of other components, you can also select them.

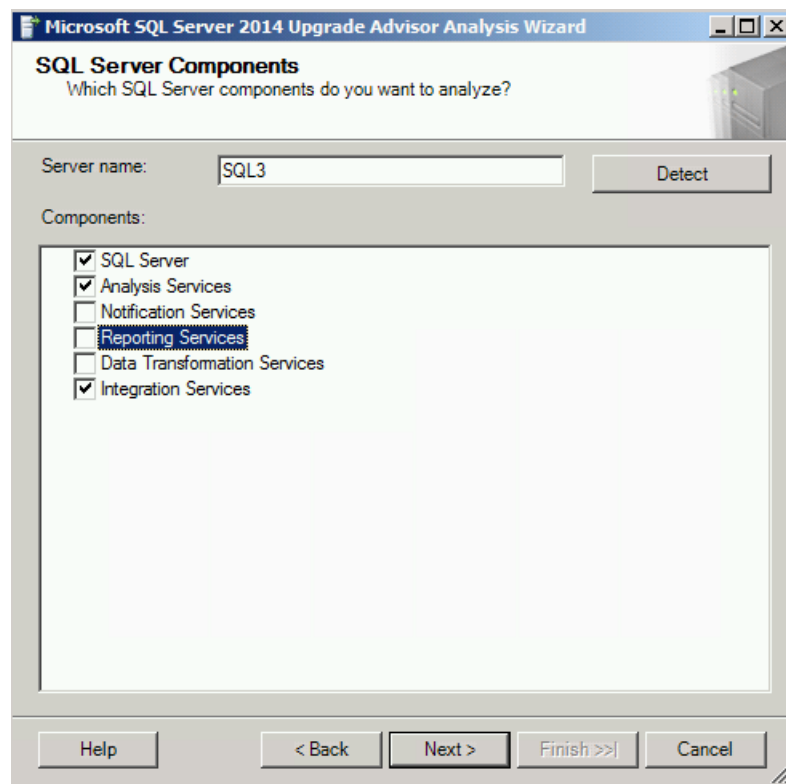


Figure 1: Selecting the components to analyze using SQL Server 2014 Upgrade Advisor

- b. Select the name of the SQL Server instance to analyze. In this example, the instance name is MSSQLSERVER, as Figure 2 shows. You also need to specify your authentication type and credentials for running the analysis.

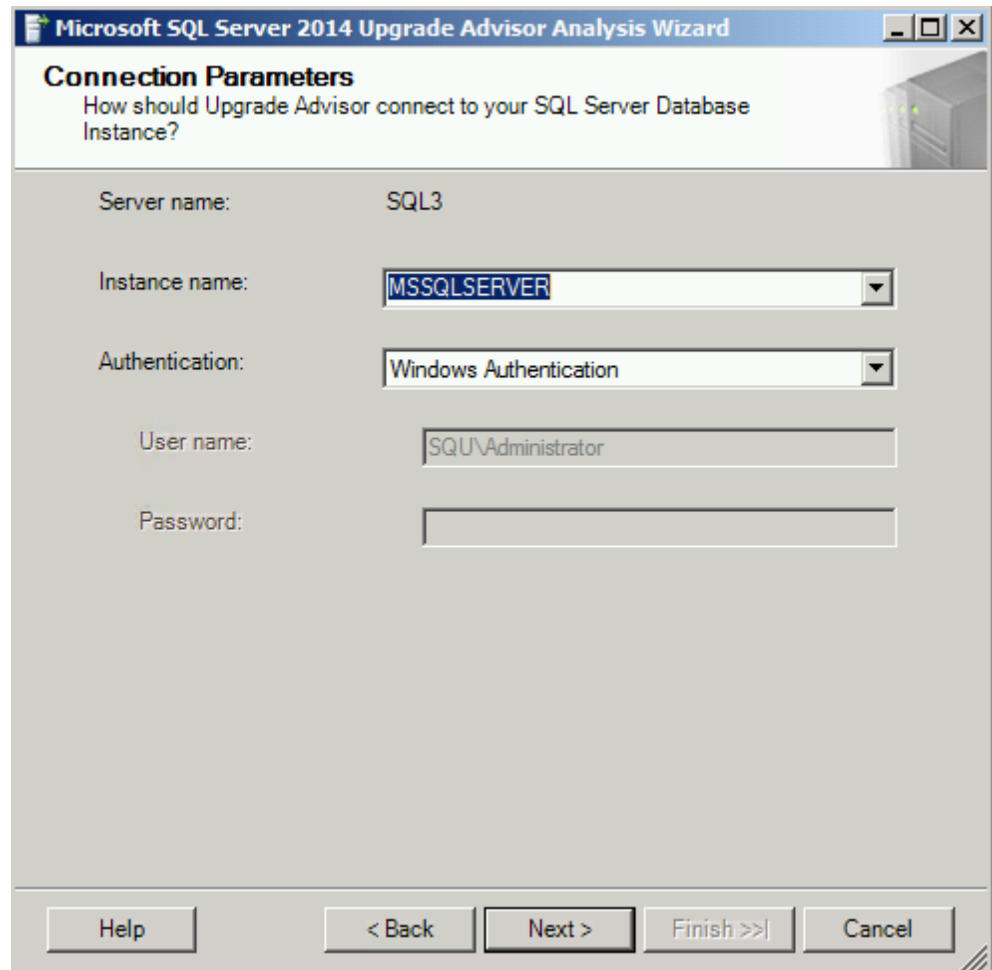


Figure 2: Selecting the instance of SQL Server to analyze

- c. Select one or more databases to analyze, as shown in Figure 3. You also have the option to analyze specified trace files and SQL batch files to help identify any T-SQL code that might not port well to SQL Server 2014.

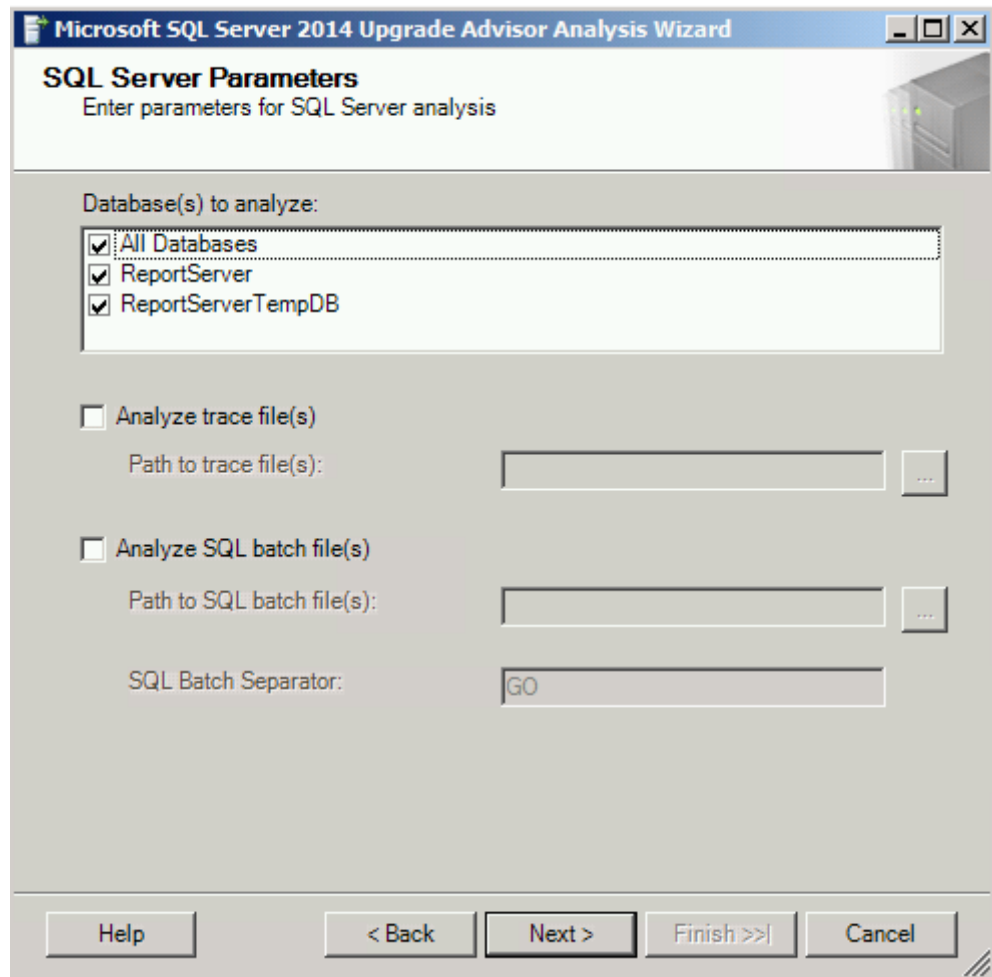


Figure 3: Selecting the SQL Server 2005/2008/2008 R2/2012 databases to analyze

- d. After Upgrade Advisor finishes its analysis, review the generated report. Figure 4 shows an example of the upgrade issue report, which lists 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.

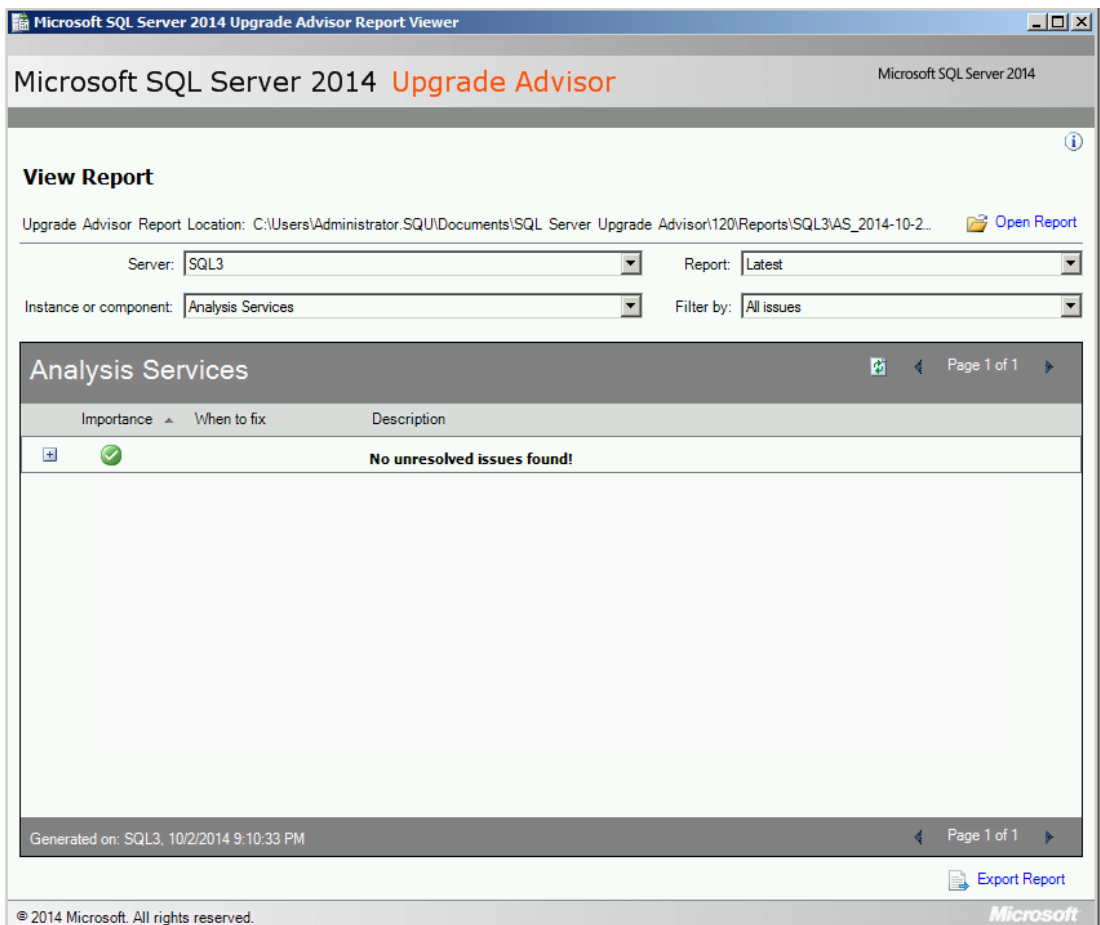


Figure 4: Reviewing Upgrade Advisor analysis results

3. Back up all system and user databases to make sure that you can roll back the upgrade if it is necessary.
4. Run DBCC CHECKDB on all databases to make sure that they are in a consistent state.
5. 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 2014 because of changes in the system database schema. You can turn off autogrow after the upgrade is complete.
6. 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 2014. You can turn off autogrow after the upgrade is complete.

7. 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.
8. Set the `AUTO_UPDATE_STATISTICS` option to `ON` for each database before upgrading to SQL Server 2014 (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.
9. Disable all startup procedures because they might block the upgrade process. You can re-enable the disabled startup procedures after the upgrade is complete.
10. Disable all trace flags before upgrading to SQL Server 2014. Some SQL Server 2005/2008/2008 R2/2012 trace flags may not exist in SQL Server 2014, and some trace flags may have different functionality in SQL Server 2014. If you use trace flags, you should verify each one after the upgrade to make sure there have been no changes before you enable any previously used trace flags.
11. If your systems are using SQL Server replication, stop replication and make sure that the replication log is empty.
12. 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.)
13. 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 earlier in Figures 1 through 4. Then review the generated report to verify that you have addressed all issues that must be resolved before the upgrade and to ensure 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 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 2014. 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 2014. 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 2005/2008/2008 R2/2012 instance.

Performing an Upgrade

After thoroughly preparing to move your SQL Server 2005/2008/2008 R2/2012 relational Database Engine to SQL Server 2014, 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 2014 Setup program.
2. Select the required SQL Server 2014 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 2005/2008/2008 R2/2012 to be upgraded. (You can only upgrade one instance at a time in-place.)
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 you use, the first step is installing the SQL Server 2014 relational database instance to which you plan to upgrade the SQL Server 2005/2008/2008 R2/2012 relational database. If you decide to install SQL Server 2014 on the same server together with SQL Server 2005/2008/2008 R2/2012, make sure that you install SQL Server 2014 with a named instance to avoid overwriting the existing legacy SQL Server instance on the server. (You might also verify installed instances during the SQL Server 2014 Setup process to check which instances are already present on the server.)

Note: In some cases, you might want to copy the system databases, including the master database, from the source SQL Server 2005/2008/2008 R2/2012 instance to the SQL Server 2014 instance before transferring user databases. For more information, see [Move System Databases](http://msdn.microsoft.com/en-us/library/ms345408.aspx) (<http://msdn.microsoft.com/en-us/library/ms345408.aspx>).

Next, you can upgrade the SQL Server 2005/2008/2008 R2/2012 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 2005/2008/2008 R2/2012 instance by using either SSMS or the BACKUP DATABASE T-SQL statement.

2. Use SSMS to connect to the SQL Server 2014 relational database instance to which you want to restore the legacy 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 or transfer the master and msdb database objects (e.g., logins, jobs, alerts) to the SQL Server 2014 instance from the legacy server.

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 2005/2008/2008 R2/2012 instance by using SSMS or the `sp_detach_db` stored procedure.
2. Copy 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 2014 instance by using SSMS or the `CREATE DATABASE T-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 2005/2008/2008 R2/2012.

Important: You must manually move or transfer the master and msdb database objects (e.g., logins, jobs, alerts) to the SQL Server 2014 instance from the legacy server.

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.
 - a. For the detach/attach approach, you must be a member of the `sysadmin` fixed server role on both the source and destination servers.
 - b. For the SMO approach, 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.
 - a. For the detach/attach approach, 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.
 - b. For the SMO approach, 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://msdn.microsoft.com/en-us/library/ms190698(v=sql.105).aspx) ([http://msdn.microsoft.com/en-us/library/ms190698\(v=sql.105\).aspx](http://msdn.microsoft.com/en-us/library/ms190698(v=sql.105).aspx)).

Important: You must manually move or transfer the master and msdb database objects (e.g., logins, jobs, alerts) to the SQL Server 2014 instance from the legacy server.

Post-Upgrade Tasks

You should take the following actions after you upgrade a relational database to SQL Server 2014 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 2014 instance.
2. Reregister your servers. For information about registering servers, see [Create a New Registered Server \(SQL Server Management Studio\)](http://msdn.microsoft.com/en-us/library/ms188231(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms188231\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms188231(v=sql.120).aspx)).

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 or 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 and 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 2014 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 2005/2008/2008 R2/2012 consist of T-SQL commands executed by SQL Server Agent. Starting with SQL Server 2005, database maintenance plans are SSIS packages and will be automatically upgraded. For more information, see [Maintenance Plans](http://technet.microsoft.com/en-us/library/ms187658(v=sql.120).aspx) ([http://technet.microsoft.com/en-us/library/ms187658\(v=sql.120\).aspx](http://technet.microsoft.com/en-us/library/ms187658(v=sql.120).aspx)).

Changes in the Cardinality Estimator

There is a redesigned SQL Server Query Optimizer cardinality estimation process that may have significant query performance impacts. Review the changes from the whitepaper "[Optimizing Your Query Plans with the SQL Server 2014 Cardinality Estimator](http://download.microsoft.com/download/D/2/0/D20E1C5F-72EA-4505-9F26-FEF9550EFD44/Optimizing%20Your%20Query%20Plans%20with%20the%20SQL%20Server%202014%20Cardinality%20Estimator.docx)" (<http://download.microsoft.com/download/D/2/0/D20E1C5F-72EA-4505-9F26-FEF9550EFD44/Optimizing%20Your%20Query%20Plans%20with%20the%20SQL%20Server%202014%20Cardinality%20Estimator.docx>).

Review Use Plan Hints

Another post-upgrade task that you need 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 R2 changed 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 2014. We recommend the following procedures after you upgrade to SQL Server 2014.

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://msdn.microsoft.com/en-us/library/ms345599.aspx) (<http://msdn.microsoft.com/en-us/library/ms345599.aspx>).

Important Information about Query Plans

Be aware that how SQL Server determines query plans varies between versions—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, review the SQL Server 2014 Books Online documentation [Migrate Query Plans](http://msdn.microsoft.com/en-us/library/bb895281(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/bb895281\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/bb895281(v=sql.120).aspx)).

Connecting Client Applications to SQL Server 2014

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 highlights some potential issues that might affect client connectivity.

Table 3: Issues That Might Affect Client Connectivity

Issue	Description
Network protocols	The only supported network protocols are now TCP/IP Sockets, named pipes, VIA, and 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.

Issue	Description
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 2014 installations and must be enabled through the SQL Server Configuration Manager tool.

Conclusion

The SQL Server 2014 relational Database Engine offers many improvements over SQL Server 2005, 2008, 2008 R2, and SQL Server 2012, as well as significant new features. The first step in taking advantage of these improvements is upgrading your existing databases to SQL Server 2014. As this chapter explains, you have two main choices for upgrading legacy SQL Server 2005/2008/2008 R2/2012 databases to SQL Server 2014: in-place or side-by-side. If you select the side-by-side method, you have additional choices to make, including whether to use the backup/restore, detach/attach, or Copy Database Wizard method—or an alternative. Each of these options has its pros and cons, so you need 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 2014, see the following links:

- [Upgrade to SQL Server 2014](http://msdn.microsoft.com/en-us/library/bb677622(v=sql.120).aspx)
(http://msdn.microsoft.com/en-us/library/bb677622(v=sql.120).aspx)
- [SQL Server 2014 Web Site](http://www.microsoft.com/sqlserver/en/us/default.aspx)
(http://www.microsoft.com/sqlserver/en/us/default.aspx)
- [Books Online for SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms130214(v=SQL.120).aspx)
(http://msdn.microsoft.com/en-us/library/ms130214(v=SQL.120).aspx)
- [SQL Server Developer Center](http://msdn.microsoft.com/en-us/sqlserver/)
(http://msdn.microsoft.com/en-us/sqlserver/)
- [SQL Server TechNet Resources](http://technet.microsoft.com/en-us/sqlserver)
(http://technet.microsoft.com/en-us/sqlserver)

Chapter 4: High Availability

Introduction

An important goal of any upgrade should be to minimize the downtime and subsequently the impact to applications and end users. This chapter discusses how to minimize downtime when upgrading to SQL Server 2014 as well as how to upgrade when the high availability features of failover clustering, log shipping, database mirroring, and replication are part of the current configuration. This chapter does not cover how each high availability feature works unless it is different in SQL Server 2014 and has a bearing on the upgrade process. Prior basic knowledge of the high availability features is assumed.

Preparing to Upgrade

Before looking at how to have highly available upgrades to SQL Server 2014, this section details the types of upgrades possible and how each should be thought about from the perspective of 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 the upgrade types and their advantages and disadvantages for different scenarios, see Chapter 1, "Upgrade Planning and Deployment."

During an 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 copy operations, 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 SQL Server 2014 Setup to directly upgrade an instance of SQL Server 2005, SQL Server 2008, SQL Server 2008 R2, SQL Server 2012, or pre-release SQL Server 2014 to SQL Server 2014 release to manufacturing (RTM) on the same server or Windows Server Failover Clustering (WSFC) cluster if you have already deployed the supported version of SQL Server for upgrading on a supported operating system version (Windows Server 2008 Service Pack 2 (SP2) or later or Windows Server 2008 R2 SP1 or later). The "Upgrade Strategies" section in Chapter 1 describes the in-

place upgrade process in detail.

An in-place upgrade has the highest risk of extended downtime compared with the other upgrade methods because the upgrade occurs where the source instance is configured. There is no way to isolate or minimize downtime; the instance and user databases will be unavailable during the entire time of the upgrade. For some organizations, the downtime and the ability to reuse hardware may offset the cost and resources necessary to use another upgrade method.

The more important consideration with an in-place upgrade is that the installation of SQL Server may not be usable should the process fail for whatever reason. This worst-case scenario would need to be remedied by 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. Backup and restore is the only fallback plan, so if pursuing an in-place upgrade, make sure that you have good backups and tested plans for bare metal recovery.

Side-by-Side Upgrade on the Same Standalone Server or Cluster

This type of side-by-side upgrade is when data is moved from an existing instance of SQL Server 2005, SQL Server 2008, SQL Server 2008 R2, SQL Server 2012, or pre-release SQL Server 2014 to a new and separate instance of SQL Server 2014 that resides on the same standalone server or WSFC cluster. The "Upgrade Strategies" section in Chapter 1 describes this process in detail.

A side-by-side upgrade on the same hardware or WSFC cluster offers slightly better protection than an in-place upgrade because the old configuration is technically still available if something fails. The real cost is the downtime to switch the databases to the instance of SQL Server 2014, as the moving of data and objects to the new instance could be handled transparently. It also means that you will incur storage costs because you will need to account for two copies of the data. The downtime to applications and end users could translate into minutes of downtime depending on the process used, which is significantly lower than an in-place upgrade.

With both Windows Server 2008 and Windows Server 2008 R2, depending on what is already deployed, you could potentially have SQL Server 2005, SQL Server 2008, SQL Server 2008 R2, SQL Server 2012, *and* SQL Server 2014 instances co-existing on the same server or WSFC cluster.

With Windows Server 2012 and Windows Server 2012 R2, you could potentially have SQL Server 2008, SQL Server 2008 R2, SQL Server 2012, *and* SQL Server 2014 instances co-existing on the same server or WSFC cluster.

Note: There are resource DLLs for both SQL Server and SQL Server Agent that are shared by all versions of clustered instances of SQL Server on the same WSFC cluster. This is expected and fully supported.

Problems with a side-by-side upgrade on the same server or WSFC cluster include the following:

- It is impossible to go back to the original configuration as it existed before the upgrade process began. The reason is that installing SQL Server 2014 will replace some shared components of the older SQL Server version(s) already installed (such as connectivity) with their newer counterparts. This should not pose any major problems, but you will no longer have a “pure” SQL Server 2005, SQL Server 2008, SQL Server 2008 R2, or SQL Server 2012 installation.
- You may still encounter downtime because some SQL Server 2014 components could require a reboot to install (for example, the resource DLL for a clustered configuration).
- SQL Server 2014 is still bound by the same rules as SQL Server 2005, SQL Server 2008, SQL Server 2008 R2, and SQL Server 2012 when it comes to the individual instance names. On a single server or WSFC, there can be only one default instance. Everything else can be a named instance. If one of the existing instances (2005, 2008, 2008 R2, or 2012) is already using the default instance, the SQL Server 2014 installation must be a named instance. Ensure that this will not pose a problem to the application after the upgrade is complete, otherwise the application may not be able to connect to the new instance.
- In a side-by-side configuration, redirecting users and applications to the new SQL Server 2014 instance or database may involve some manual steps and other processes, such as synchronizing logins or modifying the application’s connection string. Ensure these processes are identified, tested, and fully documented. Failure to do so may cause unnecessary outages or extended downtime.
- It will be very hard to go back to the previous configuration that existed pre-upgrade. There are two main aspects to this:

- The existing server or cluster configuration is altered by the new SQL Server 2014 instance.
- The databases now exist in a new instance of SQL Server 2014.

For the first problem, the only way to get back to where you were would be to restore system backups or reconfigure the hardware from scratch. That will most likely not be the issue; the issue would most likely occur if an application's database that now exists in SQL Server 2014 starts to exhibit some sort of issue that would require it to be moved back to the previous version of SQL Server. Because you cannot take a backup of a SQL Server 2014 database and restore it to an earlier version of SQL Server, you will need to devise some other way to extract the data, such as using BCP or SQL Server Integration Services (SSIS). Data movement is the more trivial aspect of porting the database back. The difficult aspect is trying to figure out what data changed. That would be a manual process.

Side-by-Side Upgrade to a Separate Standalone Server or Cluster

Another variant of a side-by-side upgrade is when the user databases and its objects are migrated and upgraded to a new instance of SQL Server 2014 on brand new hardware (standalone or clustered). 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 (although you would need a method to synchronize the changed data, which is not a trivial task). In addition, you can minimize downtime by using whatever method is best for moving the databases and objects to the new instance. A major advantage to this option is that you can get the hardware configured and tested long before the cutover date, allowing you to use the new environment to test the process for moving the data.

Using a separate server has most of the same drawbacks as a side-by-side upgrade on the same server or WSFC cluster. 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 Active Directory domain, which would affect a clustered deployment.

For example, assume that the old standalone server name is MYSERVER with a default instance of SQL Server 2012. If a new server is deployed, the new server (and default instance) could not use MYSERVER since MYSERVER is already used in Active Directory.

With clusters, even the instance name must be unique in the domain since it is not the same as the name for the underlying WSFC cluster or any of the nodes. If the old clustered instance was a default instance named MYSQLINS, you could not reuse that name when configuring the new cluster. You can rename a clustered instance only after the old server has been decommissioned; however, you can only rename the portion of a clustered instance that exists in Active Directory. So a named instance cannot be fully renamed because the part *after* the slash cannot be renamed. For example, assume you have a named instance of MYINS\INSNAME. MYINS can be renamed but INSNAME cannot.

The reason this all matters is that some applications may be inflexible and require a specific name to connect to. This is more glaring if the application is installed on many desktops. The amount of work required to touch every desktop may be impossible, so if the same name must be used but the old server cannot be decommissioned or the same name cannot be used, other methods such as using aliases in DNS must be considered. That type of requirement 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 has no problems, has the right configuration, and is stable before tearing down the old environment. Decommissioning and disabling the original instance too soon could be a mistake. How long you are down will depend on your fallback plan. It is usually more effective to switch back to the original, known, and already configured environment than to have to restore and rebuild from scratch.

Before allowing users to access the new SQL Server 2014 instance, you should disable the old one so that users and applications will not accidentally connect to the wrong server. This may not be possible if all the databases have not been migrated, but you run the potential risk of users accessing the old environment and updating what is now the old data. The expense and time involved to detangle that mess 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 containing the original instance to Server B containing the SQL Server 2014 instance. (The network and the disk I/O of both servers will come into play.) The restore speed will depend on the hardware configuration (including the underlying disk subsystem) 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 terabytes range, waiting until the cutover time to start the backup, copy, and restore process might exceed the allowed outage window. Just copying a terabyte database could take the better part of 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 backup compression (either third party or what is built into SQL Server 2008 or later) to generate backups. This shrinks the backup size (which means a quicker copy time) and usually speeds up the backup and restore process.
- 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. For more information, see the "Hardware-Assisted Database Moves" section later in this chapter.

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 process 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 a risk. In addition, if the database is large, this approach is impractical for the reasons stated previously.

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 operation—can be done well in advance of the actual cutover.

The problem is that while SQL Server 2014 can restore backups from older versions of SQL Server going back to SQL Server 2005, the log shipping features of those earlier versions of SQL Server cannot be used to configure log shipping to SQL Server 2014. Custom log shipping scripts (such as the scripts listed in the "Additional References" section at the end of the chapter) would need to be used.

Hardware-Assisted Database Moves

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 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 many 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. (This can be seen in the SQL Server log.) Although this process takes a few seconds, it should be completely transparent.

It is 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) and/or use a Volume Shadow Copy Service (VSS) provider that supports it. 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 also 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 Backup Device Interface \(VDI\) Specification](http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=17282) white paper (<http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=17282>).

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 during 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 During the Upgrade

This section covers the key strategies for promoting highly available upgrades for all installations of SQL Server, minimizing the downtime in the switch to SQL Server 2014.

Prepare for SQL Server 2014

Before putting together an upgrade plan or executing an upgrade to SQL Server 2014, you must take into account the following considerations, which will influence how long of an outage you may incur during the upgrade process.

Learn What Has Changed

When it comes to minimizing downtime, one of the best things to do is anticipate changes and account for them accordingly based on the knowledge of your

applications and overall SQL Server environment. Although some changes to the database engine and the other components you use 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 could be perceived as downtime or cause an outage later (such as a code fix to the application after going live with SQL Server 2014). If an upgrade is properly planned, tested, and executed, it should not introduce large performance degradations or problems. A database administrator (DBA) should never be caught by surprise. For a complete list of backward-compatibility issues in SQL Server 2014, including deprecated features, discontinued features, breaking changes, and behavior changes, see [Backward Compatibility](http://msdn.microsoft.com/en-us/library/cc280407.aspx) (<http://msdn.microsoft.com/en-us/library/cc280407.aspx>) in SQL Server 2014 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 DBAs are fully trained and prepared for the upgrade will increase uptime. For information about how to make sure that those administering or deploying SQL Server 2014 are ready, see "Treat the Upgrade as an IT Project" in Chapter 1.

Check Minimum Hardware Requirements

Make sure that the hardware you plan to use for SQL Server 2014 meets the minimum requirements documented in the "Minimum Hardware and Software Requirements for SQL Server 2014" section in Chapter 1, as well as in [Hardware and Software Requirements for Installing SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms143506.aspx) (<http://msdn.microsoft.com/en-us/library/ms143506.aspx>). When it comes to availability, it is a bad idea 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 exceeding 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.

Understand Licensing Changes

Licensing changed significantly with SQL Server 2012 and SQL Server 2014. Where prior versions of SQL Server allowed you to license per-processor, SQL Server 2014 has

changed per processor to per core. Since cost is a factor when upgrading, contact the person in your company responsible for acquiring licensing so that they can do the right research and get the proper pricing and license for the SQL Server 2014 deployment. For more information, see the [SQL Server 2014 licensing page](http://www.microsoft.com/sqlserver/en/us/get-sql-server/licensing.aspx) (<http://www.microsoft.com/sqlserver/en/us/get-sql-server/licensing.aspx>).

Disk Space

Disk space, or lack thereof, is one of the major causes of downtime. During the installation of SQL Server 2014, SQL Server will tell you how much disk space is needed. For more information, see the "Hard Disk Space Requirements (32-Bit and 64-Bit)" section in Chapter 1. [Hardware and Software Requirements for Installing SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms143506.aspx) (<http://msdn.microsoft.com/en-us/library/ms143506.aspx>) 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, one of the biggest consumers of disk space will be the backups used to initialize the upgrade in a side-by-side scenario as well as the backups made prior to the upgrade or decommissioning. Make sure you account for this space as noted in the "Perform Database Backups" section 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 2014. 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 DBAs but also all the teams involved (application, storage, operating system, network, and so on) to minimize downtime. Attempting to do an upgrade without a plan that is tested—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 and steps 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.

Time management is crucial when coming up with your plans. 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 7 A.M. on Monday, your no-go point would be around 2 A.M. on 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

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 or hard it is to test the upgrade plan is directly related to the selected upgrade method and the complexity of your applications and environment, with most of the issues falling squarely on the application.

If you find that the plan is flawed halfway through the upgrade, it may be too late to fix the problem (or easily adjust it) if the problem encountered was not already known. Finding any problems and either accounting for or correcting them before initiating the upgrade is essential for ensuring a successful upgrade. A major 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, no matter whether it is the old version or SQL Server 2014. Without testing, it is anyone's best guess, and chances are that 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 a DBA and/or 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 due to configuration differences, such as not having a cluster in a non-production environment. Performing tests against a standalone server will validate that the databases and application work after an upgrade to SQL Server 2014, but the upgrade process for a standalone 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 2014

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 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 2014.

Install Prerequisites

SQL Server 2014 requires [Microsoft .NET Framework 3.5 SP1](http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=22) (<http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=22>), [Windows PowerShell 2.0](http://support.microsoft.com/kb/968929/en-us) (<http://support.microsoft.com/kb/968929/en-us>), and .NET Framework 4.0. Depending on your operating system version, you may have to install all of these or only some.

For Windows Server 2008, all of them will need to be installed. For Windows Server 2008 R2, .NET Framework 3.51 is an optional component enabled in Windows (not by an external installer), so you would only need to worry about .NET Framework 4.0. For a Windows Server 2012 or Windows Server 2012 R2 implementation, you would need to enable .NET Framework 3.51 in the operating system.

Important: If you plan on using the AlwaysOn capabilities of SQL Server 2014, it is recommended that you install at a minimum the 4.0.2 update for .NET Framework 4.0. You should also install the hotfix noted in the Knowledge Base article "[An update introduces support for the AlwaysOn features in SQL Server 2012 or a later version to the .NET Framework 3.5 SP1](http://support.microsoft.com/kb/2654347)" (<http://support.microsoft.com/kb/2654347>), which gives .NET Framework 3.51 all of the AlwaysOn capabilities.

SQL Server 2014's Setup will detect what required components are missing and try to install or enable them (depending on your operating system). Note, however, that this is not the case for Windows PowerShell 2.0 or .NET Framework 3.5 SP1, which are prerequisites but will not be installed by SQL Server 2014 Setup. The downside of that is the install or upgrade times will increase. A way to minimize the time is to ensure that the prerequisites are installed before you do the upgrade. If you have an outage

window prior to the upgrade, consider installing these components. For example, putting .NET Framework 4.0 on a server should introduce minimal (if any) risk. Some of these prerequisites may cause a reboot, which is why the recommendation is to do these when you would impact the systems the least. If you can't afford an outage, just account for the time in your SQL Server 2014 install or upgrade.

Ensure Database Health

SQL Server provides the Transact-SQL (T-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 Tools." The problem with running a full DBCC CHECKDB is that it generally affects a database's availability. 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 difficult to complete within the set window.

It is also recommended that you run DBCC CHECKDB after the database is upgraded to SQL Server 2014 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 the additional time.

Run SQL Server 2014 Upgrade Advisor

SQL Server 2014 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 2014 Upgrade Advisor" section in Chapter 1.

Run the Appropriate Best Practices Analyzer

The SQL Server Best Practices Analyzer (BPA) 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 2005, 2008, 2008 R2, and 2012" section in Chapter 1.

Perform Database Backups

To protect data in an upgrade, it is important 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 operation. 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, which provides instance-level protection, accounts for objects that reside outside the database. Chapter 2, "Management Tools," and Chapter 17, "Integration Services," discuss how to use scripting or 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, and user-created stored procedures.

To minimize downtime, prepare any scripts or SSIS packages before the upgrade. 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 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 required by SQL Server 2014 ahead of the actual SQL Server 2014 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 has been integrated into the Database Engine since SQL Server 2014. If upgrading from a clustered SQL Server 2005 instance, there will no longer be a dedicated cluster resource for full-text in the resource group with SQL Server. This is expected behavior. In addition, Full-Text will always be included with SQL Server 2014, even if it was not installed in SQL Server 2005. For all deployments both clustered and not clustered, if you use full-text indexes, you will have more to upgrade. Plan accordingly for the additional downtime required.

Replication

If you are upgrading from a clustered SQL Server 2005 instance, you may not have installed the replication feature in your SQL Server 2005 failover cluster. SQL Server 2014 automatically will install replication as a feature even if you are not using it. For details about upgrading replicated databases, see the "Upgrading Replicated Databases" section 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 any existing service accounts will also be used for SQL Server 2014. Anyone who has access to those accounts and knows the passwords will have full access to the new SQL Server 2014 instance. We do not recommend using the existing accounts for the upgraded instances of SQL Server 2014 if that is a concern. To remedy this situation, create new domain-based service accounts with the correct privileges on each server. After the upgrade, use SQL Server 2014 Configuration Manager (covered in Chapter 5, "Database Security") to update the newly upgraded services to use these new accounts.

Management Tools and Utilities

After the upgrade, only SQL Server 2014 tools and utilities should be used to manage SQL Server 2014 instances. This will require that the appropriate version for SQL Server 2014 is installed somewhere in your ecosystem. You may still need to manage older versions of SQL Server, so you may need to have multiple versions of SQL Server Management Studio (SSMS) to have the most optimal experience for a particular version of SQL Server.

Note: When you upgrade in-place to SQL Server 2014, the SQL Server 2014 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 necessary, uninstall them.

Upgrading Failover Cluster Instances

This section describes the considerations and processes for upgrading existing SQL Server 2005, SQL Server 2008, SQL Server 2008 R2, and SQL Server 2012 failover clustering instances (FCIs) to SQL Server 2014. This is not intended as a formal SQL Server 2014 failover clustering white paper.

For additional information about upgrading failover clusters, see [Upgrade a SQL Server Failover Cluster](http://msdn.microsoft.com/en-us/library/ms191009.aspx) (<http://msdn.microsoft.com/en-us/library/ms191009.aspx>).

Feature Changes in SQL Server 2014 Failover Clustering

SQL Server 2014 failover clustering provides availability through failover based on the underlying WSFC. However, SQL Server 2014's installation or upgrade process is very different compared with that in SQL Server 2005. (SQL Server 2014's process is the same as in SQL Server 2008, SQL Server 2008 R2, and SQL Server 2012.) In SQL Server 2005, all nodes are configured during one pass. In SQL Server 2014, each instance on every node must be installed or upgraded separately. For the installation of a new FCI, there is one main install for the SQL Server instance (also known as a "first node installation"), and everything else is an add node operation. For an in-place upgrade, the process is also node-by-node, as described later in this section.

Table 1 shows an example of how many install processes would be executed for up to four nodes and four instances of SQL Server.

Table 1: Number of Installs or Upgrades per Node Depending on the Number of Instances Required

	One Instance	Two Instances	Three Instances	Four Instances
Two nodes	2	4	6	8
Three nodes	3	6	9	12
Four 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 feature change is that if you are implementing a new FCI, SQL Server 2014, like SQL Server 2012, does not require the domain groups that were introduced with SQL Server 2005. SQL Server 2014 uses a Service SID in addition to the service accounts. For information about Service SIDs, see Cyril Voisin's [Per-service SID](#) security blog entry (<http://blogs.technet.com/b/voy/archive/2007/03/22/per-service-sid.aspx>).

Operating System Versions and SQL Server 2014 Failover Clustering

You may want to upgrade your hardware and your operating system as a part of the overall upgrade to SQL Server 2014. If you are coming from SQL Server 2005, you are probably using Windows Server 2003. Windows Server 2003 is no longer in mainstream support, nor is it a supported operating system for SQL Server 2014. You will have to use Windows Server 2008 SP2, Windows Server 2008 R2 SP1 (or later), Windows Server 2012, or Windows Server 2012 R2.

Windows Server 2008 SP2 mainstream support will end during SQL Server 2014's lifecycle. It is strongly recommended that you use Windows Server 2012 R2 for your deployments. Check the [Microsoft Support Lifecycle](#) web site (<http://support.microsoft.com/lifecycle/#tab0>) for the support dates for your country.

If you are going to upgrade both the hardware and operating system at the same time, you will need to do a side-by-side upgrade to the new hardware and use your preferred method to migrate the databases. If your current hardware is still viable and logged for your supported Windows Server version, you cannot do an in-place upgrade when it comes to a WSFC cluster. You will have to completely reinstall the operating system, which means a new installation of SQL Server 2014 on top of that. For that reason alone, it is much better to upgrade using new hardware so you have a fallback plan.

A side-by-side upgrade on the same or a different WSFC cluster requires additional dedicated disk space for the new clustered instances, as well as both a new IP address and name for the new SQL Server 2014 FCI.

Note: While WSFC has the Migrate a Cluster Wizard, it cannot be used for SQL Server. Make sure you make proper backups of everything prior to removing the old cluster.

SQL Server 2014 supports Windows Server Core if you are using Windows Server 2008 R2 SP1 or later. Windows Server Core is a “stripped down” version of Windows that has no user interface, relies on the command line, and restricts what types of applications are allowed to run. It can potentially be more secure and offer other benefits, such as reducing the amount of patches that need to be applied. It is not possible to convert existing installations to Windows Server Core during an upgrade unless you use completely new hardware.

Notable Changes to Both Windows and SQL Server 2014 Failover Clustering

If you are only used to clustering using Windows Server 2003, Windows Server 2008 (and later) has quite a few new features that you should become familiar with. SQL Server 2014 also introduces a few improvements that you should be aware of.

The Windows Server 2008 (and later) failover clustering features include the following:

- Starting with Windows Server 2008, a WSFC is 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 as long as the hardware itself is logoed for the version of the operating system you plan on deploying. 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.

Do not use this capability to cobble two or more odd pieces of hardware 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 to provide correct information about which drivers are certified for clusters using Windows Server 2008.

The support policy for Windows Server 2008 and Windows Server 2008 R2 WSFC clusters is clearly outlined in the Knowledge Base article [The Microsoft Support Policy for Windows Server 2008 or Windows Server 2008 R2 Failover Clusters](http://support.microsoft.com/kb/943984) (http://support.microsoft.com/kb/943984). Windows Server 2012 and Windows Server 2012 R2 have a matching article, [The Microsoft Support Policy for Windows Server 2012 or Windows Server 2012 R2 Failover Clusters](http://support.microsoft.com/kb/2775067) (http://support.microsoft.com/kb/2775067).

Important: It is crucial to ensure that the underlying WSFC has no errors during cluster validation. SQL Server 2014's Setup relies on the validation results. If an error is found, Setup will not proceed.

- All supported versions of Windows Server failover clustering support multiple subnets for cluster nodes natively.
- As of Windows Server 2008 R2, the command line cluster.exe is deprecated. Windows Server 2008 R2 introduced PowerShell cmdlets to replace the cluster.exe functionality.
- Quorum introduces the concept of a witness. There are four quorum models in Windows Server 2008 and Windows Server 2008 R2. They are:
 - No Majority. This is the same as the old disk-based quorum, where the witness 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.
- Quorum modes in Windows Server 2012 and Windows Server 2012 R2 have similar quorum models. In any case, it is strongly recommended that without a specific reason, users accept the quorum model selected by the Failover Clustering Wizard.
 - Node Majority (no witness). This is not recommended unless you have an odd number of nodes in the cluster. Only nodes vote; the majority wins.

- Node Majority with Witness (disk or file share). This is the same as Node and Disk Majority and Node and File Share Majority as noted previously. It is recommended for clusters with an even number of nodes.
- No Majority (Disk Witness Only). This is the same as the old disk-based quorum, where the witness disk is a single point of failure. It is not recommended.
- Windows Server 2012 R2 introduced Dynamic Quorum Management. This mechanism allows your cluster to react to changing conditions to improve availability. For more information, see the "Dynamic quorum management" section in [Configure and Manage the Quorum in a Windows Server 2012 Failover Cluster](http://technet.microsoft.com/en-us/library/jj612870.aspx#BKMK_dynamic) (http://technet.microsoft.com/en-us/library/jj612870.aspx#BKMK_dynamic).
- Clustering the Microsoft Distributed Transaction Coordinator (DTC) is not required with SQL Server 2014. If you do choose to use a clustered DTC, you should create a dedicated MSDTC resource for each SQL Server instance/group.
- All cluster nodes must be in the same domain to implement SQL Server 2014.
- The private network, also known as the heartbeat, is now unicast.
- All disks must still be Basic disks (not Dynamic). However, GUID partition table (GPT) disks are supported for sizes over 2 TB.
- If using Cluster Shared Volumes (CSV) with Windows Server 2012 R2, you must format the disks with NTFS.

SQL Server 2014 Failover Clustering Enhancements

The list below summarizes the major changes to the failover clustering feature in SQL Server 2014:

- Failover Clustering can now use CSVs for shared disks. For more information, review the [AlwaysOn Failover Cluster Instances \(SQL Server\)](http://msdn.microsoft.com/en-us/library/ms189134.aspx) topic (<http://msdn.microsoft.com/en-us/library/ms189134.aspx>) in SQL Server 2014 Books Online as well as the [Understanding Cluster Shared Volumes in a Failover Cluster](http://technet.microsoft.com/en-us/library/dd759255.aspx) topic (<http://technet.microsoft.com/en-us/library/dd759255.aspx>) in the Windows Server documentation.
- SQL Server 2014 AlwaysOn dynamic management views (DMVs) have been enhanced to return failover cluster information, including the

sys.dm_hadr_cluster, sys.dm_hadr_cluster_members, and sys.dm_hadr_cluster_networks DMVs.

Considerations for Upgrading a SQL Server 2005 Failover Cluster to SQL Server 2014

Besides the potential operating system issue noted earlier, you may also be affected by the Windows on Windows 64 (WOW64) concerns. As with SQL Server 2008, SQL Server 2008 R2, and SQL Server 2012, SQL Server 2014 does not support installing a 32-bit failover clustering instance under the WOW64 feature of 64-bit Windows. SQL Server 2005 technically allowed it, but it was not recommended. If your deployment is set up that way, an in-place upgrade of a WOW64-based SQL Server 2005 failover cluster to SQL Server 2014 is not supported. To upgrade to SQL Server 2014 if a WOW64-based SQL Server 2005 failover cluster is deployed, you must use a side-by-side approach if you want to do it on the same WSFC cluster.

Another thing to note is that as of SQL Server 2008, full-text has been integrated into the Database Engine. With SQL Server 2005, full-text got its own resource in the cluster. After the upgrade, there will no longer be a full-text resource shown separately in the cluster configuration.

Considerations for Upgrading SQL Server 2008, SQL Server 2008 R2, or SQL Server 2012 to SQL Server 2014

There are no specific cluster-related considerations when upgrading from SQL Server 2008, SQL Server 2008 R2, or SQL Server 2012 to SQL Server 2014. The generic conditions that apply to all versions apply in this scenario.

Considerations for Upgrading a Pre-Release Version of SQL Server 2014 to SQL Server 2014 RTM

SQL Server 2014 is the second major version of SQL Server to support upgrading from pre-release versions of the product. However, you can only upgrade from SQL Server 2014 Community Technology Preview 2 (CTP2) to SQL Server 2014 RTM. SQL Server 2014 CTP1 is not a supported upgrade path.

Upgrading a Failover Clustering Instance to SQL Server 2012

The process for upgrading a SQL Server 2005, SQL Server 2008, SQL Server 2008 R2, SQL Server 2012, or pre-release SQL Server 2014 failover clustering instance to SQL

Server 2014 RTM is the same. You can perform the upgrade through the command line or the Setup interface. This section will discuss those options and highlight the important factors you need to consider.

Tip: To install a new clustered instance and then use another method to upgrade databases, follow the instructions in [SQL Server Failover Cluster Installation](http://technet.microsoft.com/en-us/library/hh231721.aspx) (<http://technet.microsoft.com/en-us/library/hh231721.aspx>).

Instance Failover Behavior During the Upgrade Process

It is important to understand how the upgrade process handles failing over a clustered instance during the upgrade process. By default, SQL Server's Setup program will handle everything for you. This may be fine when you have only two nodes and a single instance, but if you have multiple nodes and instances, you may want to control how and when instances move from one node to another.

Like installing a new clustered instance, upgrading is an instance-by-instance, node-by-node affair. That means a single upgrade will only affect a specific instance on one node. The only exception is when you have multiple instances, in which any shared components—including the resource DLL—will be upgraded. The binaries specific to the instance(s) not being upgraded will not be touched. However, because the resource DLL will be updated, this will either force you to move that instance to another node or take it offline once the resource DLL is replaced. Since you want to minimize the impact to end users, you should coordinate an outage with the application owner as well as the business. The good news is that once the resource DLL is replaced, no matter how many instances you have that could potentially run on that node, you will most likely not need another reboot unless you have some other files locked.

If you are upgrading an instance of SQL Server and have another FCI already running on that node, once the resource DLL replacement happens, you could cause the other instance to go offline, as shown in Figure 1. This is why you want to control failovers. In this example, the instance SQL2K14CTP2 was upgraded on a node where there was a running SQL Server 2005 instance (SQL2k5Cluster\SQL2K5). This is also detected by one of the upgrade rules, as shown in Figure 2.

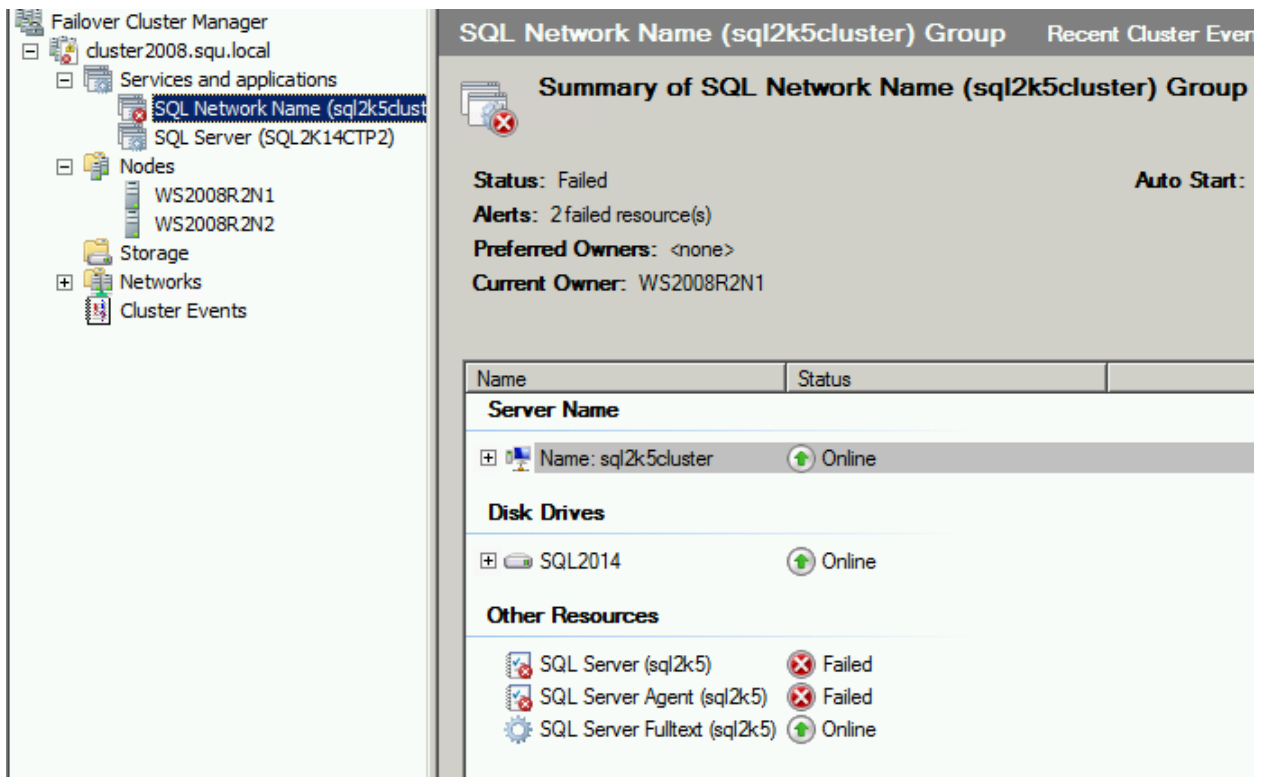


Figure 1: Upgrade causing an outage for another instance due to the resource DLL



Figure 2: Upgrade rule noting the resource DLL will be replaced

The ability to fail over (or not) is controlled by the instance's network name resource in the WSFC cluster. Specifically, if a given node can be a possible owner of the network name (and SQL Server was properly installed there), it can fail to that node. If a node is not a possible owner even if SQL Server was properly installed, failover is not possible. An example of not being able to fail over to a node is shown in Figure 3.

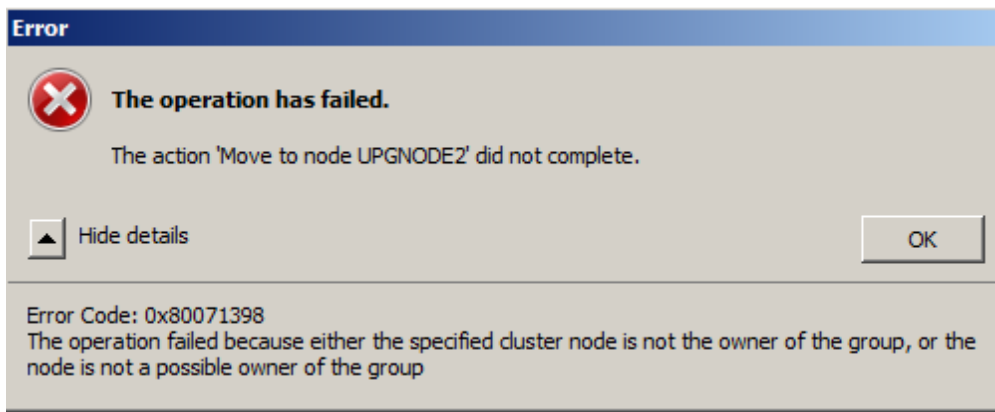


Figure 3: Error message noting you cannot move the FCI’s resources to another node
Possible owners can be found on the Advanced Policies tab of the network name resource. In Figure 4, all nodes in this WSFC cluster can potentially own SQL Server.

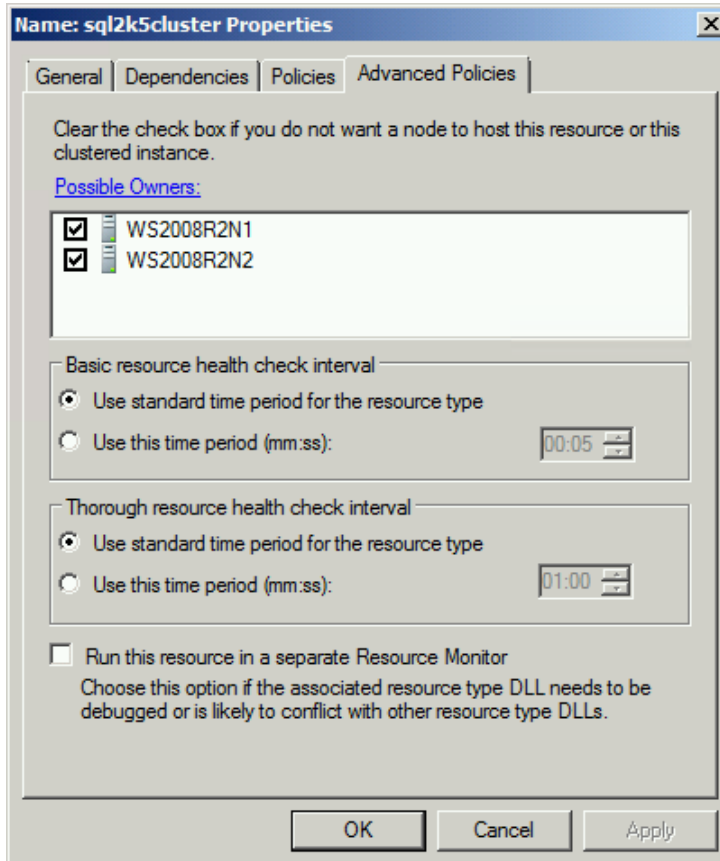


Figure 4: Possible owners for the network name resource

Here is the logic that Setup uses in an upgrade: As you upgrade nodes that do not own the resource, once you hit 50 percent (or greater) of those nodes being upgraded, Setup will initiate a failover automatically if you run the installer on the node that owns the SQL Server instance. If you have coordinated an outage, this will not be a problem.

However, if an outage is not expected, this could cause a problem with those expecting SQL Server to be up. Which node the instance fails over to is controlled by the logic of the WSFC cluster. It depends on which nodes are possible owners as well as the preferred owners as set on the SQL Server resource group. With multiple instances and in more mission critical environments, you may want to control this behavior yourself. You have a few methods at your disposal. The easiest way is to modify the possible owners of the network name resource yourself. Prior to upgrading a node that does not own the SQL Server resource, remove it as a possible owner. Using the same tab shown earlier, deselect the node. An example is shown in Figure 5.

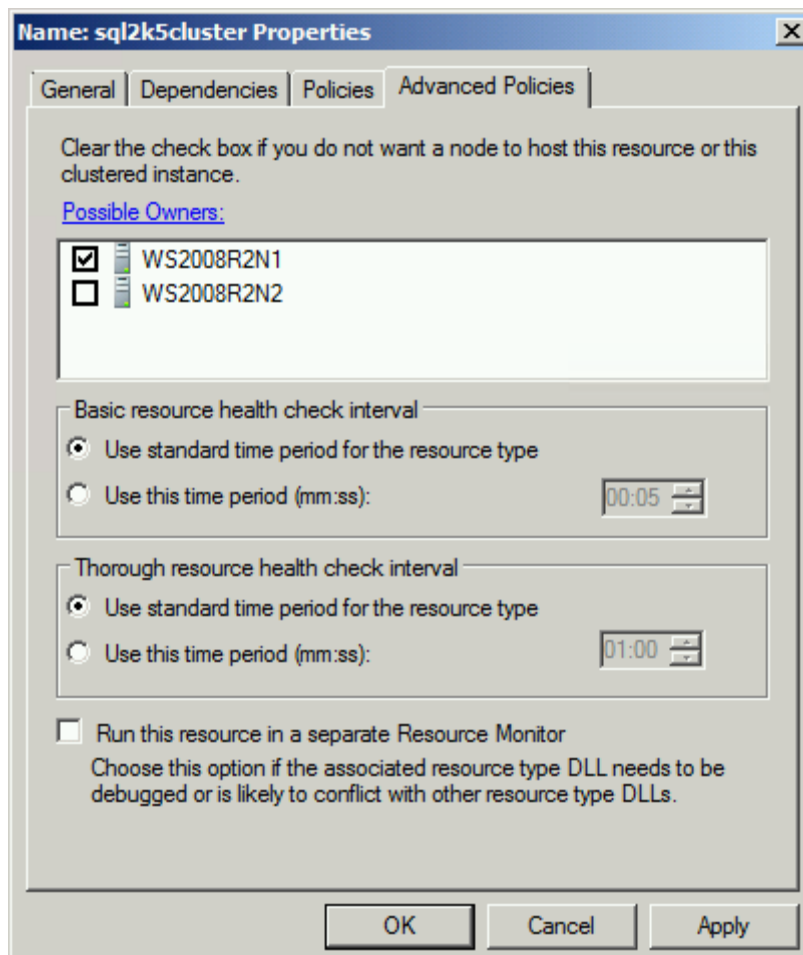


Figure 5: WS2008R2N2 removed as a possible owner to allow the upgrade

If you want to perform the same task in a scripted fashion, you can do it using cluster.exe or PowerShell. Note that cluster.exe is deprecated as of Windows Server 2008 R2, so you should use PowerShell where possible. The first thing you would need to do is figure out the name of the network name resource for that instance of SQL Server and ultimately what resource group it is in to be able to do a failover. Figure 6 shows an example using PowerShell.

```

Administrator: Windows PowerShell Modules
PS C:\> Get-ClusterResource

Name                               State                               Group                               ResourceType
----                               -
Cluster IP Address                 Online                             Cluster Group                      IP Address
Cluster IP Address 10.10.0.13     Online                             Cluster Group                      IP Address
Cluster Name                        Online                             Cluster Group                      Network Name
Quorum                             Online                             Cluster Group                      Physical Disk
SQL IP Address 1 (sql2k14C...      Online                             SQL Server (SQL2K14CTP2)          IP Address
SQL IP Address 1 (sql2k5c1...      Online                             SQL Network Name (sql2k5c1...     IP Address
SQL IP Address 2 (sql2k14C...      Online                             SQL Server (SQL2K14CTP2)          IP Address
SQL Network Name (sql2k14C...      Online                             SQL Server (SQL2K14CTP2)          Network Name
SQL Network Name (sql2k5c1...      Online                             SQL Network Name (sql2k5c1...     Network Name
SQL Server (SQL2K14CTP2)           Online                             SQL Server (SQL2K14CTP2)          SQL Server
SQL Server (sql2k5)                 Failed                             SQL Network Name (sql2k5c1...     SQL Server
SQL Server Agent (SQL2K14C...      Online                             SQL Server (SQL2K14CTP2)          SQL Server Agent
SQL Server Agent (sql2k5)          Failed                             SQL Network Name (sql2k5c1...     SQL Server Agent
SQL Server Fulltext (sql2k5)       Online                             SQL Network Name (sql2k5c1...     Generic Service
SQL2005                             Online                             SQL Server (SQL2K14CTP2)          Physical Disk
SQL2014                             Online                             SQL Network Name (sql2k5c1...     Physical Disk
PS C:\>

```

Figure 6: Listing the resources in a WSFC cluster with PowerShell

To modify the possible owners for a resource using PowerShell, you can pipe the resource through the `Get-ClusterOwnerNode` cmdlet to see the possible owners. To change the possible owners, use the `Set-ClusterOwnerNode` cmdlet with the `-Owners` parameter and list the nodes separated by commas. Unlike the `cluster.exe` implementation, the list you provide will be absolute—it’s not a simple matter of adding and removing. To control failover for a resource group, you would use the `Move-ClusterGroup` cmdlet with the `-Node` parameter, specifying the node to which you want to move the group. Explore these cmdlets further directly in PowerShell with the `Get-Help` cmdlet.

Considerations for Upgrading Multiple Instance Failover Clustering Deployments

As noted in the previous section, upgrading a cluster that has multiple instances is more challenging because you are potentially affecting more than just the instance you are looking to upgrade. This will require careful planning and coordination. Each instance will be affected by an outage at some point, even if the instance in question is not the one that is being upgraded (yet) to SQL Server 2014. This section walks you through an example of a cluster with two instances of SQL Server on two nodes. One instance is SQL Server 2005 SP4, and the other is SQL Server 2012. For this example, assume that an instance is running on each node, as shown in Figure 7.

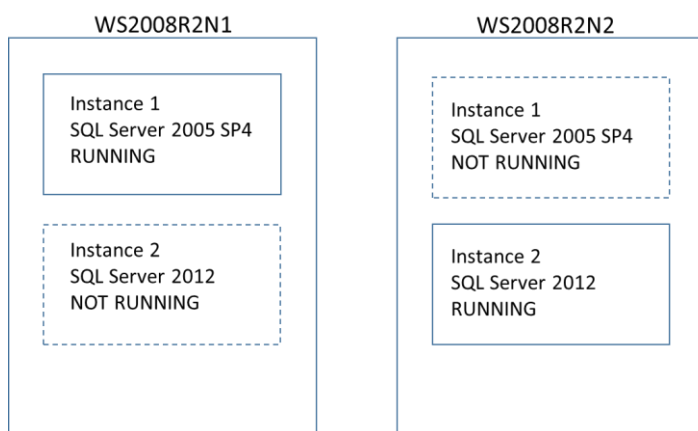


Figure 7: Initial state of the cluster

Note: With two nodes, there is not much room to move things around so you may incur multiple outages that need to be coordinated with the business, application owners, and end users. This is why having another node (or more) that is purely dedicated to failover gives you more flexibility.

As noted earlier, upgrading will replace the resource DLL that is shared among all instances. Whatever node you choose to start with, you will need to move the already running FCI on that node to the other one. This also assumes you have the capacity to run both instances with no issues on a single node. If you do not, upgrading may be hard to do.

In this example, we will start with Instance 1 (SQL Server 2005), so that it will be failed over to the other node. Once the failover occurs, upgrade the SQL Server 2005 instance. This will also upgrade any shared components at the same time, which would affect the SQL Server 2012 instance. Figure 8 shows what the cluster will look like.

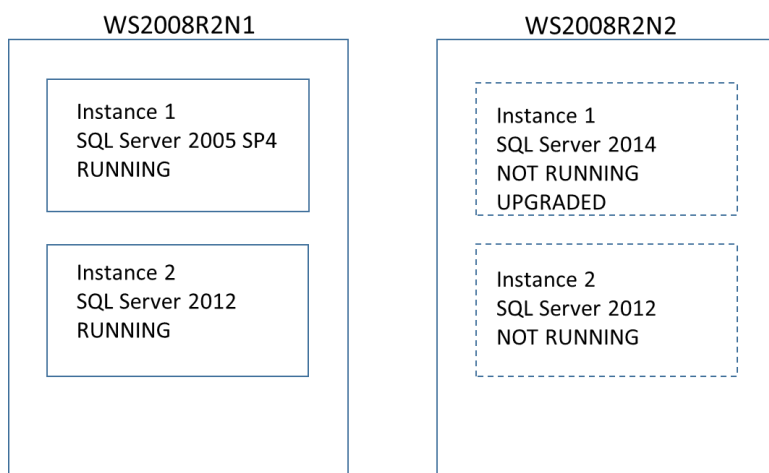


Figure 8: SQL Server 2005 upgraded on one node

At this point, you have two instances running on one node, with one half upgraded. To complete the upgrade for the 2005 instance to SQL Server 2014, it will need to be failed over to the other node. If you are looking to upgrade both the 2005 and 2012 instances at the same time, it may be best (especially in the interest of time) to upgrade the 2012 instance on the node that does not own the FCI. The cluster will now look like that in Figure 9.

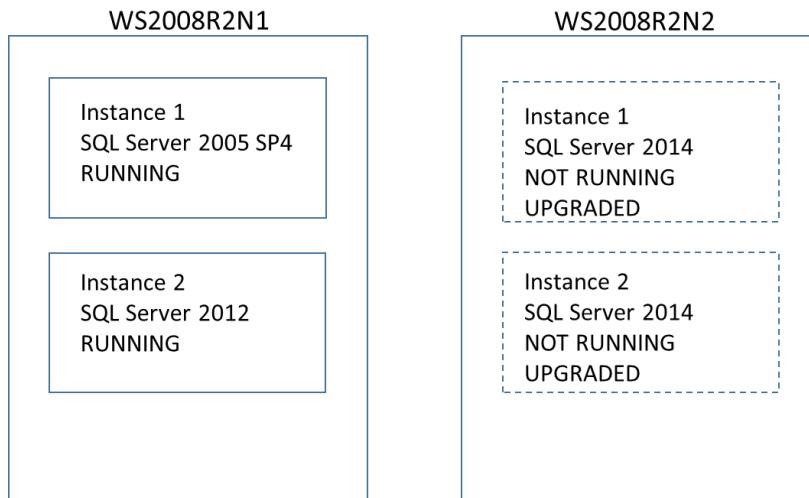


Figure 9: SQL Server 2014 CTP2 upgraded on one node (one node completed)

Since once node is completely upgraded to SQL Server 2014, two things must happen:

1. The instances in the midst of the upgrade process must be failed over to a node that has been upgraded in order to upgrade the internal components.
2. The binaries on the node not upgraded must be upgraded.

As noted earlier, you can control this process yourself or have Setup take care of it. If you have reached the “tipping point” where enough of the other nodes have been upgraded, Setup will fail the instance over to a valid node on its own but it may not be hosted where you prefer. In this example, there are only two nodes so that is not a big decision point, since there is only one place for the instances to go.

For this example, SQL Server 2005 will be upgraded first. However, as noted above, since the resource DLL must be replaced, the 2012 instance will be affected so it is recommended that you fail over both instances to the other node, which will upgrade both of them. The cluster will look like the one in Figure 10, and the first node is ready to be upgraded.

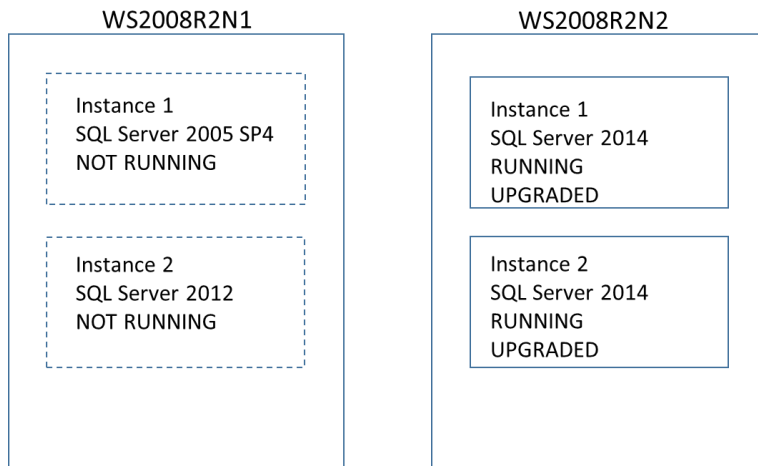


Figure 10: Manual failover of both instances to complete the internal upgrade

At this point, you have two working SQL Server 2014 instances that you can now test to ensure that the applications work correctly before opening them up for production use. This can be done while upgrading the instances on the node not owning the FCIs. Technically, you may even claim that you are done if the applications are verified, but it is always best to finish the upgrade on all nodes and test failover to ensure that nothing has changed post-upgrade. You do not want to get into a situation where you assumed that failover could happen, but found out weeks later that there was a problem in the failover process.

Although this example was relatively simple, it involved possible multiple outages. If the instances were to be upgraded at different times, one would still affect the other. The more complex your WSFC and FCI configuration, the more challenging the upgrade. Account for this in your planning.

Before Upgrading (Setup or Command Line)

Here are the tasks that you should perform before upgrading a clustered instance to SQL Server 2014:

1. Make sure the underlying Windows cluster is properly configured to standard best practices and is a fully supported cluster, as noted in the Knowledge Base articles mentioned earlier.
2. If necessary, create the security accounts in the domain for SQL Server 2014. Because it is not possible to alter the service accounts during the upgrade, we recommend that you change the older security accounts to SQL Server 2014-specific accounts post-upgrade. This step is optional.

3. Make sure that there are no errors in any of the logs (including SQL Server), that all resources can be failed over to each node, and that all names and IP addresses can be accessed. This task may need to be scheduled for a time that will not impact end users. The goal is to ensure that everything is working properly in the failover cluster before you attempt to install SQL Server 2014. If you have already validated failover at another time, it may be possible to skip this task.
4. If you have a planned outage window prior to the actual upgrade, you may want to consider ensuring that the prerequisites required for SQL Server 2014 are installed on the nodes. This will shorten the time required for the upgrade process. For example, if you are already using SQL Server 2008 R2 on Window Server 2008 R2 SP1, the only prerequisite you should need to install would be .NET Framework 4.0.

In-Place Upgrade for a Failover Clustering Instance

This section will cover the overall process and discuss some specific actions that will occur during an upgrade of a clustered instance of SQL Server. Therefore, this section does not include every Setup screen but rather the screens at key points in the upgrade process. Step-by-step instructions for using the Setup program for an in-place upgrade can be found in the topic [Upgrade a SQL Server Failover Cluster Instance \(Setup\)](http://msdn.microsoft.com/en-us/library/ms191295(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms191295\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms191295(v=sql.120).aspx)) in SQL Server 2014 Books Online.

1. Nodes that do not currently own the clustered SQL Server instance must be upgraded first. This allows SQL Server to have a place to fail the instance over to when the time is right. At that point, the internal upgrading of SQL Server will occur, which is only a brief outage. This process minimizes the downtime that you must incur from the upgrade process. To minimize the downtime involved in the upgrade, you must upgrade all the nodes that can possibly host the instance but do not currently own the instance that is being upgraded.

If you attempt to upgrade the node that currently owns the instance, you will see the message shown in Figure 11 when you reach the Cluster Upgrade Report dialog box. If you are trying to maximize your uptime, this warning is a clear message that you will incur downtime that you may not have accounted for.

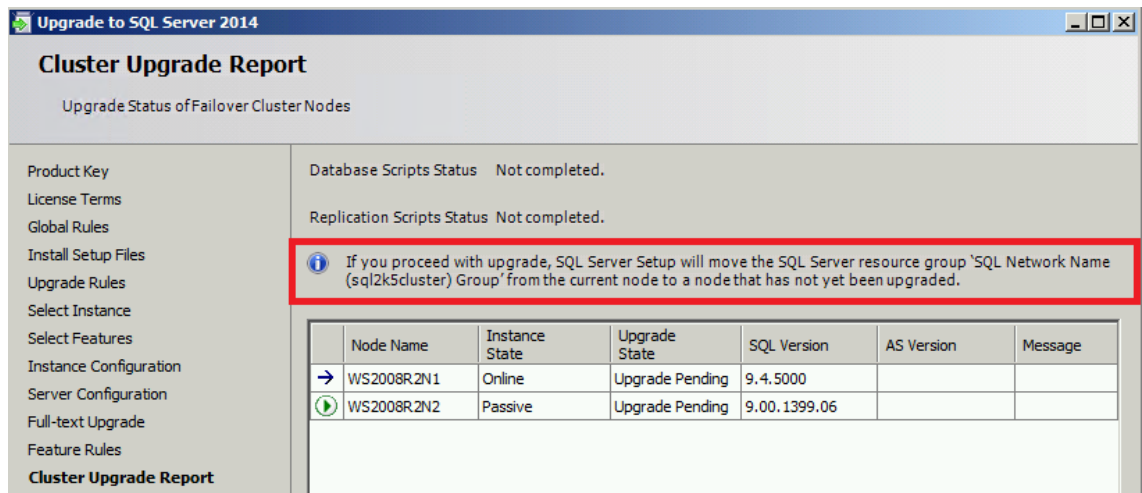


Figure 11: Message if you are attempting to upgrade the owner of the instance

2. During the upgrade process, you will first have to select which instance you are upgrading on that node. The instance choices will be displayed in the *Installed instances* table, as shown in Figure 12. You can select the instance to upgrade in the drop-down list. If there is only one instance that is eligible for upgrade, you will not be able to select anything else in the drop-down list.

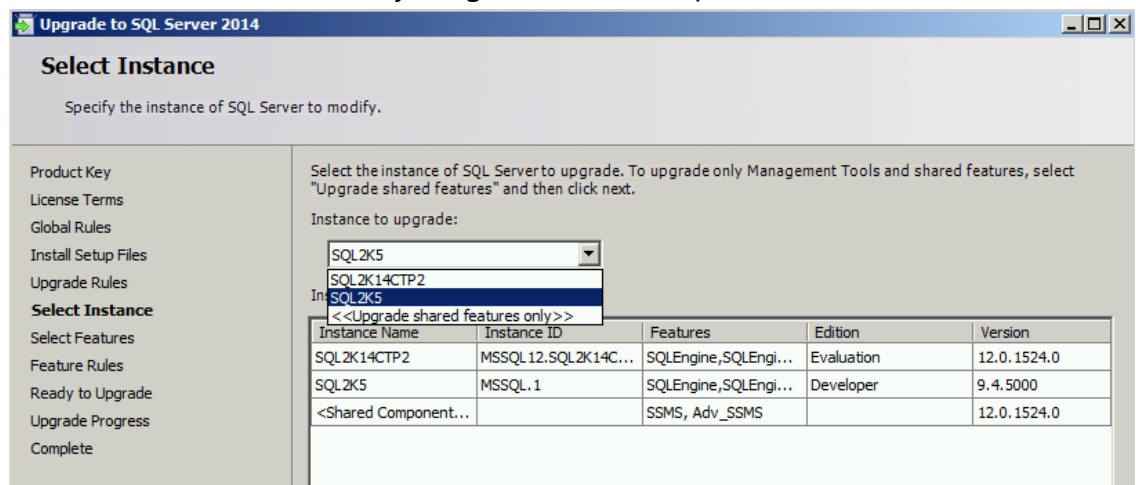


Figure 12: Choosing an instance to upgrade

3. As noted earlier, you will need to either enter a new Instance ID for the upgraded instance or leave the default value no matter what version of SQL Server you are coming from. The prefix will change to MSSQL12 before the Instance ID. For example, if the Instance ID is SQL2K5, it will appear as MSSQL12.SQL2K5 on the file system.

Important: It is crucial that the Instance ID used for an instance is the same for the upgrade of that instance on every node.

- Before the upgrade begins, Setup will display the Cluster Upgrade Report, as shown in Figure 13. This report shows the status of where you are in the process.

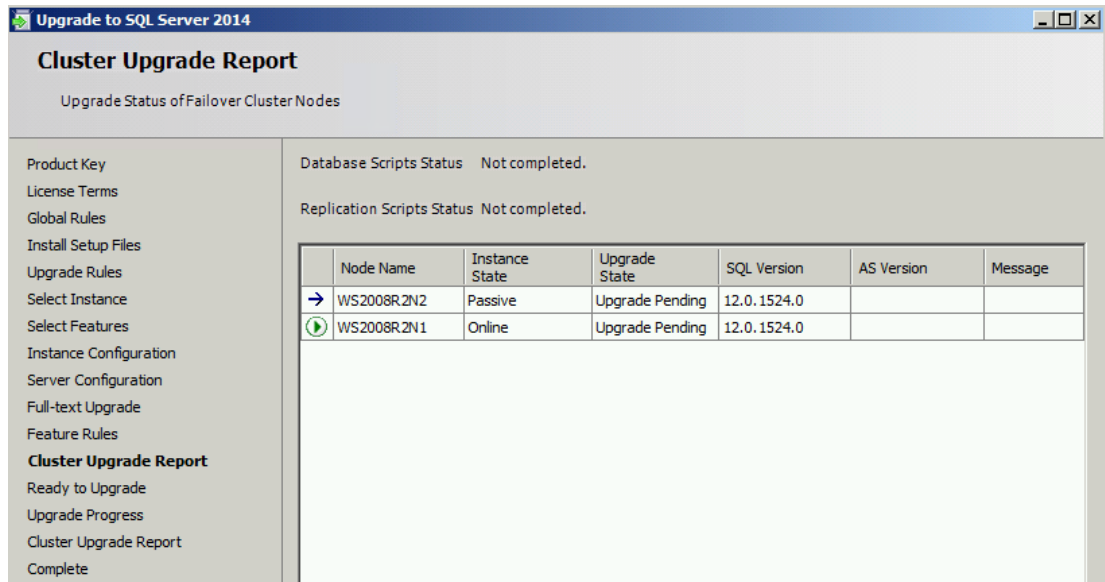


Figure 13: Initial Cluster Upgrade Report showing that nothing is upgraded

After the upgrade of a node is complete, the Cluster Upgrade Report will be displayed again with an updated status, as shown in Figure 14. Note that WS2008R2N2 now reflects the RTM version number of SQL Server 2014.

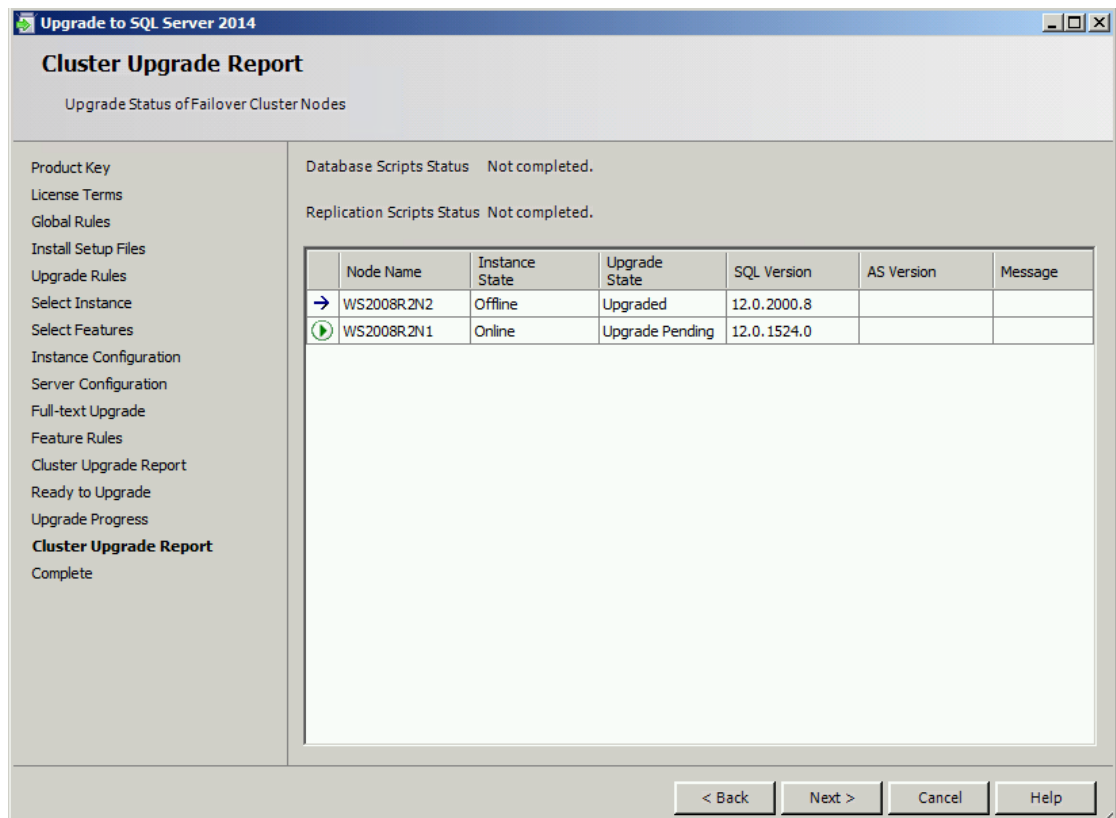


Figure 14: Updated Cluster Upgrade Report after upgrading a node

Warning: At this point, it is not possible to fail over the older SQL Server clustered 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.

5. Upgrade at least half of the nodes that do not own the FCI you are currently upgrading.
6. Once at least half of the nodes have been upgraded for an FCI, stop all traffic to the instance of SQL Server to ensure that no one tries to access the instance during the upgrade process. While SQL Server will only have a brief outage during the final upgrade steps, it is always a best practice to ensure that no changes happen during the upgrade period to ensure a consistent state before and after.
7. Follow the instructions in [Upgrade a SQL Server Failover Cluster Instance \(Setup\)](http://msdn.microsoft.com/en-us/library/ms191295(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms191295\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms191295(v=sql.120).aspx)) 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 one of the already upgraded nodes (unless you have done that manually), as the notification in Figure 15 shows. Figure 16 shows the failover during the upgrade. Figure 17 shows the instance brought online on another node.

SQL Server will be unavailable during the failover process and while the databases are being upgraded. Sometimes there is no need to stop the traffic. Setup will automatically handle the failover to the upgraded node. On average this takes about two minutes, depending on the hardware and other configurations.

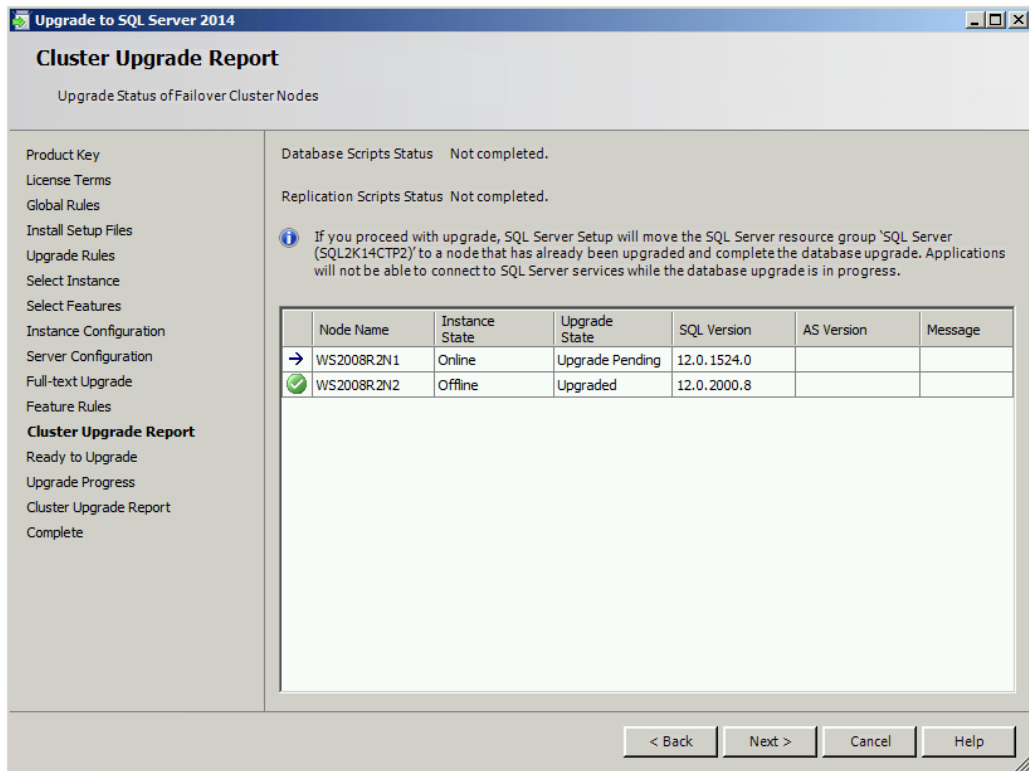


Figure 15: Cluster Upgrade Report when the instance itself is upgraded

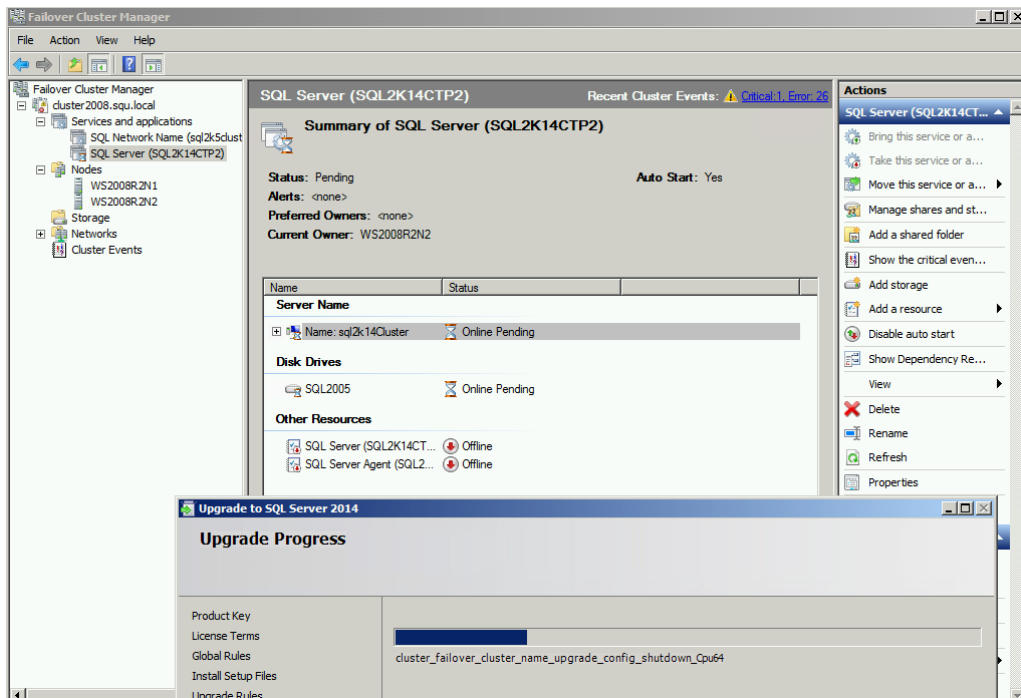


Figure 16: The failover during the upgrade

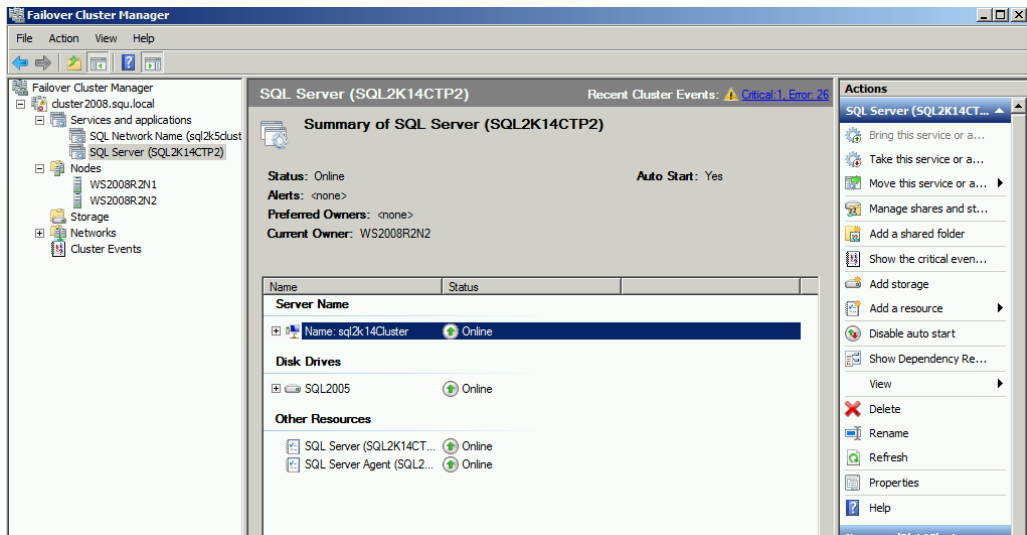


Figure 17: Completed failover and upgrade of the services

- When the upgrade is complete, Setup will show a final Cluster Upgrade Report, similar to the one in Figure 18.

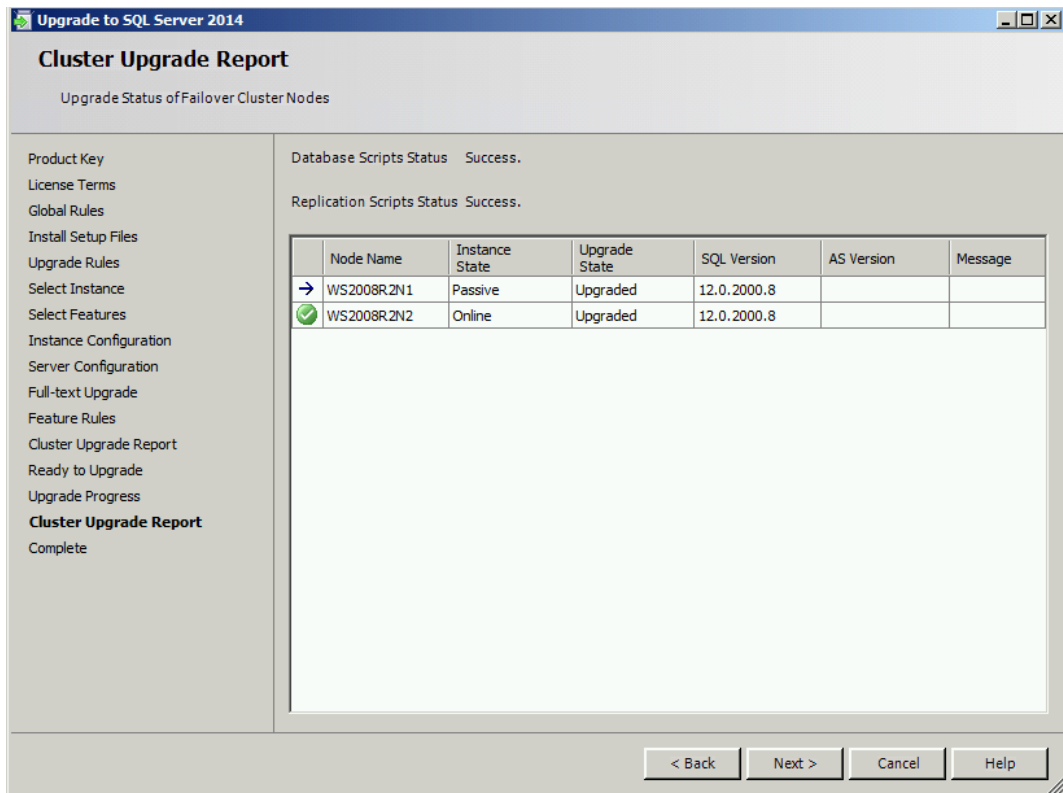


Figure 18: Final Cluster Upgrade Report after a successful upgrade

- If necessary, upgrade any nodes that have not been upgraded that could own the SQL Server 2014 instance.

The upgrade is now technically complete from a SQL Server upgrade perspective. Note that the upgraded instance was not automatically moved back to the original node it was hosted on. 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. You should also perform the following tasks post-upgrade for each FCI:

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 120 (if necessary).
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. Even if you do not plan on using availability groups immediately, enable the feature so that they can be used later. Availability groups are enabled at the service level and require a restart of SQL Server, which is why this is an opportune time to perform this task. For more information, see the SQL Server 2014 Books Online topic [Enable and Disable AlwaysOn Availability Groups \(SQL Server\)](http://technet.microsoft.com/en-us/library/ff878259(v=sql.120).aspx) ([http://technet.microsoft.com/en-us/library/ff878259\(v=sql.120\).aspx](http://technet.microsoft.com/en-us/library/ff878259(v=sql.120).aspx)).

Once all of these tasks are complete, verify that the upgrade is successful. Open the instance for testing the applications; when those are certified, allow production use.

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—basically follows the same steps as an in-place upgrade. Use one of the methods listed in the "Methods for Side-by-Side Upgrades to a Separate Server or Cluster" section to move and upgrade the databases in conjunction with installing a fresh instance of SQL Server 2014. 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 sample syntax for upgrading an instance via the command line. 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 using setup.exe only. For more information on all of the parameters available during an upgrade, see [Install SQL Server 2014 from the Command Prompt](http://msdn.microsoft.com/en-us/library/ms144259.aspx) (<http://msdn.microsoft.com/en-us/library/ms144259.aspx>).

There is one optional, but very specific parameter, that applies to failover clustering instance upgrades: FAILOVERCLUSTERROLLOWNERSHIP. Setting a value for this parameter controls the behavior of failover (as described in the "Instance Failover Behavior During the Upgrade Process" section). The three possible values for FAILOVERCLUSTERROLLOWNERSHIP are:

- 0 = Will not allow failover to one of the already upgraded nodes to upgrade the instance itself, nor will it add the node to the list of possible owners when the upgrade is completed. You will have to do failover and modification of the possible owners on your own.
- 1 = Will allow failover to one of the already upgraded nodes to upgrade the instance itself, and will it add the node to the list of possible owners when upgrade is completed.
- 2 = SQL Server determines if 0 or 1 is needed.

The default value is 2. Controlling behavior is only available via a command-line upgrade.

To use the command prompt and not a configuration file, use a slash (/) before each option and separate with a space. For strings that might contain spaces or odd characters, use double quotation marks. Here is an example of using the command line to upgrade a default instance showing all output in the command window:

```
D:\>setup.exe /q /ACTION=Upgrade /ERRORREPORTING=0  
/IACCEPTSQLSERVERLICENSETERMS /INSTANCENAME=MSSQLSERVER /INSTANCEID=SQL2K5INS  
/INDICATEPROGRESS=True /FAILOVERCLUSTERROLLOWNERSHIP=2 /FTUPGRADEOPTION=Reset
```

Here is the sample syntax for upgrading a named instance using a command file:

```
D:\>setup.exe /CONFIGURATIONFILE="c:\UpgradeSQL.ini"
```

This is in the configuration file:

```
[OPTIONS]
ACTION=UPGRADE
ERRORREPORTING=0
IACCEPTSQLSERVERLICENSETERMS=TRUE
INDICATEPROGRESS=TRUE
INSTANCEID=MSSQLSERVER2100
INSTANCENAME=MSSQLSERVER
QUIET=TRUE
SQMREPORTING=0
UPDATEENABLED=FALSE
```

During the upgrade, you will see what is going on with the upgrade if the QUIET (or /Q) and INDICATEPROGRESS actions are used (such as in the examples above).

A successful end result will end with a Setup result of 0. A Setup result of 0 generally means that no reboot is required. Any other status often means either an error (which would be clearly indicated as such) or a possible reboot.

When the upgrade is complete on a node, regardless of what status you see, you should check the Setup logs which can be found on the local node in the SQL Server directory for the binaries. The location, assuming a C drive, would be C:\Program Files\Microsoft SQL Server\120\Setup Bootstrap\Log. There you will find Summary.txt, which is a summarized view of what took place. If you have errors or want to see more detail, look at Detail.txt in the dated folder found under Log that is associated with the upgrade.

Tip: The command line or file execution is the same (outside of the one parameter for failover behavior) for standalone or clustered instances. If you have a lot of instances and nodes, scripting the upgrade will be much easier since for a single instance, the upgrade commands will be the same. It will also save a lot of time since you do not need to click through the UI.

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 [Upgrade to SQL Server 2014](http://msdn.microsoft.com/en-us/library/bb677622(v=sql.120).aspx) page ([http://msdn.microsoft.com/en-us/library/bb677622\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/bb677622(v=sql.120).aspx)).

Upgrading AlwaysOn Availability Groups

This section covers how to upgrade SQL Server 2012 AlwaysOn Availability Groups to SQL Server 2014. AlwaysOn Availability Groups were a new feature in SQL Server 2012. They are the replacement for database mirroring. For details and best practices on upgrading AlwaysOn Availability Groups, see [Upgrade and Update of Availability Group Servers with Minimal Downtime and Data Loss](http://msdn.microsoft.com/en-us/library/dn178483.aspx) (<http://msdn.microsoft.com/en-us/library/dn178483.aspx>). Upgrading servers using AlwaysOn Availability Groups is very straightforward, but it does require that the servers be upgraded in the correct order.

Feature Changes in SQL Server 2014 AlwaysOn Availability Groups

SQL Server 2014 AlwaysOn Availability Groups have several enhancements in SQL Server 2014, including:

- Replicas can now be easily added for Azure-hosted SQL Server installations. For more information, see [Use the Add Azure Replica Wizard \(SQL Server\)](http://msdn.microsoft.com/en-us/library/dn463980.aspx) (<http://msdn.microsoft.com/en-us/library/dn463980.aspx>).
- You are now allowed to have up to eight secondary replicas, rather than four in SQL Server 2012.
- When disconnected from the primary replica or during cluster quorum loss, readable secondary replicas now remain available for read-only workloads.
- A new system function, [sys.fn_hadr_is_primary_replica](#), and a new DMV, [sys.dm_io_cluster_valid_path_names](#), are available.

Considerations for Upgrading a SQL Server 2012 or SQL Server 2014 CTP2 Server with AlwaysOn Availability Groups

The process of upgrading AlwaysOn Availability Groups to SQL Server 2014 is basically the same as applying service packs to SQL Server 2012. The availability group will be upgraded during the normal database engine upgrade process. The following are the recommended best practices:

- Before starting the upgrade
 - Ensure full database backups of all user and system databases are current.
 - Run DBCC CHECKDB on every database.
 - Ensure failover is working correctly by performing manual failovers to at least one of the synchronous-commit replicas.

- Ensure that the state of the availability group is SYNCHRONIZED before beginning the upgrade.
- Change the backup preferences to only back up from the primary replica until the upgrade process is complete.
- Turn off automatic failover to synchronous-commit replicas so that unintended failovers during the upgrade process do not occur.
- Upgrade remote secondary replicas first.
- Upgrade local secondary replicas next.
- Fail over the availability group to an upgraded secondary synchronous-commit replica server. If you fail over to an asynchronous-commit secondary replica, data loss may occur.
- Upgrade the primary replica.

If you upgrade the primary replica before the secondary replicas, transaction log entries will no longer be sent to the non-upgraded secondary servers. You will have to upgrade the secondary servers and then re-establish the newly upgraded servers as part of the availability group.

Considerations for Upgrading an Availability Group with a Single Remote Secondary Replica

One configuration for AlwaysOn Availability Groups is to have a primary site in a data center and a remote site with a secondary replica of the database(s) that make up your availability group. Typically, this replica is done using asynchronous-commit so that clients on the primary server are not affected by failures in the remote site or network connectivity issues with the remote site. This is usually done strictly for disaster recovery purposes.

With this configuration, you will always have to upgrade the secondary server first, as previously mentioned. However, you then have a decision to make. You can schedule the downtime and simply upgrade the primary server (your databases will be unavailable during the upgrade period), or you can fail over to the remote site during the upgrade, and then fail back to the primary site once the primary site's SQL Server 2014 upgrade is complete.

If you choose the remote site failover option, you must perform the following steps:

1. Complete the upgrade of the remote server to SQL Server 2014.
2. Change the commit mode of the availability group to synchronous commit.
3. Wait until the synchronization state is SYNCHRONIZED, as shown in Figure 19.

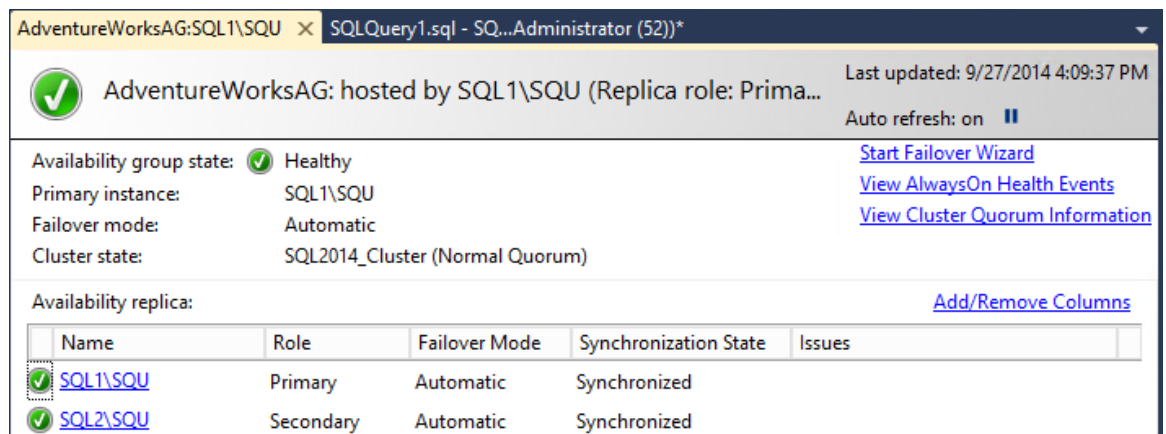


Figure 19: A synchronized availability group

4. Fail over the availability group to the remote server.
5. Upgrade the primary server.
6. Fail over the availability group back to the primary server.
7. Return the commit mode to asynchronous commit.

Considerations for Upgrading SQL Server Instances with Multiple Availability Groups

If you have more than one availability group in your configuration, it is likely you will not have all availability groups hosted on a single primary server. However, the same rules mentioned previously apply. A secondary server must be upgraded first before the primary server may be upgraded. Therefore, you may have to fail over availability groups before beginning the upgrade to ensure that at least one server is not hosting any primary workloads. Then, you would upgrade the newly made secondary server. Afterward, you can fail over workloads to that newly upgraded server and upgrade their peers in the availability group configuration. An example may help with this.

Imagine you have three availability groups spread across three SQL Server instances, as shown in Table 2.

Table 2: Three Availability Groups Spread Across Three SQL Server Instances

	SQL1	SQL2	SQL3
AvGroup1	Primary	Secondary	Secondary
AvGroup2	Secondary	Primary	Secondary
AvGroup3	Secondary	Secondary	Primary

In order to upgrade this availability group configuration, you could perform the following steps (or some variation that makes sense in your configuration):

1. Fail over AvGroup3 to SQL2.
2. Upgrade SQL3 because it now hosts only secondary replicas.
3. Fail over AvGroup 1 to SQL3.
4. Upgrade SQL1.
5. Fail over AvGroup2 and AvGroup3 to SQL1.
6. Upgrade SQL2.
7. Fail back the availability groups to where they started.

Upgrading Mirrored Databases

This section covers how to upgrade instances of SQL Server 2005, SQL Server 2008, SQL Server 2008 R2, SQL Server 2012, and pre-release SQL Server 2014 to SQL Server 2014 RTM (or later) when databases are configured with database mirroring. Similar to an upgrade with FCIs, database mirroring allows you to perform a rolling upgrade in which you can upgrade one component at a time (in this case, an instance). Upgrading is the only scenario where mixed versions in the same database mirroring configuration are supported with database mirroring.

Feature Changes in SQL Server 2008 R2 Database Mirroring

Database mirroring is the same as it was in SQL Server 2008 R2 and has not been enhanced. However, it is important to point out that the feature was officially deprecated as of SQL Server 2012. See the section “Features Not Supported in a Future Version of SQL Server” in the SQL Server 2014 Books Online topic [Deprecated Database Engine Features in SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms143729.aspx) (<http://msdn.microsoft.com/en-us/library/ms143729.aspx>) for more details.

Because database mirroring will not be immediately removed as a feature, you can still continue to use it for now. Over time, you should look at migrating any database that uses database mirroring to a feature that will be supported. Your two choices will be log shipping and the AlwaysOn Availability Groups feature. The edition of SQL Server 2014 that you deploy will determine which feature you will be able to use.

In-Place Upgrade

There are two main configurations for database mirroring: high-performance (asynchronous) and high-safety (synchronous). High-safety can also be configured with an optional witness to allow automatic failover. The steps for both are similar and will be combined in this section. Any differences will be explicitly pointed out.

1. If database mirroring is configured in high-performance mode, it must be changed to high-safety without a witness. The main reason for this is that it will ensure all transactions are committed so there will be no data loss. If you require high-performance mode, you can reconfigure it after the upgrade. To change the mode from high-performance to high-safety in SSMS, right-click the database in Object Explorer, select Tasks, and then select Mirror. In the Mirroring page of the Database Properties dialog box, choose the mode. Alternatively, you can use the T-SQL command:

```
ALTER DATABASE database SET PARTNER SAFETY FULL
```

2. If high-safety mode with a witness is configured, remove (unconfigure) the witness. This will prevent a failover during the upgrade because the principal will be unavailable during the upgrade of that instance.
3. Upgrade the instance containing the mirror. The mirroring session will then synchronize. After synchronization, the mirroring session will look similar to that shown in Figure 20, which displays it in both SSMS and the Database Mirroring Monitor.

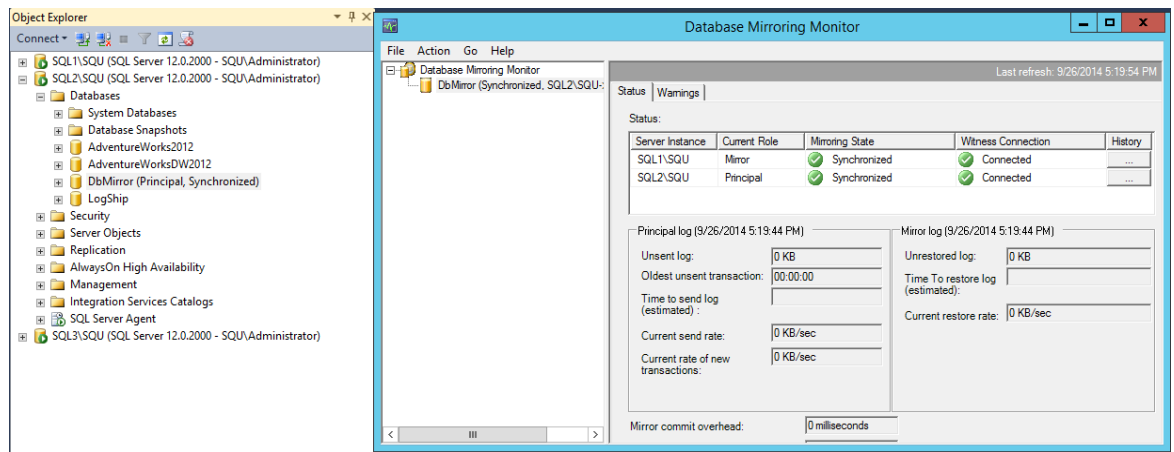


Figure 20: Synchronized mirroring session displayed in SSMS and the Database Mirroring Monitor

4. At this point, you can technically upgrade the principal but first ensure all traffic is stopped to the database so nothing will connect to it.
5. Use SSMS or T-SQL to manually fail over the mirroring session to the instance that was upgraded. To do this in SSMS, click the Failover button in the Mirroring page of the Database Properties dialog box for the database mentioned earlier. To do this in T-SQL, use the following command:

```
ALTER DATABASE database SET PARTNER FAILOVER
```

At this point, the mirroring session will be suspended, as shown in Figure 21.

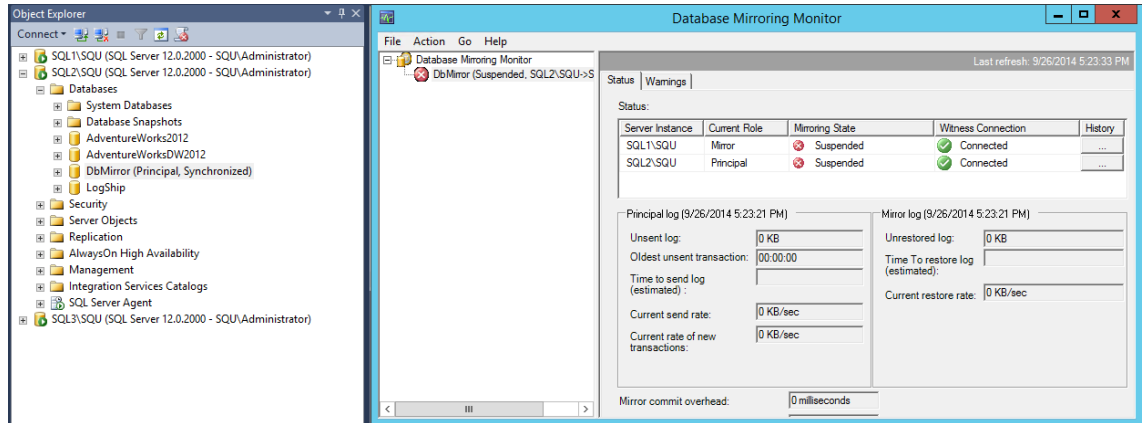


Figure 21: Suspended database mirroring session

After the failover occurs, the mirror database will go through its upgrade process and, once complete, will technically be ready for use.

6. If you want the now principal and former mirror to be the new principal production server, you must get it fully ready for use. That entails tasks such as:
 - Synchronizing logins and ensuring that all jobs and objects residing outside the database are restored via scripts to make the mirror usable
 - Changing the compatibility mode of the database to 120
 - Configuring SQL Server Agent jobs
 - Running health checks (e.g., DBCC CHECKDB) to ensure the health of the newly upgraded database

Optionally, you could wait until the old primary is upgraded and then fail back. That may be advantageous since you will be able to test the failover to ensure things are working post-upgrade, but you may not be able to tolerate the longer outage.

Assuming the newly upgraded database will be the new principal, ensure that the application testing is complete and then redirect all users and applications to the upgraded instance containing the new principal (the old mirror). This will minimize an outage because the old principal (the new mirror) will not be available during its upgrade.

7. Upgrade the instance containing the old principal to SQL Server 2014.
8. While mirroring should automatically start in most cases, you may have to manually re-establish the suspended mirroring session. To do this, click the

Resume button in the Mirroring page of the Database Properties dialog box, as shown in Figure 22.

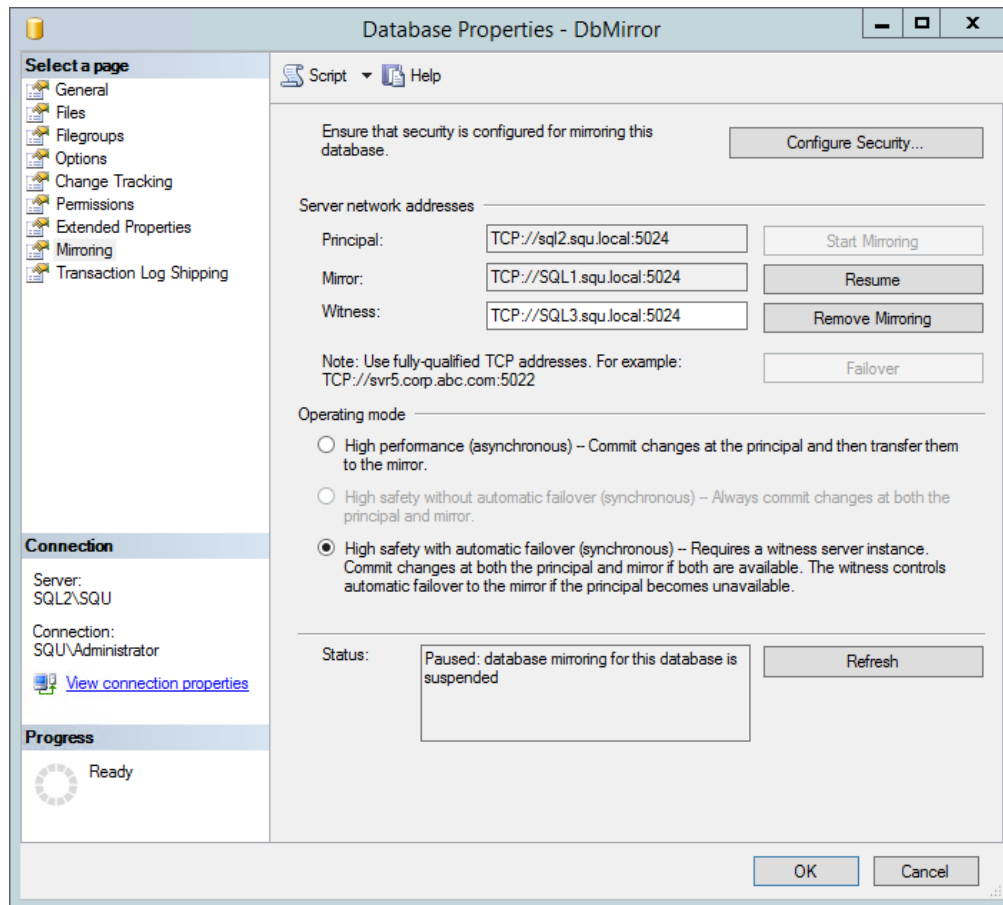


Figure 22: Clicking the Resume button in SSMS

To do this in T-SQL, use the following command:

```
ALTER DATABASE <database> SET PARTNER RESUME
```

The databases will now start synchronizing the unsent log.

9. If you have not started to use the old mirror (now principal) as the primary, test failover by failing back to the now upgraded original primary. If you want that to be the primary again, unless you have made changes, you hopefully will not have to do anything detailed in Step 6.
10. If necessary, change the mode from high-safety back to high-performance using SSMS or T-SQL. The T-SQL command to use is:

```
ALTER DATABASE <database> SET PARTNER SAFETY OFF
```
11. If necessary, add the witness back to the high-safety configuration. Follow the instructions in the SQL Server 2014 Books Online topic [Add or Replace a](#)

[Database Mirroring Witness \(SQL Server Management Studio\)](#)

(<http://msdn.microsoft.com/en-us/library/ms365603.aspx>).

Side-by-Side Upgrade to a New Server

The following steps describe how to upgrade to SQL Server 2014 when you have an existing database mirroring configuration deployed and will be using a new server for SQL Server 2014, where you will be re-establishing database mirroring. However, as noted previously, it is strongly recommended to consider deploying an AlwaysOn availability group since database mirroring is deprecated.

1. Install the new SQL Server 2014 instances for the principal and mirror.
2. Take a full backup of the existing SQL Server principal database and restore two copies: one on the new principal and one on the new mirror. Use the WITH NORECOVERY option for these backups.
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 on both the principal and mirror. Use the WITH NORECOVERY option.
5. Back up the tail of the log on the principal.
6. Copy the tail log backup to the both new principal and mirror. Restore it on the new principal (use the WITH RECOVERY option) and on the new mirror (use the WITH NORECOVERY option).
7. Get the new production server fully ready for use. This entails tasks such as:
 - Synchronizing logins and ensuring that all jobs and objects residing outside the database are restored via scripts to make the mirror usable
 - Changing the compatibility mode of the database to 120
 - Configuring administration such as SQL Server Agent jobs
 - Running health checks (e.g., DBCC CHECKDB) to ensure the health of the newly upgraded database
8. Configure database mirroring from the principal to the mirror. See the SQL Server 2014 Books Online topic [Setting Up Database Mirroring \(SQL Server\)](#) (<http://msdn.microsoft.com/en-us/library/ms190941.aspx>) for more information.

9. Test the failover from the principal to the mirror and from the mirror back to the principal.
10. Redirect all users and applications to the new SQL Server 2014 instance containing the principal.

Additional References for Upgrading with Mirrored Databases

For more information about upgrading with mirrored databases, see the SQL Server 2014 Books Online topic [Minimize Downtime for Mirrored Databases When Upgrading Server Instances](http://msdn.microsoft.com/en-us/library/bb677181.aspx) (<http://msdn.microsoft.com/en-us/library/bb677181.aspx>).

For an up-to-date collection of additional references for upgrading with mirrored databases, see the [Upgrade to SQL Server 2014](http://msdn.microsoft.com/en-us/library/bb677622(v=sql.120).aspx) page ([http://msdn.microsoft.com/en-us/library/bb677622\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/bb677622(v=sql.120).aspx)).

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 2014 Log Shipping

Log shipping functionality remains the same as it was in SQL Server 2012. There are no new enhancements in SQL Server 2014. Like database mirroring, a log shipping configuration can be converted to an AlwaysOn availability group in a similar fashion if you want to take advantage of the new functionality that the feature brings to the table.

Log Shipping Upgrade Scenarios

When it comes to upgrading databases participating in log shipping, there are two main scenarios: 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.

Performing a role change will minimize downtime to applications and end users. Once the secondary instance is upgraded, bringing the database online will upgrade it to SQL Server 2014 and enable it for use. That means no traffic can access the old primary and the newly upgraded secondary will no longer be able to accept transaction logs. However, because log shipping does not account for the server name switch, this may not be an option for some since the applications used may involve changing the

application configuration or possibly each desktop instead of a central connection on an application server. In this scenario, you would also have to re-establish log shipping from this new primary to any secondary. This work may not be trivial and must be accounted for when deciding what course of action to take.

One thing that may tip the balance toward a role change is that even though there will be a clear cutover point, leaving the original primary at the older version gives you a fallback plan in the event something goes wrong. The data will remain at the point of the switch. Should you need to go back to the previous version of SQL Server, you would need to devise a way to get any changes made to the data in the SQL Server 2014 database.

If you do not perform a role change, the secondary (or secondaries) will remain in a loading state. A database upgrade is a logged operation, so once the primary database is upgraded and the transaction logs are being copied and restored, the secondary will be upgraded as well. Upgrading without a role change allows you to maximize your uptime on the primary, but you will have downtime since the primary will be completely unavailable during its upgrade to SQL Server 2014. Having said that, a benefit of not having a role change is that the log shipping configuration will remain intact after the upgrade.

Upgrading with Multiple Secondaries

Unlike database mirroring, log shipping allows you to have multiple secondaries from a single primary database. Having multiple secondaries will complicate the upgrade scenario since it is more “stuff” to deal with. With more than just the primary, a single secondary, and optional 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—without a role change or with 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.

Steps to 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 log shipping copy and restore jobs on all instances containing a secondary. The copy job will have a name like *LSCopy_<source instance>_<database name>* and the restore job will have a name similar to *LSRestore_<source instance>_<database name>*. An example is shown in Figure 23.

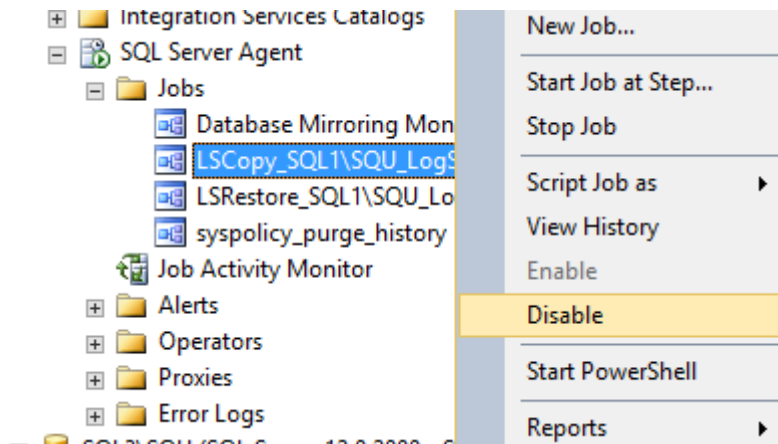


Figure 23: Disabling the copy job

2. Disable the alert job on the monitor (if it is used) or on the secondary. It will have a name like *LSAlert_<instance name>*.
3. Upgrade the instance containing the secondary database participating in log shipping. As noted earlier, because the database is in a state where it is restoring transaction logs and the primary has not been upgraded yet, it will not be upgraded to SQL Server 2014 yet.
4. If applicable, upgrade the monitor instance to SQL Server 2014.
5. Upgrade any other secondaries that may be participating in this log shipping configuration.
6. You now need to apply any transaction logs generated during the upgrade of the secondary instance. You have two options:
 - Manually copy over and restore all transaction log backups generated. Use the WITH NORECOVERY option.
 - Re-enable the copy and restore jobs and monitor their progress until things are caught up. Enable the jobs in the same way you disabled them. This is the only scenario where a mixed version log shipping configuration is supported.

7. 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.
8. Disable the transaction log backup job on the primary.
9. Upgrade the instance containing the primary database.
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 primary database.
11. Enable the existing Log Shipping transaction log backup and alert 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 2014.
12. Verify that the database compatibility level of the upgraded databases is 120.
13. Direct all users and applications to the original primary database.

Steps to 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. If you have multiple secondaries:

1. Disable the log shipping copy and restore jobs on all instances containing a secondary. The copy job will have a name like *LSCopy_<source instance>_<database name>* and the restore job will have a name similar to *LSRestore_<source instance>_<database name>*.
2. Disable the alert job on the monitor (if it is used) or on the secondary. It will have a name like *LSAlert_<instance name>*.
3. Upgrade each instance containing the secondary database. Because the log shipped database is in a state where it can restore transaction logs, it will not be upgraded to SQL Server 2014 yet.
4. On the primary, stop all incoming traffic to the primary and verify that there are no connections to ensure that, at this point, the data cannot be changed. Consider using single-user mode.

5. Disable the transaction log backup job on the primary, which will have a name like *LSBackup_<database_name>*.
6. Apply any transaction logs generated during the upgrade of the secondary instance(s). You have two options:
 - Manually copy over and restore all transaction log backups generated during the secondary upgrade. Use the WITH NORECOVERY option.
 - Re-enable the copy and restore jobs and monitor their progress until things are caught up. You can also manually start them.
7. On the current 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 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; leave the database online. However, by using WITH NORECOVERY, this instance and the database can become a secondary to the soon-to-be primary.
8. Copy and restore the tail of the log using WITH RECOVERY. As part of the restore, it will be upgraded to SQL Server 2014.
9. Get the new production server fully ready for use. This entails tasks such as:
 - Synchronizing logins and ensuring that all jobs and objects residing outside the database are restored via scripts to make the mirror usable
 - Changing the compatibility mode of the database to 120
 - Configuring administration such as SQL Server Agent jobs
 - Running health checks (e.g., DBCC CHECKDB) to ensure the health of the newly upgraded database
10. Delete the old copy and restore jobs on the former secondary database.
11. If applicable, upgrade the monitor instance to SQL Server 2014.
12. Delete the alert job from the monitor instance or the original secondary.
13. Upgrade the original primary to SQL Server 2014. Because there is a new primary, consider dropping the old primary database so that it will not go through the upgrade process. This will save time during the upgrade.
14. Delete the backup job for the original primary database on the original primary instance (the soon-to-be new secondary).

15. On the new primary (the old secondary), configure log shipping to the new secondary (the old primary) using the instructions found in the SQL Server 2014 Books Online topic [Configure Log Shipping \(SQL Server\)](http://msdn.microsoft.com/en-us/library/ms190640.aspx) (<http://msdn.microsoft.com/en-us/library/ms190640.aspx>).

Steps to Upgrade Side-by-Side to a New Server or Cluster

There are two scenarios for a side-by-side upgrade from a previous version of SQL Server: new hardware for both the primary and secondary, or new hardware for the primary only. Because secondaries can easily be initialized with an appropriate full backup, those steps are not covered here.

New Hardware for Both the Primary and Secondary

This option assumes that the instances for both the primary and warm standby will be brand new.

1. Install the new instances of SQL Server 2014 on the newly configured servers.
2. Take a full backup of the older primary database and restore two copies: one on the new primary and one on the new secondary. Use the WITH NORECOVERY option.
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 on both the primary and secondary. Use the WITH NORECOVERY option.
5. Back up the tail of the log on the primary.
6. Copy the tail log backup to the new primary and restore using the WITH RECOVERY option.
7. Copy the tail log backup to the new secondary and restore using the WITH NORECOVERY option.
8. Get the new production server fully ready for use. This entails tasks such as:
 - Synchronizing logins and ensuring that all jobs and objects residing outside the database are restored via scripts to make the mirror usable
 - Changing the compatibility mode of the database to 120
 - Configuring administration such as SQL Server Agent jobs

- Running health checks (e.g., DBCC CHECKDB) to ensure the health of the newly upgraded database
9. Configure log shipping between the SQL Server 2014 instances using the instructions found in the SQL Server 2014 Books Online topic [Configure Log Shipping \(SQL Server\)](http://msdn.microsoft.com/en-us/library/ms190640.aspx) (<http://msdn.microsoft.com/en-us/library/ms190640.aspx>).
 10. Redirect all users and applications to the new SQL Server 2014 instance containing the new log shipping primary.

New Hardware for the Primary Only

This option assumes that only the instance for the primary will be brand new and the secondary will be reused and upgraded to SQL Server 2014.

1. Install the new SQL Server 2014 instance that will be used for the primary.
2. Take a full backup of the original primary database and restore it to the new SQL Server 2014 instance using the WITH NORECOVERY option.
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 2014 instance. Use the WITH NORECOVERY option. 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 using the WITH RECOVERY option.
7. Get the new production server fully ready for use. This entails tasks such as:
 - Synchronizing logins and ensuring that all jobs and objects residing outside the database are restored via scripts to make the mirror usable
 - Changing the compatibility mode of the database to 120
 - Configuring administration such as SQL Server Agent jobs
 - Running health checks (e.g., DBCC CHECKDB) to ensure the health of the newly upgraded database
8. Upgrade the secondary to SQL Server 2014.

9. Remove the existing copy and restore jobs on the secondary.
10. Configure log shipping between the SQL Server 2014 instances using the instructions found in the SQL Server 2014 Books Online topic [Configure Log Shipping \(SQL Server\)](http://msdn.microsoft.com/en-us/library/ms190640.aspx) (<http://msdn.microsoft.com/en-us/library/ms190640.aspx>).
11. Redirect all users and applications to the new SQL Server 2014 instance containing the principal.

Additional References for Upgrading Log Shipping

For more information about upgrading log shipping, see the SQL Server 2014 Books Online topic [Upgrade Log Shipping to SQL Server 2014 \(Transact-SQL\)](http://msdn.microsoft.com/en-us/library/cc645954.aspx) (<http://msdn.microsoft.com/en-us/library/cc645954.aspx>).

For an up-to-date collection of additional references for upgrading log shipping, see the [Upgrade to SQL Server 2014](http://msdn.microsoft.com/en-us/library/bb677622(v=sql.120).aspx) page ([http://msdn.microsoft.com/en-us/library/bb677622\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/bb677622(v=sql.120).aspx)).

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 2014.

Feature Changes in SQL Server 2014 Replication

There are no significant changes to replication in SQL Server 2014. Consult Chapter 4 in the previous upgrade guides for those versions to see changes that may apply to your replication topology if you are upgrading from SQL Server 2005. If you are upgrading from SQL Server 2008, consult the [SQL Server 2008 R2 Upgrade Technical Reference Guide](http://download.microsoft.com/download/3/0/D/30DB8D46-8ACF-442A-99A2-0F4CE74AE14D/SQL_Server_2008_R2_Upgrade_Technical_Reference_Guide.docx) (http://download.microsoft.com/download/3/0/D/30DB8D46-8ACF-442A-99A2-0F4CE74AE14D/SQL_Server_2008_R2_Upgrade_Technical_Reference_Guide.docx) in addition to the information provided in this section.

Planning an Upgrade with Replication

Depending on the complexity of your replication topology, your upgrade may be straightforward or it may be complicated. For example, when you have updating subscribers where data can be updated outside of the primary source database, you will have to coordinate multiple outages or restrict which locations can update data for a period of time.

When you are mixing versions of SQL Server and SQL Server 2014 is part of the replication topology, you must follow these rules:

- The Distributor must be running the same version as the Publisher or a later version.
- The Publisher must be running the same version as the Distributor or an earlier version.
- Depending on the type of replication, the Subscriber versions may be restricted. For transactional replication, the Subscriber must be within two versions of the Publisher, which means that the oldest a Subscriber can be with SQL Server 2014 is SQL Server 2008. For merge replication, the Subscriber version can be earlier than or equal to the Publisher.

These rules influence the order in which the instances can be upgraded and where you will incur any potential outages. The key is the Distributor. With a single Distributor, there will be the least downtime. If there are multiple Distributors, you will need to coordinate to have the least impact on end users and applications.

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 is recommended over installing a new instance of SQL Server and reconfiguring the Distributor. 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, provided that two conditions are met:

- The Publisher and Subscriber are down level from the SQL Server 2014 Distributor.
- The Subscriber is running a version of SQL Server supported by the SQL Server 2014 Distributor.

Performing in-place upgrade is preferred when it comes to all aspects of replication because reinitializing a Subscriber can potentially 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.

Deprecated Features and Breaking Changes

In SQL Server 2014, two replication options are being deprecated SQL Server 2005 and SQL Server Compact are longer supported. For a full list of deprecated replication features, including features deprecated in older versions that may affect the upgrade, see [Deprecated Features in SQL Server Replication](http://technet.microsoft.com/en-us/library/ms143550(SQL.120).aspx) ([http://technet.microsoft.com/en-us/library/ms143550\(SQL.120\).aspx](http://technet.microsoft.com/en-us/library/ms143550(SQL.120).aspx)) in SQL Server 2014 Books Online.

There are no breaking changes in SQL Server 2014 with regards to replication. However, if you are upgrading from SQL Server 2005, check the SQL Server 2012 Books Online topic [Breaking Changes in SQL Server Replication](http://technet.microsoft.com/en-us/library/ms143470(SQL.110).aspx) ([http://technet.microsoft.com/en-us/library/ms143470\(SQL.110\).aspx](http://technet.microsoft.com/en-us/library/ms143470(SQL.110).aspx)) for breaking changes made in SQL Server 2008 that would apply.

Behavior Changes

Other than the deprecated features and breaking changes, there are no listed behavior changes to replication in SQL Server 2014 from where things were in SQL Server 2008 R2. If you are coming from SQL Server 2005 or 2008, review the SQL Server 2008 and/or SQL Server 2008 R2 documentation and upgrade technical guides for more information.

In-Place Upgrade

Performing an in-place upgrade to any instance that may be participating in replication may make sense if your hardware and underlying operating system will continue to be supported for a SQL Server 2014 deployment. This section will cover how to perform such an upgrade of a replication topology pointing out specific things for different replication types where applicable.

A nice feature of any replication upgrade is that you do not need to upgrade every component at once. As long as you stick to the rules of replication and upgrade the components in the right order, you can upgrade over time.

1. As components are being upgraded, verify that you will have a valid topology. If not, you may need to adjust the plans. For example, if you upgrade the Distributor but you have SQL Server 2000 Subscribers, you may need to wait to upgrade or get them to SQL Server 2008 so that they are at the minimum supported level for replication. (For information about the minimum supported levels, see the “Planning an Upgrade with Replication” section earlier in this chapter.)

2. Generate scripts for the entire existing replication topology and store them in a safe place. For documentation on how to do this, see the SQL Server 2014 Books Online topic [Generate SQL Script \(Replication Objects\)](http://msdn.microsoft.com/en-us/library/ms188503.aspx) (<http://msdn.microsoft.com/en-us/library/ms188503.aspx>). In a worst case scenario, these scripts will enable you to reconfigure replication in the exact configuration it is now.

3. Stop all application and data traffic to the Publisher that will be affected during the upgrade.

Ensure that all existing transactions marked for publication have been moved to the Subscribers. If using transactional replication, run `sp_repltrans` on the Publisher to get the outstanding transactions marked for publication. If there are transactions still pending, the stored procedure will return rows. Once the result set is empty, the upgrade can begin. Check over the span of a few minutes just to be safe. There are other places you can look to see the status of replication, such as in Replication Monitor. Each form of replication at the Publisher can show you whether things are moving or not. In SSMS, expand Replication then Local Publications. For each publication, right-click and then select one of the status options.

4. Before any major components are upgraded (the first of which will be the instance containing the Distributor), disable any SQL Server Agent jobs related to replication, including any push subscriptions at the various Publishers using the Distributor that will be upgraded and pull subscriptions at the Subscriber. This will ensure that no data propagation will occur during the upgrade process.
5. Upgrade the instance containing the Distributor to SQL Server 2014.
6. At this point, assuming the Publisher and Subscribers are valid for a SQL Server 2014-based topology and you do not want to upgrade those yet, you can stop for now. If this is the case:
 - a. Ensure that SQL Server Agent is started on the Distributor.
 - b. Enable all of the replication jobs.
 - c. Verify that replication is now working again.
7. When you are ready to start upgrading again, the next component to upgrade would be the Publisher (assuming each Subscriber's version of SQL Server is compatible with SQL Server 2014 replication). To upgrade the Publisher:

- a. If not done already, stop all traffic and end any connections to the Publisher.
 - b. Following the instructions in Step 4, verify that all transactions have been replicated.
 - c. If necessary, disable all replication SQL Server Agent jobs if this is being done at a later time than the Distributor upgrade.
 - d. Upgrade the Publisher to SQL Server 2014.
 - e. Ensure that SQL Server Agent is started.
 - f. 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.
 - g. Set the database compatibility level of the upgraded databases to 120.
 - h. If the Subscriber will not be upgraded at this time, enable all the SQL Server Agent jobs related to the Publisher and Distributor.
8. Upgrade the Subscriber(s). If there are multiple Subscribers, consider doing a phased upgrade (i.e., upgrade selected subscribers in different outages) to minimize the impact to end users. To upgrade a Subscriber, follow these steps:
- a. If not done already, disable any push or pull job associated with the Subscriber's subscriptions. You do not need to stop all traffic or kill the connections to the Publisher during an upgrade of the Subscriber. This will ensure that the Subscriber cannot be updated during the upgrade process. However, be aware that the Publisher will not be able to flush transactions until the Subscriber is back online. It may also affect the Publisher's ability to back up the transaction log.
 - b. Upgrade the instance containing the Subscriber to SQL Server 2014.
 - c. Run DBCC CHECKDB against all upgraded databases.
 - d. Set the database compatibility level of the upgraded databases to 120.
 - e. Ensure that SQL Server Agent is started.
 - f. 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 (if they were somehow disabled).

- g. Verify that the replication is working properly and now sending what it needs to (e.g., a transaction in transactional replication). The easiest way to do this is to manually kick off the SQL Server Agent jobs involved in snapshot replication and view the status using some of the methods shown earlier in Step 4. They should reflect a successful execution.
9. Generate new scripts for the upgraded replication architecture.

Side-by-Side Upgrade to a New Server or Cluster

Using new hardware and brand new SQL Server 2014 instances may complicate an upgrade with replication since you will need to either re-point components (such as a Publisher to a new Distributor) or reinitialize subscriptions. The upgrade process is similar to, but slightly different from, an in-place upgrade.

1. As components are being upgraded, verify that you will have a valid topology. If not, you may need to adjust the plans. For example, if you upgrade the Distributor but you have SQL Server 2000 Subscribers, you may need to wait to upgrade or get them to SQL Server 2008 so that they are at the minimum supported level for replication. (For information about the minimum supported levels, see the “Planning an Upgrade with Replication” section earlier in this chapter.)
2. Generate scripts for the entire existing replication topology and store them in a safe place. For documentation on how to do this, see the SQL Server 2014 Books Online topic [Generate SQL Script \(Replication Objects\)](http://msdn.microsoft.com/en-us/library/ms188503.aspx) (<http://msdn.microsoft.com/en-us/library/ms188503.aspx>). In a worst case scenario, these scripts will enable you to reconfigure replication in the exact configuration it is now.
3. Install and configure the hardware and SQL Server 2014 instance(s) that will participate in the replication topology.
4. Update the scripts generated in Step 2 to SQL Server 2014 and make sure that the instance names and databases are updated to reflect their new locations. Before updating the scripts, make copies of the originals so that the old environment can be restored if necessary.
5. At the minimum, ensure that the existing Publisher is at a version that can participate in the replication topology when SQL Server 2014 is added since the Distributor must be running SQL Server 2014.

6. Stop all application and data traffic to the Publisher that will be affected during the upgrade to ensure that the data will not be updated during the cutover.
7. Run the appropriate upgraded replication script on the new Distributor to create the publication and distribution with your settings. At this point, the old replication topology is still being used and you have not incurred any downtime.
8. If the Distributor is the only component being upgraded, to do the cutover, stop all traffic and kill any connections to the Publisher to ensure that no one tries to access it during the switch.
9. To point existing Publishers and Subscribers to the new Distributor, you must drop the publications and subscriptions, and then recreate them pointing to the new Distributor. If this completes successfully, replication works, and you will not be upgrading any other components, your upgrade is complete.
10. If you are also moving the Publisher to a new SQL Server 2014 instance:
 - a. Drop the existing publication(s) and subscription(s).
 - b. Re-create the publication on the new Publisher and point it to the SQL Server 2014 Distributor.
 - c. Recreate the subscription pointing to the new publication.
 - d. Verify that replication is working.
11. If you are also moving the Subscriber(s) to a new SQL Server 2014 instance:
 - a. To point at the new Distributor, you should have already dropped the subscription. If not, do so now.
 - b. Re-create the subscription on the new Subscriber.
 - c. Re-create the subscription pointing to the new publication.
 - d. Verify that replication is working.
12. Configure administration such as backup jobs for the databases.
13. Generate new scripts for the upgraded replication architecture.
14. 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.

Additional Information for Upgrading Replicated Databases

See the following SQL Server 2014 Books Online topics for more information about upgrading replicated databases:

- [Deprecated Features in SQL Server Replication](http://msdn.microsoft.com/en-us/library/ms143550(v=sql.120).aspx)
[http://msdn.microsoft.com/en-us/library/ms143550\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143550(v=sql.120).aspx)
- [Breaking Changes in SQL Server Replication](http://msdn.microsoft.com/en-us/library/ms143470(v=sql.120).aspx)
[http://msdn.microsoft.com/en-us/library/ms143470\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143470(v=sql.120).aspx)
- [Upgrade Replicated Databases](http://msdn.microsoft.com/en-us/library/ms143699(v=sql.120).aspx)
[http://msdn.microsoft.com/en-us/library/ms143699\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143699(v=sql.120).aspx)

For an up-to-date collection of additional references for upgrading replication, see [Upgrade to SQL Server 2014](http://technet.microsoft.com/en-us/library/bb677622.aspx) (<http://technet.microsoft.com/en-us/library/bb677622.aspx>).

Conclusion

While it is not possible to completely avoid downtime during certain points of an upgrade to SQL Server 2014, minimizing downtime is achievable by following the advice presented in this chapter. Preparation and testing are the ultimate defense against failures and extended outages. Each high availability feature currently configured on your existing instance will have different considerations and steps that you will need to account for in your overall upgrade plan.

Additional References

For an up-to-date collection of additional SQL Server 2014 high-availability references, see [Upgrade to SQL Server 2014](http://technet.microsoft.com/en-us/library/bb677622.aspx) (<http://technet.microsoft.com/en-us/library/bb677622.aspx>).

Also see the following links:

- [SQL Server 2014 Web Site](http://www.microsoft.com/sqlserver/en/us/default.aspx)
(<http://www.microsoft.com/sqlserver/en/us/default.aspx>)
- [Books Online for SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms130214(v=sql.120).aspx)
([http://msdn.microsoft.com/en-us/library/ms130214\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms130214(v=sql.120).aspx))
- [SQL Server Developer Center](http://msdn.microsoft.com/en-us/sqlserver)
(<http://msdn.microsoft.com/en-us/sqlserver>)
- [SQL Server TechNet Resources](http://technet.microsoft.com/en-us/sqlserver)
(<http://technet.microsoft.com/en-us/sqlserver>)

Chapter 5: Database Security

Introduction

The number of security factors affecting the upgrade of the SQL Server relational engine from SQL Server 2005, SQL Server 2008, SQL Server 2008 R2, or SQL Server 2012 to SQL Server 2014 is fairly small. 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 (DBA). 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 2014 has been enhanced and strengthened. But DBAs 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://msdn.microsoft.com/en-us/library/ms144228(v=SQL.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms144228\(v=SQL.120\).aspx](http://msdn.microsoft.com/en-us/library/ms144228(v=SQL.120).aspx)) in SQL Server 2014 Books Online. Let's look at how the new and enhanced security features in SQL Server 2014 could affect your upgrade planning and process.

Table 1 provides a summary of the new security features in SQL Server 2014 and how they might affect the upgrade process from SQL Server 2005, SQL Server 2008, SQL Server 2008 R2, or SQL Server 2012.

Table 1: SQL Server 2014 Security Features

New SQL Server 2014 Security Features	Benefit	When Available	Affects the Upgrade Process
New CONNECT ANY DATABASE permission	Allows access to current and future databases on the server, but only provides the ability to connect to the database	Immediately after upgrade	No
New IMPERSONATE ANY LOGIN permission	Either allows (or explicitly denies) the ability for a login to impersonate other logins	Immediately after upgrade	No

New SQL Server 2014 Security Features	Benefit	When Available	Affects the Upgrade Process
New SELECT ALL USERS SECURABLES permission	Allows a login to view data in all databases that the login can connect to	Immediately after upgrade	No

As you can see from this table, the new security features will not immediately affect your upgrade. You can take advantage of these new features after your upgrade, but they will not block or impede the upgrade process.

New Configuration Tools

If you upgrade from SQL Server 2005, 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 R2 onward 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://msdn.microsoft.com/en-us/library/ms174212(v=SQL.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms174212\(v=SQL.120\).aspx](http://msdn.microsoft.com/en-us/library/ms174212(v=SQL.120).aspx)) in SQL Server 2014 Books Online.

You use the Policy-Based Management capability introduced in SQL Server 2012 to help set Database Engine, Analysis Services, and Reporting Services configurations. You can import the best practice policies that SQL Server 2014 provides to evaluate the configurations for your instances, instance objects, databases, and database objects.

Note: You can also use `sp_configure` to set Database Engine features.

Configuring Services and Connections

SQL Server Configuration Manager lets you configure SQL Server 2014 services and (when relevant) remote connections. If you use an in-place upgrade, SQL Server 2014 Setup preserves your source SQL Server instance's service configurations.

Note: There is an exception if you upgrade from SQL Server 2005 because a new full-text search service was introduced in SQL Server 2008. Even for an in-place upgrade, the service settings for SQL Server 2005 full-text search will not be preserved.

SQL Server 2014 supports Managed Service Accounts and Virtual Accounts to be used for SQL Server services. Managed Service Accounts and Virtual Accounts provide the isolation of their own accounts, while eliminating the need for an administrator to manually administer the Service Principal Name (SPN) and credentials for these accounts. These changes make the long-term management of service account users, passwords, and SPNs much easier. For more information on these types of accounts see [Service Accounts Step-by-Step Guide](http://technet.microsoft.com/en-us/library/dd548356(W.S.10).aspx) ([http://technet.microsoft.com/en-us/library/dd548356\(W.S.10\).aspx](http://technet.microsoft.com/en-us/library/dd548356(W.S.10).aspx)).

As a default, SQL Server 2014 uses Virtual Accounts for new installations for all services, except the SQL Server Browser and SQL Server VSS Writer services, where LOCAL SERVICE and LOCAL SYSTEM are the default accounts.

If you use the in-place upgrade method, you can use SQL Server Configuration Manager to change to the new available Managed Service Accounts or Virtual Accounts after the upgrade.

If you use a side-by-side upgrade method, you might need to adjust the default service settings based on your needs. To change the service accounts, password, service startup type, or other properties of any SQL Server–related service after installation, 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://msdn.microsoft.com/en-us/library/ms174212(v=SQL.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms174212\(v=SQL.120\).aspx](http://msdn.microsoft.com/en-us/library/ms174212(v=SQL.120).aspx)) in SQL Server 2014 Books Online.

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 2014. Use the principle that if a service is not needed, it should be disabled. You can also enable and disable remote connections for many of the services.

For details about service accounts and services, see [Configure Windows Service Accounts and Permissions](http://msdn.microsoft.com/en-us/library/ms143504(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms143504\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143504(v=sql.120).aspx)) in SQL Server 2014 Books Online.

Service Account Security

Microsoft SQL Server 2014 enables a per-service SID for each of its services, except the SQL Server Browser service, to provide service isolation and in-depth defense. The per-service SID is derived from the service name and is unique to that service. Service isolation enables access to specific objects without the need to run a high-privilege account or weaken the security protection of the object. By using an access control entry (ACE) that contains a service SID, a SQL Server service can restrict access to its resources. For all SQL Server services, except for the SQL Server Browser and Analysis Services, SQL Server configures the permissions for the per-service account directly, so changing the service account can be done without having to repeat the resource ACL process. For the two services not using per-service SIDs, SQL Server Setup creates local Windows groups and configures the required permissions directly on the created groups. For more information, see [Configure Windows Service Accounts and Permissions](http://msdn.microsoft.com/en-us/library/ms143504(v=SQL.120).aspx) (http://msdn.microsoft.com/en-us/library/ms143504(v=SQL.120).aspx) in SQL Server 2014 Books Online.

This is a big difference compared to older SQL Server installations. Earlier versions created local Windows groups for all services and used per-service SIDs in only some configurations. For more information, see [Configure Windows Service Accounts and Permissions](http://msdn.microsoft.com/en-us/library/ms143504(v=sql.120).aspx) (http://msdn.microsoft.com/en-us/library/ms143504(v=sql.120).aspx).

In case of an in-place upgrade from SQL Server 2008, SQL Server 2008 R2, or SQL Server 2012, SQL Server Setup will preserve the ACEs for the per-service SIDs and the local Windows groups used by SQL Server.

In case of an upgrade from SQL Server 2005, SQL Server Setup will change the permissions to use per-service SIDs for the database engine and its file folders and registry keys. The local Windows groups are preserved but renamed.

In both cases, all permissions are preserved and no special actions are needed for a successful upgrade.

As a best practice, use Managed Service Accounts or Virtual Accounts if possible and grant as few additional privileges as possible. In cases where it is not possible to use a Managed Service Account or Virtual Account, use a separate low-privilege user account or domain account for each service and don't give additional privileges directly to it. Instead apply privileges on security groups or per-service SIDs accordingly. For more information, see [Configure Windows Service Accounts and Permissions](http://msdn.microsoft.com/en-us/library/ms143504(v=SQL.120).aspx) (http://msdn.microsoft.com/en-us/library/ms143504(v=SQL.120).aspx).

Configuring Features

Through the new Policy-Based Management capability introduced 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 2014 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/SQL Mail
- Native XML Web services (if HTTP endpoints are defined)
- OLE automation
- Service Broker
- `xp_cmdshell`
- Web Assistant

You can find out-of-the-box security best practices policies, including Surface Area Configuration, in the SQL Server 2014 installation path in the `\120\Tools\Policies\DatabaseEngine\1033` folder. For more information about Policy-Based Management, see [Administer Servers by Using Policy-Based Management](http://msdn.microsoft.com/en-us/library/bb510667(v=SQL.120).aspx) ([http://msdn.microsoft.com/en-us/library/bb510667\(v=SQL.120\).aspx](http://msdn.microsoft.com/en-us/library/bb510667(v=SQL.120).aspx)).

SQL Server Audit

SQL Server Audit is a feature introduced in SQL Server 2008. It lets you create customized audits of Database Engine security events that can write to a file, the Windows Security event log, or the Windows Application event log. It is based on Extended Events and provides a low overhead, scalable, reliable, and secure way to perform auditing. SQL Server 2014 Enterprise Edition supports all the features in SQL Server Audit, whereas all other editions support only the core features in SQL Server Audit. The core features support everything that is also available through SQL Trace, making it easy to move from SQL Trace to SQL Server Audit in all editions of SQL Server. For more information on SQL Server Audit, see [SQL Server Audit \(Database Engine\)](http://msdn.microsoft.com/en-us/library/cc280386(v=SQL.120).aspx) ([http://msdn.microsoft.com/en-us/library/cc280386\(v=SQL.120\).aspx](http://msdn.microsoft.com/en-us/library/cc280386(v=SQL.120).aspx)) in SQL Server 2014 Books Online.

If you are using SQL Trace for auditing purposes, you should consider changing to SQL Server Audit after the update. Beside the fact that SQL Server Audit is more suitable and reliable for security auditing and easier to manage than SQL Trace, SQL Trace is deprecated and will be removed in a future version of SQL Server. By changing to SQL Server Audit now, you can take advantage of its features and make the upgrade to future versions of SQL Server easier.

Preparing to Upgrade

After you become familiar with the new features in SQL Server 2014, you need to prepare for your upgrade plan and process. Specifically, you need to understand which features have been deprecated or discontinued in SQL Server 2014 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 2014. Although they will continue to operate in SQL Server 2014, they will be removed in a future version of SQL Server. These deprecated features have no immediate effect on your upgrade to SQL Server 2014, but they will affect your upgrade to a later version. For comprehensive information about these features, see [Deprecated Database Engine Features in SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms143729(v=SQL.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms143729\(v=SQL.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143729(v=SQL.120).aspx)) in SQL Server 2014 Books Online.

The deprecated security features consist primarily of system stored procedures that have been replaced by Transact-SQL (T-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 to using catalog views such as sys.objects.

Table 2 lists the deprecated security features in SQL Server 2014 and their replacements.

Table 2: 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 ALTER AUTHORIZATION
sp_defaultdb sp_defaultlanguage	ALTER LOGIN
sp_denylogin sp_grantlogin sp_revokellogin	ALTER LOGIN DISABLE CREATE LOGIN DROP LOGIN
USER_ID	DATABASE_PRINCIPAL_ID
xp_grantlogin xp_revokellogin 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 starting in SQL Server 2008. For more information, see Permissions of Fixed Server Roles (Database Engine) (http://msdn.microsoft.com/en-us/library/ms175892(v=sql.105).aspx) in SQL Server 2008 R2 Books Online.

Deprecated Feature	Replacement
sp_addremotelogin sp_addserver sp_dropremotelogin sp_helpremotelogin sp_remoteoption	Use linked servers instead of remote servers; sp_addserver can only be used with the local option.
sp_control_dbmasterkey_password	A master key must exist and password must be correct
Encryption using RC4 or DESX is deprecated	Use another encryption algorithm such as AES.
GRANT ALL DENY ALL REVOKE ALL	GRANT, DENY, and REVOKE specific permissions
PERMISSIONS intrinsic function	Query sys.fn_my_permissions
SETUSER	EXECUTE AS

To fully understand these security upgrade issues, make sure you review [SQL Server Database Engine Backward Compatibility](http://msdn.microsoft.com/en-us/library/ms143532(v=SQL.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms143532\(v=SQL.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143532(v=SQL.120).aspx)) in SQL Server 2014 Books Online.

Discontinued Features

A number of security features are discontinued in SQL Server 2012, but no new security features were discontinued in SQL Server 2014. If upgrading from SQL Server 2012, there are no changes. However, if you are upgrading from SQL Server 2005, SQL Server 2008, or SQL Server 2008 R2, 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 3 lists the security-related features that were discontinued in SQL Server 2012.

Table 3: Discontinued Security-Related Features in SQL Server 2012

Discontinued Feature	Explanation/Replacement
Remote server	The ability for users to create new remote servers by using sp_addserver is discontinued; sp_addserver with the 'local' option remains available. Remote servers preserved during the upgrade or created by replication can be used.
sp_dropalias	Replace aliases with a combination of user accounts and database roles.
The version parameter of PWDCOMPARE representing a value from a login earlier than SQL Server 2000	None
sys.database_principal_aliases	Replace aliases with a combination of user accounts and database roles.

For comprehensive information about these features, see [Discontinued Database Engine Functionality in SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms144262(v=SQL.120).aspx) (http://msdn.microsoft.com/en-us/library/ms144262(v=SQL.120).aspx) in SQL Server 2014 Books Online.

Note: If you are upgrading from SQL Server 2005, there might be features that were discontinued in SQL Server 2008 and SQL Server 2008 R2 that could affect your upgrade process. For a full list, see [Discontinued Database Engine Functionality in SQL Server 2008 R2](http://msdn.microsoft.com/en-us/library/ms144262(v=SQL.105).aspx) (http://msdn.microsoft.com/en-us/library/ms144262(v=SQL.105).aspx) in SQL Server 2008 R2 Books Online.

Breaking Changes

Table 4 lists the security-related breaking changes in the database engine. These changes happened in SQL Server 2012, and there are no new issues in SQL Server 2014. As you can see, the list is quite small, but you should check if they apply to the system that you are about to upgrade and fix them in advance.

Table 4: Security-Related Breaking Changes

Blocking Issue	Solution
sp_setapprole, sp_unsetapprole, and EXECUTE AS	The cookie OUTPUT parameter is currently documented as varbinary(8000), which is the correct maximum length. However, the current implementation returns varbinary(50) for application roles and varbinary(100) for EXECUTE AS. Applications should continue to reserve varbinary(8000) so that the application continues to operate correctly if the cookie return size increases in a future release.
sys.fn_get_audit_file	Two additional columns (user_defined_event_id and user_defined_information) have been added to support user-defined audit events. Applications that do not select columns by name might return more columns than expected. Either select columns by name, or adjust the application to accept these additional columns.

For comprehensive information about these changes, see [Breaking Changes to Database Engine Features in SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms143179(v=SQL.120).aspx) (http://msdn.microsoft.com/en-us/library/ms143179(v=SQL.120).aspx) in SQL Server 2014 Books Online.

Note: If you are upgrading from SQL Server 2005, there might be breaking changes in SQL Server 2008 and SQL Server 2008 R2 that could affect your upgrade process. For a full list, see [Breaking Changes to Database Engine Features in SQL Server 2008 R2](http://msdn.microsoft.com/en-us/library/ms143179(v=SQL.105).aspx) (http://msdn.microsoft.com/en-us/library/ms143179(v=SQL.105).aspx) in SQL Server 2008 R2 Books Online.

Behavior Changes

There are no security-related behavior changes in SQL Server 2014. For comprehensive information about other changes, see [Behavior Changes to Database Engine Features in SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms143359(v=SQL.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms143359\(v=SQL.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143359(v=SQL.120).aspx)) in SQL Server 2014 Books Online.

Note: If you are upgrading from SQL Server 2005, there might be behavior changes in SQL Server 2008 and SQL Server 2008 R2 that could affect your upgrade process. For a full list, see [Behavior Changes to Database Engine Features in SQL Server 2008 R2](http://msdn.microsoft.com/en-us/library/ms143359(v=SQL.105).aspx) ([http://msdn.microsoft.com/en-us/library/ms143359\(v=SQL.105\).aspx](http://msdn.microsoft.com/en-us/library/ms143359(v=SQL.105).aspx)) in SQL Server 2008 R2 Books Online.

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 2014. 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
 - Encryption algorithms used in databases

Since SQL Server 2008, SQL Server has enhanced SQL Server login passwords. DBAs 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.

2. 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. Like you do for other backups, apply the same security considerations to the backup you take before upgrading: 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.
3. Review and resolve issues identified by SQL Server 2014 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 the Best Practices Analyzer for SQL Server.
4. 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?
5. Determine service account security. Determine the service accounts used for your SQL Server 2014 installation. As mentioned earlier in the "Service Account Security" section, you do not need to grant local administrator rights to SQL Server 2014 service accounts. Before you upgrade, you need to determine the rights those accounts will have or let SQL Server Setup do that for you.
6. 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 the "In-Place Upgrade" and "Side-by-Side Upgrade" sections later in this chapter.
7. Create, document, and test the upgrade plan. Good planning is the best method for preventing errors. You should not only document your upgrade plan but also 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 2014

Here are the key security considerations for an in-place or side-by-side upgrade.

In-Place Upgrade

In SQL Server 2014, services and configuration settings are off by default unless required. During an in-place upgrade, the SQL Server 2014 Setup program will maintain the SQL Server 2014 services 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 instances, 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 2014 Setup program can start the new SQL Server 2014 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 2014 on a new server or alongside the legacy instance on your current server. In this case, you should ensure that services and configuration options are set in such a way as to guarantee your successful upgrade. For example, the following features are off by default in a new SQL Server 2014 installation to help reduce the attack surface of your new installation: ad hoc distributed queries, OLE automation, 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 one, you must choose a transfer method such as backup/restore, detach/attach, BCP, Data Transformation Services (DTS), or Copy Database Wizard. In the case of the first two methods, you will be copying files over some distance, so 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 2014 destination instance. For more information about using this wizard, see [Use the Copy Database Wizard](http://msdn.microsoft.com/en-us/library/ms188664(v=SQL.120).aspx) (http://msdn.microsoft.com/en-us/library/ms188664(v=SQL.120).aspx) in SQL Server 2014 Books Online.

Note: If you are using a side-by-side upgrade method, you need to configure the SQL Server 2014 services to match the needs of your SQL Server installation. In an in-place upgrade, SQL Server 2014 Setup will preserve the services settings of the SQL Server instance you are upgrading, except for the new service for full-text search if you are upgrading from SQL Server 2005. Even for an in-place upgrade, the service setting for SQL Server 2005 full-text will not be preserved.

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 2014 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 2014 instance, and ensure that they are the minimum 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.
6. Change keys. Change the Service Master Key and Database Master Keys encryption from 3DES to AES.

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 [Microsoft Baseline Security Analyzer](http://technet.microsoft.com/en-us/security/cc184924.aspx) (<http://technet.microsoft.com/en-us/security/cc184924.aspx>).

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 2014 system.

Conclusion

SQL Server 2014 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 2005, SQL Server 2008, SQL Server 2008 R2, or SQL Server 2012 to SQL Server 2014. 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 following links:

- [SQL Server 2014 Web Site](http://www.microsoft.com/sqlserver/en/us/default.aspx)
(<http://www.microsoft.com/sqlserver/en/us/default.aspx>)
- [Books Online for SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms130214(v=SQL.120).aspx)
([http://msdn.microsoft.com/en-us/library/ms130214\(v=SQL.120\).aspx](http://msdn.microsoft.com/en-us/library/ms130214(v=SQL.120).aspx))
- [SQL Server Developer Center](http://msdn.microsoft.com/en-us/sqlserver)
(<http://msdn.microsoft.com/en-us/sqlserver>)
- [SQL Server TechNet Resources](http://technet.microsoft.com/en-us/sqlserver)
(<http://technet.microsoft.com/en-us/sqlserver>)

Chapter 6: Full-Text Search

Introduction

SQL Server 2014 provides a 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. 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), work with the DLL syntax, and 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 related to full-text search that you should consider before, during, and after the upgrade process from previous SQL Server versions to SQL Server 2014.

For more information about the full-text architecture, see [Full-Text Search \(SQL Server\)](http://msdn.microsoft.com/en-us/library/ms142571.aspx) (<http://msdn.microsoft.com/en-us/library/ms142571.aspx>) in SQL Server 2014 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 2008 R2 or SQL Server 2012 to SQL Server 2014.

For a complete list of deprecated features, discontinued functionality, breaking changes, and behavior changes to full-text search in SQL Server 2014, see [Full-Text](#)

[Search Backward Compatibility](http://msdn.microsoft.com/en-us/library/ms143544(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms143544\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143544(v=sql.120).aspx)) in SQL Server 2014 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. 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://msdn.microsoft.com/en-us/library/bb510662(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/bb510662\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/bb510662(v=sql.120).aspx)) in SQL Server 2014 Books Online.

The most relevant deprecated features are as follows:

- `CREATE FULLTEXT CATALOG IN PATH`. Using the `CREATE FULLTEXT CATALOG` statement with the `IN PATH` clause shows that an error was encountered. This clause has no effect in this version of SQL Server. Occurs once per use.
- `DATABASEPROPERTYEX('IsFullTextEnabled')`. This statement references the `DATABASEPROPERTYEX IsFullTextEnabled` property. The value of this property has no effect. User databases are always enabled for full-text search. Do not use this property. Occurs once per compilation.
- The catalog views `fulltext_catalogs`, `fulltext_catalogs.data_space_id`, `fulltext_catalogs.file_id`, and `fulltext_catalogs.path`.
- The properties `FULLTEXTCATALOGPROPERTY('LogSize')`, `FULLTEXTCATALOGPROPERTY('PopulateStatus')`, `FULLTEXTSERVICEPROPERTY('ConnectTimeout')`, `FULLTEXTSERVICEPROPERTY('DataTimeout')`, `FULLTEXTSERVICEPROPERTY('ResourceUsage')`. Avoid using these properties.

For a completed list of deprecated full-text search features, see [Deprecated Full-Text Search Features in SQL Server 2014](http://msdn.microsoft.com/en-us/library/bb510662(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/bb510662\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/bb510662(v=sql.120).aspx)) in SQL Server 2014 Books Online.

Breaking Changes

No breaking changes related to full-text search in SQL Server 2014 have been documented at the time of this writing. Before upgrading to SQL Server 2014, check the [Breaking Changes to Full-Text Search](http://msdn.microsoft.com/en-us/library/bb510662(v=sql.120).aspx) web page ([http://msdn.microsoft.com/en-us/library/bb510662\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/bb510662(v=sql.120).aspx)) in SQL Server 2014 Books Online.

us/library/ms143709(v=sql.120).aspx) to see whether any breaking changes have been documented since that time.

The [Breaking Changes to Full-Text Search](http://msdn.microsoft.com/en-us/library/ms143709(v=sql.120).aspx) web page (http://msdn.microsoft.com/en-us/library/ms143709(v=sql.120).aspx) also lists the breaking changes related to full-text search in previous versions of SQL Server.

Behavior Changes

No behavior changes related to full-text search in SQL Server 2014 have been documented at the time of this writing. Before upgrading to SQL Server 2014, check the [Behavior Changes to Full-Text Search](http://msdn.microsoft.com/en-us/library/ms143709(v=sql.120).aspx) web page (http://msdn.microsoft.com/en-us/library/ms143709(v=sql.120).aspx) to see whether any behavior changes have been documented since that time.

The [Behavior Changes to Full-Text Search](http://msdn.microsoft.com/en-us/library/ms143709(v=sql.120).aspx) web page (http://msdn.microsoft.com/en-us/library/ms143709(v=sql.120).aspx) also lists the behavior changes related to full-text search in previous versions of SQL Server. One of the most important behavior changes in SQL Server 2012 is the introduction of improved word breakers, stemmers, and filters for some languages. The improved word breakers might return different results than the older components. The recommendation is to repopulate existing full-text indexes.

Running Upgrade Advisor

To obtain a report that identifies potential issues before you start an upgrade, you should run SQL Server 2014 Upgrade Advisor to analyze the relational database that you want to upgrade. For information about how to install and run this tool, see Chapter 1, "Upgrade Planning and Deployment."

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 2008/2008 R2/2012 versions of the SQL Server Best Practices Analyzer (BPA) to prepare for an upgrade.

Preparing for a Possible Rollback

Before you start an upgrade of a database that has full-text search enabled, you need to make sure that sufficient space exists in the database for the upgrade to complete successfully and make sure that you can roll back a failed upgrade if necessary.

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 2008/2008 R2/2012 instance. Therefore, planning for a failed upgrade process is important.

A side-by-side upgrade of an existing full-text enabled database to SQL Server 2014 should not encounter the same kinds of problems that can affect an in-place upgrade. However, you should follow similar steps to ensure 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 2008/2008 R2/2012 and restore the installation to its state before the upgrade process began. To make sure that all the data and configuration files 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:

- Review the requirements to determine whether your hardware and software can support SQL Server 2014. For more information, see [Hardware and Software Requirements for Installing SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms143506(v=sql.120).aspx) (http://msdn.microsoft.com/en-us/library/ms143506(v=sql.120).aspx).
- Use System Configuration Checker (SCC) to scan the server for any conditions that might prevent a successful installation of SQL Server 2014. For more information, see [Check Parameters for the System Configuration Checker](http://msdn.microsoft.com/en-us/library/ms143753(v=sql.120).aspx) (http://msdn.microsoft.com/en-us/library/ms143753(v=sql.120).aspx).
- 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(v=sql.120).aspx) (http://msdn.microsoft.com/en-us/library/ms144228(v=sql.120).aspx).
- Run Upgrade Advisor on the server to determine whether there are any issues that might prevent you from successfully upgrading. For more information, see [Use Upgrade Advisor to Prepare for Upgrades](http://msdn.microsoft.com/en-us/library/ms144256(v=sql.120).aspx) (http://msdn.microsoft.com/en-us/library/ms144256(v=sql.120).aspx).
- Make sure that the filegroup associated with the base table of a full-text index has sufficient space to accommodate the additional space that is required by SQL Server 2014 full-text indexes.

Upgrading a Full-Text-Enabled Database

You can choose between two options when you upgrade a full-text enabled database to SQL Server 2014: in-place upgrade or side-by-side upgrade.

In-Place Upgrade

For an in-place upgrade, an instance of SQL Server 2014 is set up side-by-side with the old version of SQL Server 2008/2008 R2/2012, and data is moved to the new version. 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 uses its own set of word breakers, stemmers, and filters instead of relying on the operating system version of these components. These components are easier to register and configure at a per-instance level. For more information, see [Configure and Manage Word Breakers and Stemmers for Search](http://msdn.microsoft.com/en-us/library/ms142509(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms142509\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms142509(v=sql.120).aspx)) and [Configure and Manage Filters for Search](http://msdn.microsoft.com/en-us/library/ms142499(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms142499\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms142499(v=sql.120).aspx)) in SQL Server 2014 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 introduced 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 [Set the Service Account for the Full-text Filter Daemon Launcher](http://msdn.microsoft.com/en-us/library/ms345189(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms345189\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms345189(v=sql.120).aspx)) in SQL Server 2014 Books Online.

In SQL Server 2005 and earlier versions, each full-text index is located in a full-text catalog that belongs to a filegroup, has a physical path, and is treated as a database file. In SQL Server 2014, 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 filegroup 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 filegroup if the root path exists. If the old full-text catalog path is invalid, the upgrade keeps the full-text index in the same filegroup as the base table or, for a partitioned table, in the primary filegroup.

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, follow 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.

Side-by-Side Upgrade

For a side-by-side upgrade, an instance of SQL Server 2014 is set up side-by-side with the old version of SQL Server on the same server or another server, and you move data to the new version.

Full-Text Upgrade Option

SQL Server 2014 provides an instance option, Full-Text Upgrade, which lets you control how SQL Server manages full-text indexes in side-by-side upgrade scenarios. You can set the Full-Text Upgrade option to one of three possible values:

- **Import.** This is the default option after the setup of a SQL Server 2014 instance. It works only for SQL Server 2005 databases. If this option is enabled, SQL Server 2014 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 index structures. Import is the fastest option for an upgrade. But if you have a specific set of the improved word breakers and stemmers found in SQL Server 2012 and SQL Server 2014, Microsoft cannot guarantee that your queries will return the same results as the SQL Server 2005 version. (See the "Semantic Consistency" section later in this topic.) At import time, this option does not use the improved word breakers and stemmers found in SQL Server 2012 and SQL Server 2014.

- **Rebuild.** With this option, SQL Server 2014 will rebuild all full-text indexes (triggering a full population) and use the 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 2014 deletes the existing full-text catalogs. In addition, 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 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 2014, the data in these indexes is obtained by using the old word breakers and stemmers. But when you query the new SQL Server 2014 full-text index, the full-text engine will use the improved word breakers so the results of the queries might change. This behavior is known as semantic consistency.

SQL Server 2014 does not improve all word breakers. The following word breakers have not changed from the earlier version so semantic inconsistency does not apply to them:

- Chinese (Hong Kong SAR, PRC)
- Chinese (Macau SAR)
- Chinese (Singapore)
- Korean
- Simplified Chinese
- Thai
- Traditional Chinese

For more information about semantic consistency, review the "Ensuring Consistent Query Results after Importing a SQL Server 2005 Full-Text Index" section in [Upgrade](#)

[Full-Text Search](http://msdn.microsoft.com/en-us/library/ms142490(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms142490\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms142490(v=sql.120).aspx)) in SQL Server 2014 Books Online.

Considerations for Choosing a Full-Text Upgrade Option

When choosing the appropriate strategy for an upgrade of full-text search, consider the following questions:

- **How do you use word breakers?** The SQL Server 2014 full-text search service includes the improved 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 2014 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 improved word breakers by rebuilding your full-text catalogs.
 - If you care about recall accuracy and you use one of the improved word breakers, 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 the 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 an 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/Restore Method

The first option for doing a side-by-side upgrade is to perform a restore of a SQL Server 2005/2008/2008 R2/2012 full-text-enabled database on a SQL Server 2014 instance.

When you restore the database on SQL Server 2014, 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 2014 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/Attach Method

When you attach a full-text-enabled database to SQL Server 2014, catalog files are attached from their previous locations together with the other database files. If SQL Server 2014 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 2014 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 2014 even if the full-text upgrade option is configured as `Import`.

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:

4. Install the third-party filter on the SQL Server 2014 server.
8. Use the `sp_fulltext_service` stored procedure to set the service property, `load_os_resources`, for the third-party filter.
9. Turn off the `Verify_signature` option.
10. 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 stopwords in SQL Server 2008 and later versions. When you upgrade a database from a previous release of SQL Server to SQL Server 2014, the noise-word files are no longer used in SQL Server 2014. However, the old noise-word files are stored in the FTDATA\FTNoiseThesaurusBak folder, and you can use them later when you update or build corresponding SQL Server 2014 stoplists.

After the upgrade:

- If you never added, modified, or deleted any noise-word files in your installation of 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 2014 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 2014 does not create stoplists to implement noise-word files in any upgrade scenario, so you have to create them manually. For more information, see [Configure and Manage Stopwords and Stoplists for Full-Text Search](http://msdn.microsoft.com/en-us/library/ms142551(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms142551\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms142551(v=sql.120).aspx)) in SQL Server 2014 Books Online.

Conclusion

Full-text search is now fully integrated in the SQL Server 2014 Database Engine, and you upgrade full-text indexes by using the same process as for the base database. SQL Server 2014 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 References

For an up-to-date collection of additional references for upgrading full-text to SQL Server 2014, see the following links:

- [SQL Server 2014 Web Site](http://www.microsoft.com/en-us/server-cloud/products/sql-server/)
(<http://www.microsoft.com/en-us/server-cloud/products/sql-server/>)
- [Books Online for SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms130214(v=sql.120).aspx)
([http://msdn.microsoft.com/en-us/library/ms130214\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms130214(v=sql.120).aspx))
- [SQL Server Developer Center](http://msdn.microsoft.com/en-us/sqlserver/)
(<http://msdn.microsoft.com/en-us/sqlserver/>)
- [SQL Server TechNet Resources](http://technet.microsoft.com/en-us/sqlserver)
(<http://technet.microsoft.com/en-us/sqlserver>)

Chapter 7: 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 2014 retains all the features and functionality from SQL Server 2005, and there are no behavior changes that would affect an upgrade. However, 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 2014.

Feature Changes

The major features for Service Broker in SQL Server 2014 (which were actually introduced in SQL Server 2008) are as follows:

- Conversation priority capability
- A diagnostic tool
- More detailed support in SQL Server Management Studio (SSMS)
- Windows System Monitor objects and counters

- Tutorials to help you get started with Service Broker
- More flexible activation options

This guide is focused on upgrade. Therefore, exploring these 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 2014 Service Broker solutions, see [Planning and Architecture \(Service Broker\)](http://msdn.microsoft.com/en-us/library/bb522900(v=sql.105).aspx) (http://msdn.microsoft.com/en-us/library/bb522900(v=sql.105).aspx) in SQL Server 2008 R2 Books Online. (This documentation is not included in SQL Server 2014 Books Online due to the small number of changes since SQL Server 2008 R2).

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 about these options, 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 the Service Broker conversation priority feature, when upgrading from a previous version of SQL Server, SQL Server 2014 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. SQL Server 2014 Upgrade Advisor helps you find and fix issues that could prevent an upgrade and identifies items to modify after your move to SQL Server 2014. And SQL Server Best Practices Analyzer (BPA) 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 2014.

Running Upgrade Advisor

SQL Server 2014 Upgrade Advisor helps you prepare for upgrades to SQL Server 2014. 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

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 2014 Feature Pack](http://www.microsoft.com/en-us/download/details.aspx?id=42295) (<http://www.microsoft.com/en-us/download/details.aspx?id=42295>).

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 all previous versions of SQL Server. SQL Server BPA gathers data from Windows and SQL Server configuration settings, using a predefined list of SQL Server 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 2012 Best Practices Analyzer](http://www.microsoft.com/en-us/download/details.aspx?id=29302) (<http://www.microsoft.com/en-us/download/details.aspx?id=29302>), which is the latest release of this tool, from the Microsoft Download Center.

Once you have upgraded to SQL Server 2014, you can use the SQL Server 2012 Best Practices Analyzer to further refine your systems.

64-bit Considerations

Table 1 shows the supported architectures for SQL Server 2014 Service Broker. Please note that SQL Server 2014 no longer supports the Itanium-based architecture.

Table 1: Architectures Supported in SQL Server 2014 Service Broker

Architecture	Supported
X86 (32 bit)	Yes
X64	Yes
IA64	No

Upgrading from previous version of SQL Server

There are a few Service Broker-related items to keep in mind during your upgrade from previous versions of SQL Server to SQL Server 2014.

In-Place Upgrade

Service Broker operations do not change when a database or an instance of the Database Engine is upgraded from previous versions of SQL Server to SQL Server 2014. The Service Broker features available in previous SQL Server versions have the same behavior in SQL Server 2014.

Previous SQL Server version databases are upgraded to SQL Server 2014 when the following are true:

- They are attached to an instance of the SQL Server 2014 Database Engine after they are detached from an instance of SQL Server 2005/2008/2008 R2/2012 Database Engine.
- The instance of the Database Engine they are in is upgraded from SQL Server 2005/2008/2008 R2/2012 to SQL Server 2014.

You do not have to process existing data/messages in the queues before the 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://msdn.microsoft.com/en-us/library/ms166056(v=sql.105).aspx) ([http://msdn.microsoft.com/en-us/library/ms166056\(v=sql.105\).aspx](http://msdn.microsoft.com/en-us/library/ms166056(v=sql.105).aspx)) in SQL Server 2008 R2 Books Online. (This documentation is not reproduced in SQL Server 2014 Books Online due to the small number of changes in Service Broker in SQL Server 2014.)

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 sending a message might not use the same collation as the database that hosts the service receiving 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://msdn.microsoft.com/en-us/library/ms166129(v=sql.105).aspx) (http://msdn.microsoft.com/en-us/library/ms166129(v=sql.105).aspx) in SQL Server 2008 R2 Books Online. (This documentation is not reproduced in SQL Server 2014 Books Online due to the small number of changes in Service Broker in SQL Server 2014.)

Preserving Settings

In a side-by-side upgrade, you detach databases from a previous version of SQL Server and attach them to SQL Server 2014. The result is that Service Broker is disabled, meaning that the following bits are set to 0 and you must note which bits are set to 1 before detaching:

- is_broker_enabled
- is_honor_broker_priority_on
- is_trustworthy_on

You can see these settings for all databases by running the following query:

```
SELECT * FROM sys.databases
```

Post-Upgrade Tasks

After the upgrade to SQL Server 2014, 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 introduced 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.

If you are upgrading a SQL Server 2005 database to SQL Server 2014, 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://msdn.microsoft.com/en-us/library/bb934439(v=sql.105).aspx) (http://msdn.microsoft.com/en-us/library/bb934439(v=sql.105).aspx) in SQL Server 2008 R2 Books Online. (This documentation is not reproduced in SQL Server 2014 Books Online due to the small number of changes in Service Broker in SQL Server 2014.)

Conclusion

Upgrading a previous version of SQL Server Service Broker to SQL Server 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 2014, see the following links:

- [SQL Server Service Broker](http://msdn.microsoft.com/library/bb522893.aspx)
(<http://msdn.microsoft.com/library/bb522893.aspx>)
- [SQL Server 2014 Web Site](http://www.microsoft.com/sqlserver/en/us/default.aspx)
(<http://www.microsoft.com/sqlserver/en/us/default.aspx>)
- [Books Online for SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms130214(v=sql.120).aspx)
([http://msdn.microsoft.com/en-us/library/ms130214\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms130214(v=sql.120).aspx))
- [SQL Server Developer Center \(MSDN\)](http://msdn.microsoft.com/en-us/sqlserver)
(<http://msdn.microsoft.com/en-us/sqlserver>)
- [SQL Server Resources \(Technet\)](http://technet.microsoft.com/en-us/sqlserver/dn135309)
(<http://technet.microsoft.com/en-us/sqlserver/dn135309>)

Chapter 8: SQL Server Express

Introduction

SQL Server 2014 Express is a free version of SQL Server 2014 that delivers a rich set of data features, data protection, and performance. SQL Server Express can be used in the following ways:

- As a local data store
- Embedded with an application
- As a lightweight database server for applications and small web sites

The following SQL Server 2014 Express packages are available:

- LocalDB (.msi installer) is a lightweight version of Express that has all its programmability features, yet runs in user mode. It has a fast, zero-configuration installation and a short list of prerequisites.
- Express (database engine only) is the core Express database server.
- Management Studio contains the tools needed to manage SQL Server instances, including LocalDB, SQL Express, SQL Azure, and the full version of SQL Server.
- Express with Tools contains everything needed to install and configure SQL Server as a database server, including the full version of SQL Server 2014 Management Studio.
- Express with Advanced Services contains all the components of SQL Server Express, including the full version of SQL Server 2014 Management Studio. This is a larger download than “with Tools,” as it also includes both full-text search and Reporting Services.

SQL Server 2014 Express is the ideal upgrade from the SQL Server 2005/2008/2008 R2/2012 Express Editions. It includes many features that make it a compelling upgrade proposition from any of these previous versions.

A direct upgrade path from Microsoft Data Engine (MSDE) to SQL Server 2014 Express is not supported. To upgrade an existing MSDE instance, you must first upgrade to a SQL Server 2005/2008/2008 R2 Express instance and then upgrade to SQL Server 2014 Express.

LocalDB

LocalDB is a version of SQL Server Express created specifically for developers. It is very easy to install and requires no management, yet it offers the same Transact-SQL (T-SQL) language, programming surface, and client-side providers as the regular SQL Server Express. Developers who target SQL Server no longer have to install and manage a full instance of SQL Server Express on their laptops and other development machines. Moreover, if the simplicity (and limitations) of LocalDB fit the needs of the target application environment, developers can continue using it in production, as LocalDB makes a pretty good embedded database, too.

You can download SQL Server 2014 LocalDB as separated package from the [Microsoft SQL Server 2014 Express](http://www.microsoft.com/en-US/download/details.aspx?id=42299) web site (<http://www.microsoft.com/en-US/download/details.aspx?id=42299>).

Feature Changes

SQL Server 2014 Express supports all the core database functionality that SQL Server 2005/2008/2008 R2/2012 Express provides. This lets almost all existing database applications work without modifications. This functionality includes support for most of the SQL Server features, including Common Language Runtime (CLR) support, XQuery, dynamic management views, and user-schema separation.

In addition, SQL Server Express can rely on a set of management tools. SQL Server Express users can use SQL Server Computer Manager to start and stop database services. You can use SQL Server Configuration Manager to limit potential security risks by controlling network connections and shutting down unused services. You can also manage SQL Server Express by using the full version of SQL Server Management Studio (SSMS), which is included in SQL Server 2014 Express with Tools and with Advanced Services. It is also available as a separate download.

You can download SQL Server 2014 Express from the [Microsoft SQL Server 2014 Express](http://www.microsoft.com/en-US/download/details.aspx?id=42299) web site (<http://www.microsoft.com/en-US/download/details.aspx?id=42299>).

Preparing to Upgrade

Table 1 shows the upgrade paths that Microsoft supports to SQL Server 2014 Express.

Table 1: Upgrade Paths to SQL Server 2014 Express

Upgrade From	Supported Upgrade Paths
SQL Server 2005 SP4 Express, SQL Server 2005 SP4 Express with Tools, and SQL Server 2005 SP4 Express with Advanced Services	SQL Server 2014 Enterprise SQL Server 2014 Business Intelligence SQL Server 2014 Standard SQL Server 2014 Web SQL Server 2014 Express
SQL Server 2008 SP3 Express, SQL Server 2008 SP3 Express with Tools, and SQL Server 2008 SP3 Express with Advanced Services	SQL Server 2014 Enterprise SQL Server 2014 Business Intelligence SQL Server 2014 Standard SQL Server 2014 Web SQL Server 2014 Express
SQL Server 2008 R2 SP2 Express, SQL Server 2008 R2 SP2 Express with Tools, and SQL Server 2008 R2 SP2 Express with Advanced Services	SQL Server 2014 Enterprise SQL Server 2014 Business Intelligence SQL Server 2014 Standard SQL Server 2014 Web SQL Server 2014 Express
SQL Server 2012 SP1 Express, SQL Server 2012 SP1 Express with Tools, SQL Server 2012 SP1 Express Management Studio, and SQL Server 2012 SP1 Express with Advanced Services	SQL Server 2014 Enterprise SQL Server 2014 Business Intelligence SQL Server 2014 Standard SQL Server 2014 Web SQL Server 2014 Express

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. 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 2 shows the features in three types of packages available for SQL Server Express.

Table 2: SQL Server Express Packages Features

Feature	SQL Server 2014 Express	SQL Server 2014 Express with Tools	SQL Server 2014 Express with Advanced Services	LocalDB
	Management			
Windows PowerShell integration	Yes	Yes	Yes	Yes
Policy-Based Management	Yes (manual only)*	Yes (manual only)*	Yes (manual only)*	Yes (manual only)*

Feature	SQL Server 2014 Express	SQL Server 2014 Express with Tools	SQL Server 2014 Express with Advanced Services	LocalDB
SSMS	No	Yes	Yes	No (use SQLLocalDB.exe or SSMS as separate download)
SQL Engine				
Integrated full-text search	No	No	Yes	No
MERGE and UPSERT	Yes	Yes	Yes	No
New data type support				
Filestream support	Yes	Yes	Yes	No
New Date and Time data types	Yes	Yes	Yes	Yes
Geodetic data types	Yes	Yes	Yes	Yes
Advanced spatial libraries	Yes	Yes	Yes	Yes
Support for spatial standards	Yes	Yes	Yes	Yes
New tools				
Import/Export Wizard	Yes	Yes	Yes	Yes
Replication				
Change tracking	Yes	Yes	Yes	Yes
Synchronization Services	Yes (separate installation)**	Yes (separate installation)**	Yes	Yes (separate installation)**
Reporting Services				
Increase RS Memory Limit	No	No	Yes	No
RS Word/Rich Text Export	No	No	Yes	No
IIS Agnostic Report Deployment	No	No	Yes	No
Enhanced gauges and charting	No	No	Yes	No
Business Intelligence Development Studio	No	No	Yes	No

* Policies can be created in SQL Server 2014 Express and run manually. There is no support for automated policy-based management.

** Synchronization Services support in SQL Server 2014 Express requires that you install the component separately from the [Microsoft Download Center](http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=23217) (<http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=23217>).

Deprecated Features

There are no deprecated features for SQL Server 2014 Express. For details about deprecated features, see the following SQL Server 2014 Books Online topics:

- [Deprecated SQL Server Features in SQL Server 2014](http://msdn.microsoft.com/en-us/library/cc707789(v=sql.120).aspx)
(http://msdn.microsoft.com/en-us/library/cc707789(v=sql.120).aspx)
- [Deprecated Features in SQL Server Reporting Services in SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms143509(v=sql.120).aspx)
(http://msdn.microsoft.com/en-us/library/ms143509(v=sql.120).aspx)

Discontinued Functionality

There are no discontinued functionalities for SQL Server 2014 Express. For details about discontinued functionality, see the following SQL Server 2014 Books Online topics:

- [Discontinued SQL Server Features in SQL Server 2014](http://msdn.microsoft.com/en-us/library/cc707782(v=sql.120).aspx)
(http://msdn.microsoft.com/en-us/library/cc707782(v=sql.120).aspx)
- [Discontinued Functionality to SQL Server Reporting Services in SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms144231(v=sql.120).aspx)
(http://msdn.microsoft.com/en-us/library/ms144231(v=sql.120).aspx)

Breaking Changes

Starting in SQL Server 2014 Express, LocalDB moved away from the automatic numbering of the automatic instance and was named MSSQLLocalDB instead of the more confusing "v12.0". If you want to connect to a newly installed LocalDB instance, you should use "(LocalDB)\MSSQLLocalDB".

There are no other breaking changes in SQL Server 2014 Express. For details about breaking changes, see the following SQL Server 2014 Books Online topics:

- [Breaking Changes to SQL Server Features in SQL Server 2014](http://msdn.microsoft.com/en-us/library/cc707784(v=sql.120).aspx)
(http://msdn.microsoft.com/en-us/library/cc707784(v=sql.120).aspx)
- [Breaking Changes in SQL Server Reporting Services in SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms143380(v=sql.120).aspx)
(http://msdn.microsoft.com/en-us/library/ms143380(v=sql.120).aspx)

Behavior Changes

There are no documented behavior changes in SQL Server 2014 Express. For more details about behavior changes that you need to watch out for, see the following SQL Server 2014 Books Online topics:

- [Behavior Changes to SQL Server Features in SQL Server 2014](http://msdn.microsoft.com/en-us/library/cc707784(v=sql.120).aspx)

([http://msdn.microsoft.com/en-us/library/cc707785\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/cc707785(v=sql.120).aspx))

- [Behavior Changes to SQL Server Reporting Services in SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms143200(v=sql.120).aspx)
([http://msdn.microsoft.com/en-us/library/ms143200\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143200(v=sql.120).aspx))

Upgrade Tools

You can take advantage of a variety of tools designed to make the upgrade to SQL Server 2014 Express an easier process:

- SQL Server 2014 Upgrade Advisor analyzes installed components from earlier versions of SQL Server and generates a report that identifies issues to fix either before or after you upgrade.
- SQL Server Best Practices Analyzer analyzes the system and generates a report based on a predefined list of SQL Server recommendations.

Running Upgrade Advisor

SQL Server 2014 Upgrade Advisor helps you prepare for upgrades to SQL Server 2014. 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

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 Upgrade Advisor as part of the [Microsoft SQL Server 2014 Feature Pack](http://www.microsoft.com/en-us/download/details.aspx?id=42295) (<http://www.microsoft.com/en-us/download/details.aspx?id=42295>).

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 all SQL Server versions. After gathering data from Windows and SQL Server configuration settings, BPA uses a predefined list of SQL Server 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. In the Microsoft Download Center, you can download the [SQL Server 2005 BPA](http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=23864) (http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=23864) or [SQL Server 2008 R2 BPA](http://www.microsoft.com/download/en/details.aspx?id=15289) (http://www.microsoft.com/download/en/details.aspx?id=15289).

Once you have upgraded to SQL Server 2014, you can use the SQL Server 2012 BPA, which is the latest version of the tool, to further refine your systems. You can download this tool from the [Microsoft SQL Server 2012 Best Practices Analyzer](http://www.microsoft.com/download/en/details.aspx?id=29302) web page (http://www.microsoft.com/download/en/details.aspx?id=29302).

64-Bit Considerations

Table 3 shows the supported 64-bit architectures for SQL Server 2014 Express.

Table 3: 64-Bit Architectures Supported by SQL Server 2014 Express

Architecture	Supported
X86 (32 bit)	Yes
X64	Yes
IA64	No

SQL Server 2014 Express Packages

Table 4 lists the SQL Server 2014 Express packages that are available.

Table 4: Available SQL Server 2014 Express Packages

Package Name	Purpose
LocalDB 32-bit\SqlLocalDB.msi	32-bit native or 32-bit WOW on 64-bit systems – LocalDB only installation
LocalDB 64-bit\SqlLocalDB.msi	64-bit only – LocalDB only installation
SQLEXP32_x86_ENU.exe	32-bit only – database server-only installation
SQLEXP_x86_ENU.exe	32-bit native or 32-bit WOW on 64-bit systems – database server-only installation
SQLEXP_x64_ENU.exe	64-bit only – database server-only installation
SQLEXPWT_x86_ENU.exe	32-bit native or 32-bit WOW on 64-bit systems – with Tools and LocalDB

SQLEXPRWT_x64_ENU.exe	64-bit only – with Tools and LocalDB
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
SQLManagementStudio_x86_ENU.exe	SSMS – 32-bit only
SQLManagementStudio_x64_ENU.exe	SSMS – 64-bit only

Note that in the table, ENU refers to the English-language version. Other language versions are also available.

System Requirements for SQL Server 2014 Express

The following sections list the minimum hardware and software requirements to install and run SQL Server 2014 Express.

For both the 32-bit and 64-bit editions of SQL Server 2014 Express, the following apply:

- We recommend that you run SQL Server 2014 Express on computers with the NTFS file format. Installing SQL Server 2014 Express on a computer with the FAT32 file system is supported but not recommended as it is less secure than the NTFS file system.
- We recommend using native 64-bit SQL Server Express on 64-bit Windows versions.
- To make sure that the Visual Studio component can be installed correctly, SQL Server requires you to install an update. SQL Server Setup checks for the presence of this update and then requires you to download and install the update before you can continue with the SQL Server installation. To avoid the interruption during SQL Server Setup, you can download and install the update before running SQL Server Setup as described below (or install all the updates for .NET Framework 3.5 SP1 available on Windows Update):
 - If you install SQL Server 2014 on a computer with the Windows 7 SP1 or Windows Server 2008 R2 SP1 operating system, this update is included.
- The installation of SQL Server 2014 fails if you launch the setup through Terminal Services Client. Launching SQL Server Setup through Terminal Services Client is not supported.

Table 5 shows the system requirements for SQL Server Express (all versions), taken from [Hardware and Software Requirements for Installing SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms143506(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms143506\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143506(v=sql.120).aspx)).

Table 5: SQL Server 2014 Express (All Versions) System Requirements

Component	Requirement
.NET Framework	<p>Based on selected features during the setup of SQL Server 2014 Express edition, you may have different .NET framework prerequisites.</p> <p>.NET Framework 4.0 is a requirement for SQL Server 2014. SQL Server Setup installs the following software components required by the product:</p> <ul style="list-style-type: none"> • .NET Framework 4.0 • SQL Server Native Client • SQL Server Setup support files <p>Ensure that an Internet connection is available on the computer. SQL Server Setup downloads and installs the .NET Framework 4.0 because it is not included in the SQL Server Express media. SQL Server Setup will download .NET Framework 4.0 to complete the installation of the prerequisites.</p> <p>SQL Server Express does not install .NET 4.0 on the Server Core mode of Windows Server 2008 R2 SP1 or Windows Server 2012. You must install .NET 4.0 before you install SQL Server Express on a Server Core installation of Windows Server 2008 R2 SP1 or Windows Server 2012.</p> <p>.NET Framework 3.5 SP1 is a requirement for SQL Server 2014 Express Edition only when you select Database Engine, Reporting Services, Replication, or SSMS, but it is no longer installed by SQL Server Setup.</p> <ul style="list-style-type: none"> • If you run Setup and you do not have .NET 3.5 SP1, SQL Server Setup requires you to download and install .NET 3.5 SP1 before you can continue with the SQL Server installation. (Install .NET 3.5 SP1 from Microsoft .NET Framework 3.5 Service Pack 1.) The error message includes a link to the download center, or you can download .NET 3.5 SP1 from Windows Update. To avoid interruption during SQL Server Setup, you can download and install .NET 3.5 SP1 before you run SQL Server Setup. • If you run Setup on a computer with Windows Server 2008 R2 SP1 or Windows 8, you must enable .NET Framework 3.5 SP1 before you install SQL Server 2014. <p>SQL Server LocalDB does not have these requirements.</p>
Windows PowerShell	<p>SQL Server 2014 does not install or enable Windows PowerShell; however, Windows PowerShell 2.0 is an installation prerequisite. If Setup reports that Windows PowerShell 2.0 is not present, you can install or enable it by following the instructions on the Windows Management Framework page (http://support.microsoft.com/kb/968929).</p>
Network software	<p>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.</p> <p>Standalone named and default instances support the following network protocols:</p> <ul style="list-style-type: none"> • Shared memory

Component	Requirement
	<ul style="list-style-type: none"> • Named pipes • TCP/IP • VIA <p>Note: Shared memory and VIA are not supported on failover clusters. The VIA protocol is deprecated and 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.</p>
Virtualization	<p>SQL Server 2014 is supported in virtual machine environments running on the Hyper-V role in:</p> <ul style="list-style-type: none"> • Windows Server 2008 SP2 Standard, Enterprise, and Datacenter editions • Windows Server 2008 R2 SP1 Standard, Enterprise, and Datacenter editions. • Windows Server 2012 Datacenter and Standard editions. <p>In addition to resources required by the parent partition, each virtual machine (child partition) must be provided with sufficient processor resources, memory, and disk resources for its SQL Server 2014 instance.</p> <p>Within the Hyper-V role on Windows Server 2008 SP2 and Windows Server 2008 R2 SP1, a maximum of four virtual processors can be allocated to virtual machines running Windows Server 2008 SP2 or Windows Server 2008 R2 SP1 32-bit or 64-bit editions.</p>
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 2014 Express requires a Super-VGA (800x600) or higher resolution monitor.

Table 6 shows the processor, memory, and operating system requirements for the 32-bit version of SQL Server 2014 Express (Express, Express with Tools, and Express with Advanced Services packages).

Table 6: SQL Server 2014 Express (32-Bit) Processor, Memory, and Operating System Requirements

Component	Requirement
Processor	<p>Minimum:</p> <ul style="list-style-type: none"> • X86 processor 1.0 GHz - Pentium III-compatible processor or faster • X64 processor 1.6 GHz: AMD Opteron, AMD Athlon 64, Intel Xeon with Intel EM64T support, Intel Pentium IV with EM64T support <p>Recommended: 2.0 GHz or faster</p>
Operating system	<p>Windows Server 2012 R2 Datacenter 64-bit</p> <p>Windows Server 2012 R2 Standard 64-bit</p> <p>Windows Server 2012 R2 Essentials 64-bit</p>

Component	Requirement
	Windows Server 2012 R2 Foundation 64-bit
	Windows Server 2012 Datacenter 64-bit
	Windows Server 2012 Standard 64-bit
	Windows Server 2012 Essentials 64-bit
	Windows Server 2012 Foundation 64-bit
	Windows Server 2008 R2 SP1 Datacenter 64-bit
	Windows Server 2008 R2 SP1 Enterprise 64-bit
	Windows Server 2008 R2 SP1 Standard 64-bit
	Windows Server 2008 R2 SP1 Foundation 64-bit
	Windows Server 2008 R2 SP1 Web 64-bit
	Windows 8.1 32-bit
	Windows 8.1 Pro 32-bit
	Windows 8.1 Enterprise 32-bit
	Windows 8.1 64-bit
	Windows 8.1 Pro 64-bit
	Windows 8.1 Enterprise 64-bit
	Windows 8 32-bit
	Windows 8 Pro 32-bit
	Windows 8 Enterprise 32-bit
	Windows 8 64-bit
	Windows 8 Pro 64-bit
	Windows 8 Enterprise 64-bit
	Windows 7 SP1 Ultimate 64-bit
	Windows 7 SP1 Enterprise 64-bit
	Windows 7 SP1 Professional 64-bit
	Windows 7 SP1 Home Premium 64-bit
	Windows 7 SP1 Home Basic 64-bit
	Windows 7 SP1 Ultimate 32-bit
	Windows 7 SP1 Enterprise 32-bit
	Windows 7 SP1 Professional 32-bit
	Windows 7 SP1 Home Premium 32-bit
	Windows 7 SP1 Home Basic 32-bit
	Windows Server 2008 SP2 Datacenter 64-bit
	Windows Server 2008 SP2 Enterprise 64-bit
	Windows Server 2008 SP2 Standard 64-bit
	Windows Server 2008 SP2 Foundation 64-bit
	Windows Server 2008 SP2 Web 64-bit
	Windows Server 2008 SP2 Datacenter 64-bit
	Windows Server 2008 SP2 Enterprise 32-bit
	Windows Server 2008 SP2 Standard 32-bit
	Windows Server 2008 SP2 Web 32-bit

Component	Requirement
Memory	RAM: <ul style="list-style-type: none"> • Minimum: 512 MB • Recommended: 1 GB

Table 7 shows the processor, memory and operating system requirements for the 64-bit version of SQL Server 2014 Express (Express, Express with Tools, and Express with Advanced Services packages).

Table 7: SQL Server 2014 Express (64-Bit) Processor, Memory, and Operating System Requirements

Component	Requirement
Processor	<p>Minimum:</p> <ul style="list-style-type: none"> • X86 processor 1.0 GHz - Pentium III-compatible processor or faster • X64 processor 1.6 GHz: AMD Opteron, AMD Athlon 64, Intel Xeon with Intel EM64T support, Intel Pentium IV with EM64T support <p>Recommended: 2.0 GHz or faster</p>
Operating system	<p>Windows Server 2012 R2 Datacenter 64-bit</p> <p>Windows Server 2012 R2 Standard 64-bit</p> <p>Windows Server 2012 R2 Essentials 64-bit</p> <p>Windows Server 2012 R2 Foundation 64-bit</p> <p>Windows Server 2012 Datacenter 64-bit</p> <p>Windows Server 2012 Standard 64-bit</p> <p>Windows Server 2012 Essentials 64-bit</p> <p>Windows Server 2012 Foundation 64-bit</p> <p>Windows Server 2008 R2 SP1 Datacenter 64-bit</p> <p>Windows Server 2008 R2 SP1 Enterprise 64-bit</p> <p>Windows Server 2008 R2 SP1 Standard 64-bit</p> <p>Windows Server 2008 R2 SP1 Foundation 64-bit</p> <p>Windows Server 2008 R2 SP1 Web 64-bit</p> <p>Windows 8.1 64-bit</p> <p>Windows 8.1 Pro 64-bit</p> <p>Windows 8.1 Enterprise 64-bit</p> <p>Windows 8 64-bit</p> <p>Windows 8 Pro 64-bit</p> <p>Windows 8 Enterprise 64-bit</p> <p>Windows 7 SP1 Ultimate 64-bit</p> <p>Windows 7 SP1 Enterprise 64-bit</p> <p>Windows 7 SP1 Professional 64-bit</p> <p>Windows 7 SP1 Home Premium 64-bit</p> <p>Windows 7 SP1 Home Basic 64-bit</p> <p>Windows Server 2008 SP2 Datacenter 64-bit</p>

Component	Requirement
	Windows Server 2008 SP2 Enterprise 64-bit Windows Server 2008 SP2 Standard 64-bit Windows Server 2008 SP2 Foundation 64-bit Windows Server 2008 SP2 Web 64-bit
Memory	RAM: <ul style="list-style-type: none"> • Minimum: 512 MB • Recommended: 1 GB

Upgrading from SQL Server 2000 (MSDE)

Upgrading from MSDE to SQL Server 2014 Express is not supported. Existing MSDE instances must be first upgraded to SQL Server 2005/2008/2008 R2 Express before being upgraded to SQL Server 2014 Express.

Upgrading to SQL Server 2014 Express

In-Place Upgrade

To perform an in-place upgrade from previous SQL Server Express versions (excluding MSDE) to SQL Server 2014 Express, take the following steps:

1. Download and install .NET Framework 3.5 SP1, if missing. It is a prerequisite for SQL Server 2014 Express, but is no longer installed by SQL Server Express setup. You can download [.NET Framework 3.5 SP1](http://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=22) (http://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=22) from the Microsoft Download Center and then install it by running the dotnetfx.exe program.
2. Start the SQL Server 2014 Setup program, and install the prerequisite software. SQL Server 2014 Express is installed by running the executable (i.e., .exe) program for the package type you are installing.
3. Specify the remaining configuration options (generally accept all defaults), and then click Install in the Ready to Install dialog box. This will upgrade the specified instance to SQL Server 2014 Express.

Side-by-Side Upgrade

To upgrade from previous SQL Server Express versions 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.

1. Log in to the previous SQL Server 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 previous SQL Server Express system by using SSMS (Express or another 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 using the backup/restore method instead, but the detach/attach method is generally easier.)
4. Shut down the previous SQL Server Express instance by opening SQL Server Configuration Manager and stopping the SQL Server services.
5. Repeat Step 4 for the Distributed Transaction Coordinator and 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 perform this step later if the names of the instances containing the old and new versions of SQL Server Express are different.
7. Download and install .NET Framework 3.5 SP1, if missing. It is a prerequisite for SQL Server 2014 Express but is no longer installed by SQL Server Express setup. You can download [.NET Framework 3.5 SP1](http://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=22) (http://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=22) from the Microsoft Download Center. After downloading .NET Framework 3.5 SP1, install it by running the dotnetfx.exe program.
8. Install SQL Server 2014 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: In SQL Server 2005 Express and later, the Setup program sets the name of a default instance to SQLEXPRESS rather than the old 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's name.
9. After SQL Server 2014 Express is installed, connect to it using SSMS (Express or another edition).
10. Attach each of the user databases that were detached from the SQL Server previous 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 the detach/attach method.)

11. Enable any needed protocols.

The default installation for SQL Server 2014 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 2014 Network Configuration node, select Protocols for SQLEXPRESS (or your non-default instance name), 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 2014 Express installation by performing the following post-upgrade steps:

1. Use SQL Server 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 2014 Express program group.
2. Within Configuration Manager, open the SQL Server 2014 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 2005/2008/2008 R2 Express that support connections from networked users, it is important to know that SQL Server 2014 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 2014 Network Configuration node, select Protocols for SQLEXPRESS, and then enable the required protocols by right-clicking the protocol and selecting the Enable option from the context menu.

Upgrading to LocalDB

To upgrade from previous SQL Server Express versions to LocalDB, use the following steps if you have the SQL Server Express Management Tools installed. Otherwise, perform the detach/attach operations.

1. Log in to the previous SQL Server 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 previous SQL Server Express system by using SSMS (Express or another 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 using the backup/restore method instead, but the detach/attach method is generally easier.)
4. Shut down previous SQL Server Express instance by opening SQL Server Configuration Manager and stopping the SQL Server services.
5. Repeat Step 4 for the Distributed Transaction Coordinator and 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 perform this step later if the names of the instances containing the old and new versions of SQL Server Express are different.
7. Install SQL Server 2014 LocalDB by running the .msi installer. 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.
8. After SQL Server 2014 LocalDB is installed, connect to it using SSMS (Express or another edition) or the SQLCMD command-line utility.
9. Attach each of the user databases that were detached from the previous SQL Server 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 method instead of the detach/attach method.) If you are using the command-line option, you can attach the databases using the proper T-SQL command.

Upgrading to Other Editions of SQL Server 2014

Although the core database capabilities of SQL Server Express versions are similar, the feature sets and limitations are different. These differences or projected requirements for features outside the SQL Server 2014 Express feature set could cause you to select a different edition of SQL Server 2014 to upgrade to.

Table 8 compares features between the SQL Server 2014 Express, Web, and Standard Editions.

Table 8: Comparing the SQL Server 2014 Express, Web, and Standard Editions

Feature	SQL Server 2014 Express	SQL Server 2014 Web	SQL Server 2014 Standard
Maximum number of instances	50	50	50
Maximum number of processors	Lesser of 1 socket or 4 cores	Lesser of 4 sockets or 16 cores	Lesser of 4 sockets or 16 cores
Maximum RAM	1 GB	64 GB	128 GB
Maximum database size	10 GB	524 TB	524 TB
Server Core Support	Yes	Yes	Yes
High Availability - log shipping	No	Yes	Yes
High Availability - database mirroring	Witness only	Witness only	Yes (Safety Full only)
AlwaysOn Failover Cluster Instances	No	No	Yes (2 nodes)
Backup compression	No	No	Yes
Replication publishing	No	No	Yes
SQL Server Agent	No	Yes	Yes
Database Tuning Advisor	No	Yes	Yes
Business intelligence features (Analysis Services, Integration Services)	No	No	Yes
Report Server	Express with Advanced Services only	Yes	Yes
Supported Report Server catalog database	Yes	Yes	Yes
Supported report data source	Yes	Yes	Yes
Service Broker	Client-only	Client-only	Yes
Full-text semantic search	Express with Advanced Services only	Yes	Yes
SQL Profiler	No	No	Yes
Database Mail	No	Yes	Yes
StreamInsight	No	Yes	Yes
Delayed durability	Yes	Yes	Yes
Buffer pool extension	No	No	Yes
Contained databases	Yes	Yes	Yes

Upgrading to the SQL Server 2014 Web Edition

Upgrading to the SQL Server 2014 Web Edition might be a compelling option in some scenarios. Consider the following four scenarios.

RAM Requirements Beyond the Level Supported by SQL Server 2014 Express

SQL Server 2014 Express supports only 1 GB of RAM. If your SQL Server Express applications need more than this, consider upgrading to SQL Server 2014 Web.

Processor Requirements Beyond the Level Supported by SQL Server 2014 Express

Although SQL Server 2014 Web supports 4 processors with a maximum of 16 cores compared with SQL Server 2014 Express' support for 1 processor with a maximum of 4 cores, it is unlikely that this would necessitate a move to SQL Server 2014 Web. 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.

Applications Need Scheduling Capability

SQL Server 2014 Express does not supply SQL Server Agent. If your application needs to schedule jobs and database tasks, you can use the Windows Task Scheduler or consider upgrading to SQL Server 2014 Web.

Instance Needs to Act as a Replication Publisher

SQL Server 2014 Express does not support using a SQL Server Express instance as a replication Publisher to other SQL Server Express databases. If your application needs to act as a Publisher in a replication scenario, you need to consider upgrading to SQL Server 2014 Web.

Upgrading to SQL Server 2014 Standard Edition

The primary reason you would consider upgrading from SQL Server 2012 Express to SQL Server 2014 Standard is that you predict your future application requirements will exceed the capabilities or feature set available in SQL Server 2014 Express or Web. This upgrade scenario would be based on projections that your future database requirements could exceed 1 GB of RAM, that you will need a database size larger than 10 GB, or that you need the high availability or business intelligence features in SQL Server 2014 Standard.

Note: If you need enterprise features such as data compression, Resource Governor, or table partitioning, you will need to upgrade to SQL Server 2014 Enterprise.

Conclusion

SQL Server 2014 Express is the ideal upgrade path for older SQL Server Express database management systems. The upgrade is straightforward, and there are only a

few issues to review and prepare for before upgrading from previous versions of SQL Server Express. 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 2014 Express, see the following links:

- [SQL Server 2014 Web Site](http://www.microsoft.com/sqlserver/en/us/default.aspx)
(<http://www.microsoft.com/sqlserver/en/us/default.aspx>)
- [Books Online for SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms130214(v=sql.120).aspx)
([http://msdn.microsoft.com/en-us/library/ms130214\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms130214(v=sql.120).aspx))
- [SQL Server Developer Center](http://msdn.microsoft.com/en-us/sqlserver/)
(<http://msdn.microsoft.com/en-us/sqlserver/>)
- [SQL Server TechNet Resources](http://technet.microsoft.com/en-us/sqlserver/)
(<http://technet.microsoft.com/en-us/sqlserver/>)

Chapter 9: SQL Server Data Tools

Introduction

SQL Server Data Tools (SSDT) is a declarative and ubiquitous model that includes all the phases of database development, maintenance, and update inside Visual Studio. It is a rich SQL Server development environment with full Visual Studio integration, so database and application developers can work on SQL Server or SQL Azure projects in Visual Studio. SSDT was introduced in SQL Server 2012 and carries forward in SQL Server 2014. It includes model-based tools that can be used for online and offline development.

This chapter lays out the general guidelines for installing and upgrading from Visual Studio 2010 and SSDT 2012 database projects to SSDT 2014 database projects. This chapter will also cover the most common behavior and breaking changes while upgrading projects from Visual Studio 2010 or SSDT 2012 database projects to SSDT 2014 database projects.

Preparing to Upgrade

Before you upgrade, make sure you can install SSDT:

- The database tools in SSDT have been fully integrated into the following Visual Studio 2013 editions: Express for Web, Express for Windows Desktop, Professional, Premium, and Ultimate. All editions of Visual Studio 2013 Update 3 with integrated SSDT can be downloaded from [Visual Studio Downloads](http://www.visualstudio.com/downloads/download-visual-studio-vs) (<http://www.visualstudio.com/downloads/download-visual-studio-vs>). Because the database tools are included in Visual Studio 2013, updates are pushed through VS Update. Users are prompted about the updates when Visual Studio 2013 is open. You can also manually check for updates using the Extension and Updates dialog box, which you access through the Tools menu in Visual Studio 2013.
- If you want SSDT with Visual Studio 2012, you will need to manually install it from [Download SQL Server Tools for Visual Studio 2012](http://go.microsoft.com/fwlink/?LinkID=400928) (<http://go.microsoft.com/fwlink/?LinkID=400928>).

For a complete overview of the SSDT installation process, see [Microsoft SQL Server Data Tools](http://msdn.microsoft.com/en-us/data/hh297027) (<http://msdn.microsoft.com/en-us/data/hh297027>).

Upgrading Visual Studio 2010/SSDT 2012/Visual Studio 2013 Database Projects to SSDT 2014

SSDT 2012 Database Projects can be used directly in SSDT 2014 (Visual Studio 2013). To convert a Visual Studio 2010 Database Project to an SSDT Database Project, simply right-click the project and choose *Convert to SQL Server Database project*, as shown in Figure 1.

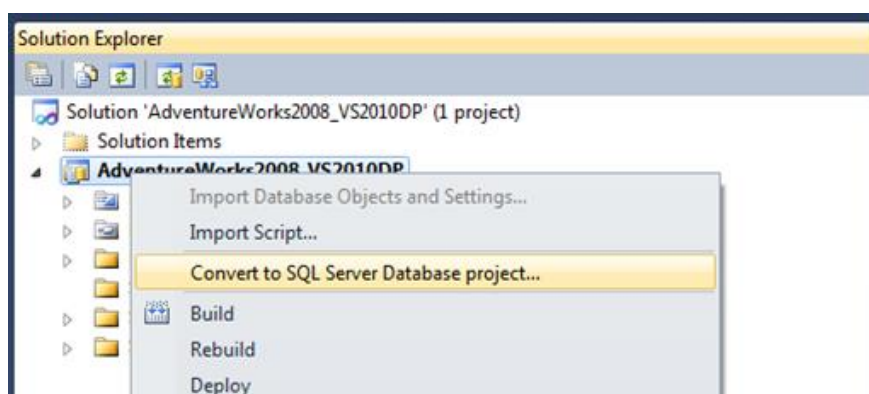


Figure 1: Converting a Visual Studio 2010 Database Project to an SSDT Database Project

While the conversion of the project itself is a fairly straightforward process, while converting Visual Studio 2010 projects to SSDT for SQL 2014, you have to keep in mind that some project types, file types, properties, and settings cannot be upgraded either because of the nature of the element itself or simply because SSDT uses another project structure and different underlying technologies. Table 1 shows the components that will be upgraded without errors.

Table 1: Components That Can Be Upgraded

Element	Notes
Project (.dbproj) files	
Schema comparison (.scmp) files	There are different ways of building schema comparison files in Visual Studio 2010 database projects. Not all of them are supported for conversion.
.sqldeployment, .sqlsettings, and .sqlpolicy files	These files are converted to their respective project properties.
.sqlpermissions files	These files are converted to Transact-SQL (T-SQL) scripts.
.sql files and their folder structure	
Deployment scripts	

Table 2 shows the elements that will not be upgraded. You can expect them either to

be highlighted in the conversion report at the end of the conversion or to fire an error when building the project.

Table 2: Elements That Cannot Be Upgraded

Element	Error Upon	Solution
Referencing other databases with schema files	Conversion	Pre-upgrade task
.sqlcmd files defining SQLCMD variables	Conversion	Pre-upgrade task
Data Generation Plans (.dgen)	Conversion	None
Database Unit Test projects	Conversion	None
Extensibility files	Conversion	None
Server logins	Build	Pre-upgrade task
Full-text search	Build	Post-upgrade task
Linked server definitions	Build	Post-upgrade task

The following sections discuss the issues encountered while converting Visual Studio 2010 Database Projects to SSDT Database Projects and present a solution (if one exists) either as a pre- or post-upgrade task.

Breaking and Behavior Changes

There are a number of important breaking and behavior changes you need to be aware of when upgrading Visual Studio 2010 Database Projects to SSDT Database Projects.

SQL Server Database Project Template

After installing SSDT, you will find the SQL Server Database Project template under New Project | Installed | Templates | SQL Server, as shown in Figure 2.

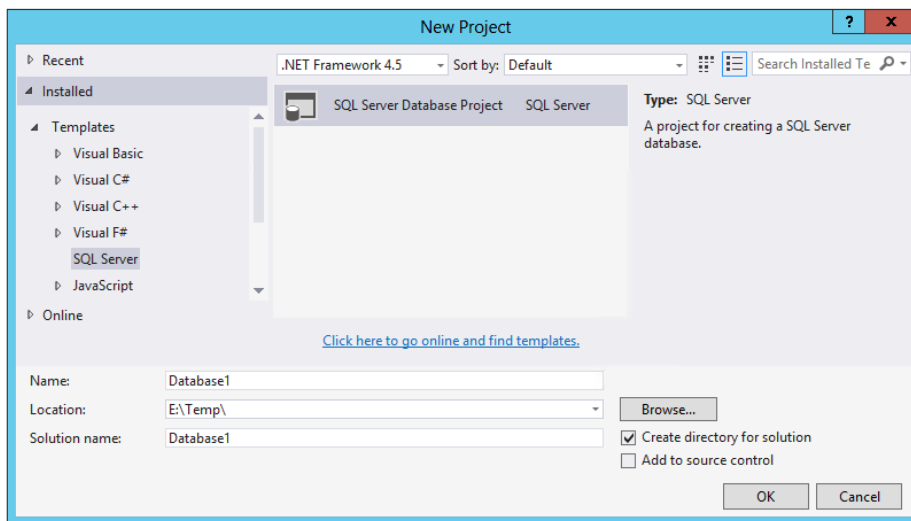


Figure 2: Adding a new SSDT Database Project

Schema Definition Files

In Visual Studio 2010 Database Projects, you can add reference to another user or

system database using a .dbschema file, which is a XML representation of the database model and properties. Figure 3 shows an example of a Visual Studio 2010 Database Project with a database reference.

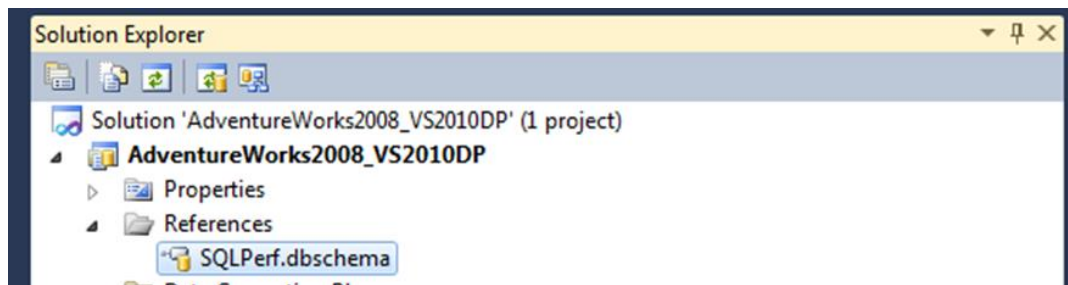


Figure 3: A Visual Studio 2010 Database Project with a database reference

SSDT uses .dacpac files to reference external databases and does not support .dbschema files. Upon conversion, the referenced full path will be converted from a .dbschema file to a .dacpac file, resulting in a broken reference.

Schema Comparison Files

In Visual Studio 2010 Database Projects, you can create schema comparison files in a number of ways:

- Compare with an existing live database by pointing to the database directly
- Compare with a .dbschema file
- Compare with another project in the solution

In SSDT, only schema comparisons pointing to an existing database will be converted successfully.

SQLCMD Variable Definitions

In Visual Studio 2010 Database Projects, the .sqlcmdvar property files allow you to add a set of custom variables and values to be used in scripts, as shown in Figure 4. SSDT will not convert variables in the default build configuration.

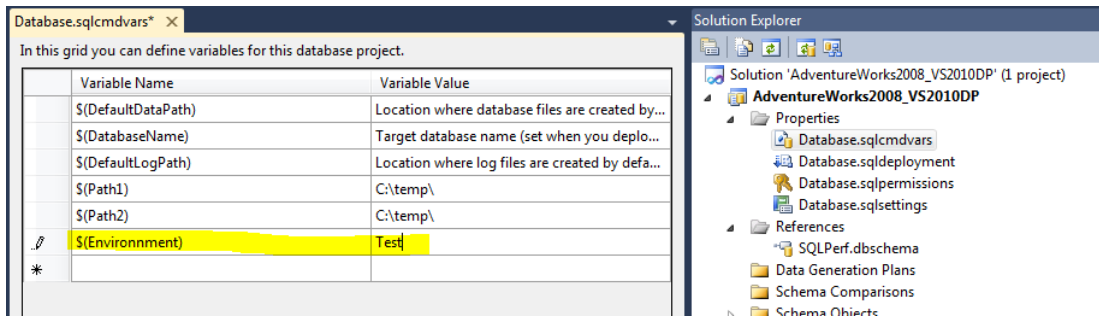


Figure 4: A user-defined variable

Server Logins

In Visual Studio 2010, there are two types of projects: Database Projects and Server Projects. A Database Project that references a Server Project can have server logins configured in the Server Project that are used in the Database Project, as shown in Figure 5.

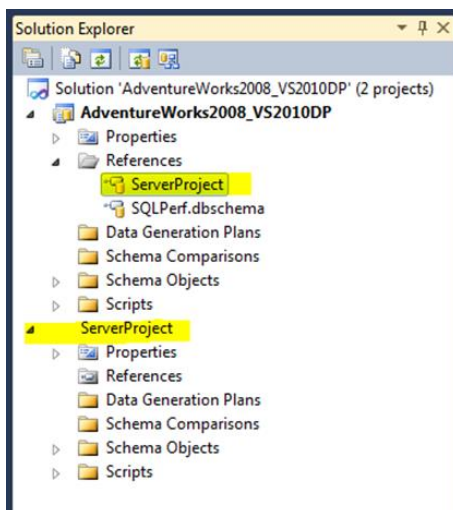


Figure 5: Reference to a Server Project containing logins

This type of referenced server login cannot be converted. Any references to the login in the SSDT database project will result in an error.

Other Breaking and Behavior Changes

Here are some additional breaking and behavior changes:

- **Data generation.** SSDT does not have a data generation tool at this time, but it is a feature that will be added in a future release.
- **Database Unit Test Projects.** This type of project is not currently supported by SSDT.

- **Extensibility files.** This type of file cannot be converted to SSDT.
- **Full-text search.** When you use SSDT and debug your project, you use the new LocalDB, which is an on-demand local instance of SQL Server 2014.
- **Linked server definitions.** In Visual Studio 2010 Database Projects, you are able to define linked servers and use the value from a SQLCMD variable. Used this way, it will convert to SSDT but will raise an error when deploying or debugging.

Pre-Upgrade Tasks

By completing some pre-upgrade tasks, you will be able to solve some of the conversion problems described in the “Breaking and Behavioral Changes” section when you convert your Visual Studio 2010 Database Projects to SSDT Database Projects.

Convert .dbschema Files to .dacpac Files

If you rely on schema definition files from external providers for referencing databases, make sure that you get a DACPAC description of the databases that you want to use in the SSDT Database Project.

You need to create a database from a .dbschema file and then create a SSDT Database Project from the newly created database. Run the VSDBCMD program found in the C:\Program Files\Microsoft Visual Studio 10.0\VSTSDDB\Deploy folder to generate a database from the .dbschema file. For example, the following will create a database named SQLPerf on the SQL Server note1\RC0 based on the schema sqlperf.dbschema:

```
vsdbcmd /a:deploy /dd:+
/cs: "Data source=note01\RC0; Integrated Security=true;"
/model: "c:\temp\sqlperf.dbschema"
/p:TargetDatabase="SQLPerf"
```

In Visual Studio with SSDT, you can use SQL Server Object Explorer to create a new project based on your new database, as shown in Figure 6.

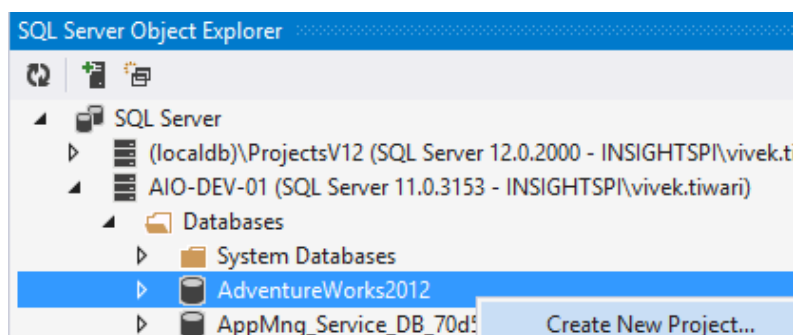


Figure 6: Creating a new SSDT database project from an existing database

Then you can take a snapshot of your project, which will create a .dacpac file that you can use for referencing databases.

Convert SQLCMD Variable Definitions

To include your variable definition when converting a project, you need to use MSBuild by either changing the current configuration or making a new one. Follow these steps:

1. Create a new configuration before converting. Use the Build | Configuration Manager menu to create a new configuration, as shown in Figure 7.

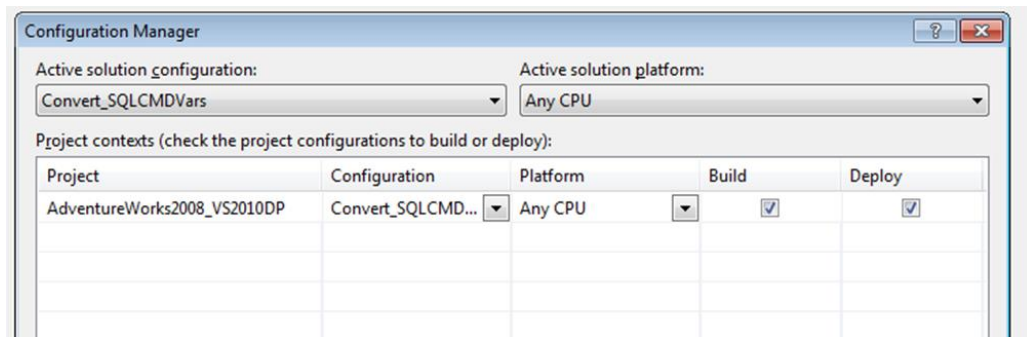


Figure 7: Creating a new configuration

2. Use the Deploy tab of the project properties to indicate where to locate the .sqlcmdvars file, as shown in Figure 8.

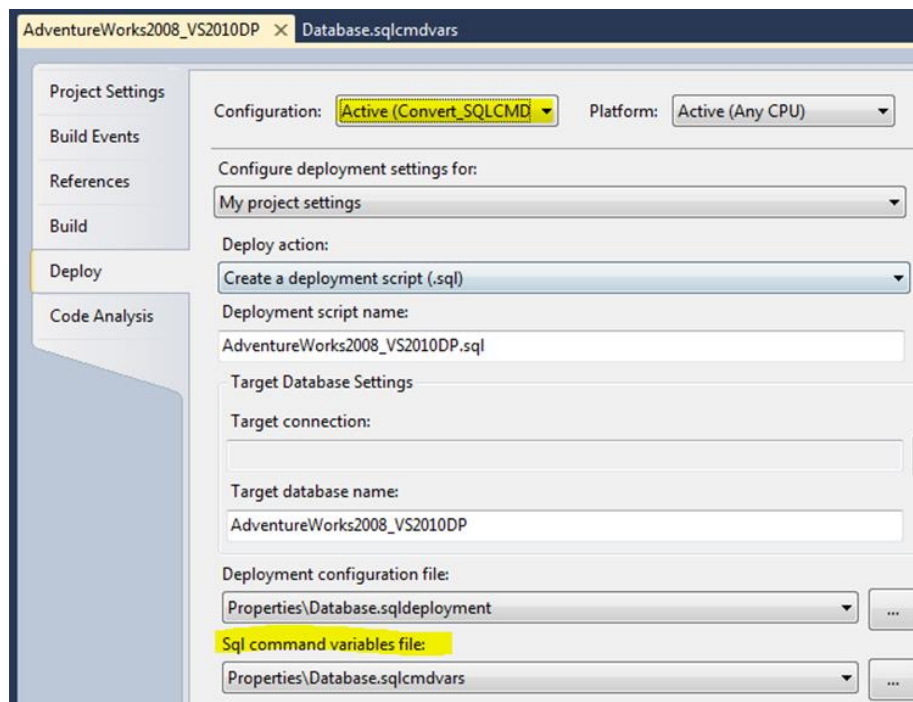


Figure 8: Specifying the location for the .sqlcmdvars file

3. After the conversion, make sure that your variables and their values appear with the name of your custom build configuration in the .publish.xml document, as shown in Figure 9.

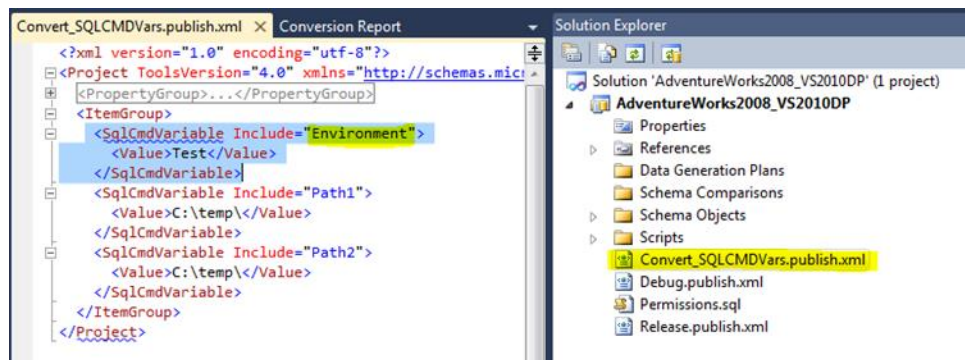


Figure 9: User defined variables after successful conversion

Post-Upgrade tasks

After the conversion, there are still some changes to be made before you can compile, deploy, or publish your projects. The following topics were detailed in the “Breaking and Behavior Changes” section earlier, but if they are still present after your upgrade, you should address them now.

Change the Debugging Instance for Full-Text Search

Since LocalDB does not currently support full-text search, if full-text search is used in a project, the only solution is to change your debugging instance from LocalDB to a SQL Server instance by using the Debug tab on the SSDT project properties page.

Correct Linked Server Definitions That Use SQLCMD Variables

Linked server definitions that use SQLCMD variables in the way shown in Figure 10 cannot be converted.

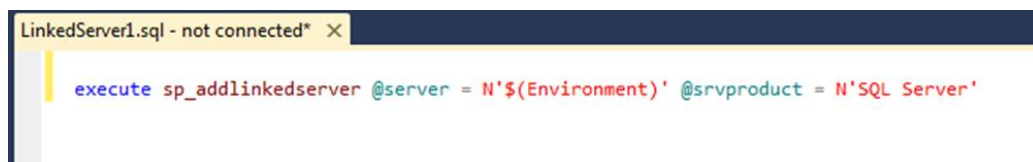


Figure 10: SQLCMD variable being used to define the linked server

There are two alternatives for resolving this issue:

- Use SQLCMD variables in a custom build configuration as described earlier in the “Correct Linked Server Definitions that Use SQLCMD Variables” section.

- Change the syntax of the call to `sp_addlinkedserver` so that it uses the `@datasrc` parameter to reference the variable:

```
execute sp_addlinkedserver
    @server = N'MyLinkedServer' , @srvproduct = N'',
    @provider = N'SQLNCLI', @datasrc = N'$(Environment)'
```

Additional References

For more information, see these resources:

- [Command-Line Reference for VSDBCMD.EXE \(Deployment and Schema Import\)](http://msdn.microsoft.com/en-us/library/dd193283.aspx)
(<http://msdn.microsoft.com/en-us/library/dd193283.aspx>)
- [Microsoft SQL Server Data Tools](http://msdn.microsoft.com/en-us/data/tools.aspx)
(<http://msdn.microsoft.com/en-us/data/tools.aspx>)
- [Download SQL Server Data Tools](http://msdn.microsoft.com/en-us/data/hh297027)
(<http://msdn.microsoft.com/en-us/data/hh297027>)
- [Get Started with Microsoft SQL Server Data Tools](http://msdn.microsoft.com/en-us/data/hh297028)
(<http://msdn.microsoft.com/en-us/data/hh297028>)
- [SQL Server Data Tools Forum](http://social.msdn.microsoft.com/Forums/en-US/ssdt/threads)
(<http://social.msdn.microsoft.com/Forums/en-US/ssdt/threads>)
- [SQL Server Data Tools Team Blog](http://blogs.msdn.com/b/ssdt)
(<http://blogs.msdn.com/b/ssdt>)
- [Books Online for SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms130214(v=sql.120).aspx)
([http://msdn.microsoft.com/en-us/library/ms130214\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms130214(v=sql.120).aspx))

Chapter 10: Transact-SQL Queries

Introduction

A significant but often overlooked component of every upgrade process is upgrading queries and scripts. Although most database administrators (DBAs) 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 2014 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 (T-SQL) features in SQL Server that are not in the version your code is using. 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 SQL Server 2014 Upgrade Advisor.

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 2014 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 T-SQL changes you need to understand for a smooth transition to SQL Server 2014 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 first 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 2014 still supports but that will be removed in a future 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. SQL Server 2014 includes a System Monitor object, the Deprecation Announcement event class, and the `deprecation_announcement` extended event to help you identify deprecated features on your upgraded system.

SQLServer:Deprecated Features Object

After you have upgraded to SQL Server 2014, 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 2014 counterparts. SQL Server 2014 has a System Monitor object named `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 with System Monitor or by using the dynamic management view (DMV) `sys.dm_os_performance_counters`. For more information about this object, see [SQL Server, Deprecated Features Object](http://msdn.microsoft.com/en-us/library/bb510662.aspx) (<http://msdn.microsoft.com/en-us/library/bb510662.aspx>) in SQL Server 2014 Books Online.

Deprecation Announcement Event Class

You can use the Deprecation Announcement event class from SQL Server Profiler to identify deprecated features. For more information about this event class, see [Deprecation Announcement Event Class](http://msdn.microsoft.com/en-us/library/ms186302.aspx) (<http://msdn.microsoft.com/en-us/library/ms186302.aspx>) in SQL Server 2014 Books Online.

Extended Event deprecation announcement

You can use the `deprecation_announcement` extended event to identify deprecated features. For more information about extended events, see [Extended Events](#) ([http://msdn.microsoft.com/en-us/library/bb630282\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/bb630282(v=sql.120).aspx)).

For a comprehensive list of deprecated functionality in SQL Server 2008 R2, SQL Server 2012, and SQL Server 2014, see the following topics in SQL Server Books Online:

- [Deprecated Database Engine Features in SQL Server 2008 R2](#)
([http://msdn.microsoft.com/en-us/library/ms143729\(v=sql.105\).aspx](http://msdn.microsoft.com/en-us/library/ms143729(v=sql.105).aspx))
- [Deprecated Database Engine Features in SQL Server 2012](#)
([http://msdn.microsoft.com/en-us/library/ms143729\(v=sql.110\).aspx](http://msdn.microsoft.com/en-us/library/ms143729(v=sql.110).aspx))
- [Deprecated Database Engine Features in SQL Server 2014](#)
(<http://msdn.microsoft.com/en-us/library/ms143729.aspx>)

Discontinued Features

There are no discontinued features in SQL Server 2014. However, if you are upgrading from pre SQL Server 2012 (let's say SQL Server 2008), you may run into features that were discontinued in SQL Server 2012. If you are upgrading from one of the older versions, review the [SQL Server 2012 Upgrade Technical Guide](#) (http://download.microsoft.com/download/9/5/3/9533501A-6F3E-4D03-A6A3-359AF6A79877/SQL_Server_2012_Upgrade_Technical_Reference_Guide_White_Paper.pdf)

For a comprehensive list of discontinued functionality in SQL Server 2008 R2, SQL Server 2012, and SQL Server 2014, see the following topics in SQL Server Books Online:

- [Discontinued Database Engine Functionality in SQL Server 2008 R2](#)
([http://msdn.microsoft.com/en-us/library/ms144262\(v=sql.105\).aspx](http://msdn.microsoft.com/en-us/library/ms144262(v=sql.105).aspx))
- [Discontinued Database Engine Functionality in SQL Server 2012](#)
([http://msdn.microsoft.com/en-us/library/ms144262\(v=sql.110\).aspx](http://msdn.microsoft.com/en-us/library/ms144262(v=sql.110).aspx))
- [Discontinued Database Engine Functionality in SQL Server 2014](#)
([http://msdn.microsoft.com/en-us/library/ms144262\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms144262(v=sql.120).aspx))

Breaking Changes

There are no breaking changes in the T-SQL area in SQL Server 2014 when upgrading from SQL Server 2012.

Additional Information

For a complete list of breaking changes in SQL Server 2008 R2 and SQL Server 2012, see the following topics in SQL Server Books Online:

- [Breaking Changes to Database Engine Features in SQL Server 2008 R2](http://msdn.microsoft.com/en-us/library/ms143179(v=sql.105).aspx)
(http://msdn.microsoft.com/en-us/library/ms143179(v=sql.105).aspx)
- [Breaking Changes to Database Engine Features in SQL Server 2012](http://msdn.microsoft.com/en-us/library/ms143179(v=sql.110).aspx)
(http://msdn.microsoft.com/en-us/library/ms143179(v=sql.110).aspx)

Behavior Changes

The only behavior change for SQL Server 2014 in the T-SQL area is the correct handling of XML documents larger than 4020 characters. This was a problem in prior versions of SQL Server.

For a complete list of behavior changes in SQL Server 2008 R2 and SQL Server 2012, see the following topics in SQL Server Books Online:

- [Behavior Changes to Database Engine Features in SQL Server 2008 R2](http://msdn.microsoft.com/en-us/library/ms143359(v=sql.105).aspx)
(http://msdn.microsoft.com/en-us/library/ms143359(v=sql.105).aspx)
- [Behavior Changes to Database Engine Features in SQL Server 2012](http://msdn.microsoft.com/en-us/library/ms143359(v=sql.110).aspx)
(http://msdn.microsoft.com/en-us/library/ms143359(v=sql.110).aspx)

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 T-SQL language that SQL Server uses to parse and understand T-SQL statements and batches.

Although you can use T-SQL reserved keywords as identifiers or names of databases or database objects (e.g., tables, columns, views), 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.

SQL Server 2014 Upgrade Advisor will flag stored procedures for usage of new reserved keywords. But be sure to perform a manual review of T-SQL code embedded in application code and other external sources. Table 1 lists all the SQL Server 2014 reserved keywords.

Table 1: SQL Server 2014 Reserved Keywords

ADD	EXTERNAL	PROCEDURE
ALL	FETCH	PUBLIC
ALTER	FILE	RAISERROR
AND	FILLFACTOR	READ
ANY	FOR	READTEXT
AS	FOREIGN	RECONFIGURE
ASC	FREETEXT	REFERENCES
AUTHORIZATION	FREETEXTTABLE	REPLICATION
BACKUP	FROM	RESTORE
BEGIN	FULL	RESTRICT
BETWEEN	FUNCTION	RETURN
BREAK	GOTO	REVERT
BROWSE	GRANT	REVOKE
BULK	GROUP	RIGHT
BY	HAVING	ROLLBACK
CASCADE	HOLDLOCK	ROWCOUNT
CASE	IDENTITY	ROWGUIDCOL
CHECK	IDENTITY_INSERT	RULE
CHECKPOINT	IDENTITYCOL	SAVE
CLOSE	IF	SCHEMA
CLUSTERED	IN	SECURITYAUDIT
COALESCE	INDEX	SELECT
COLLATE	INNER	SEMANTICKEYPHRASETABLE
COLUMN	INSERT	SEMANTICSIMILARITYDETAILSTABLE
COMMIT	INTERSECT	SEMANTICSIMILARITYTABLE
COMPUTE	INTO	SESSION_USER
CONSTRAINT	IS	SET
CONTAINS	JOIN	SETUSER
CONTAINSTABLE	KEY	SHUTDOWN
CONTINUE	KILL	SOME
CONVERT	LEFT	STATISTICS
CREATE	LIKE	SYSTEM_USER
CROSS	LINENO	TABLE
CURRENT	LOAD	TABLESAMPLE
CURRENT_DATE	MERGE	TEXTSIZE
CURRENT_TIME	NATIONAL	THEN
CURRENT_TIMESTAMP	NOCHECK	TO
CURRENT_USER	NONCLUSTERED	TOP
CURSOR	NOT	TRAN
DATABASE	NULL	TRANSACTION
DBCC	NULLIF	TRIGGER

DEALLOCATE	OF	TRUNCATE
DECLARE	OFF	TRY_CONVERT
DEFAULT	OFFSETS	TSEQUAL
DELETE	ON	UNION
DENY	OPEN	UNIQUE
DESC	OPENDATASOURCE	UNPIVOT
DISK	OPENQUERY	UPDATE
DISTINCT	OPENROWSET	UPDATETEXT
DISTRIBUTED	OPENXML	USE
DOUBLE	OPTION	USER
DROP	OR	VALUES
DUMP	ORDER	VARYING
ELSE	OUTER	VIEW
END	OVER	WAITFOR
ERRLVL	PERCENT	WHEN
ESCAPE	PIVOT	WHERE
EXCEPT	PLAN	WHILE
EXEC	PRECISION	WITH
EXECUTE	PRIMARY	WITHIN GROUP
EXISTS	PRINT	WRITETEXT
EXIT	PROC	

In addition, the ISO standard defines a list of reserved keywords. Avoid using ISO reserved keywords for object names and identifiers. The ISO reserved keyword list is the same as the ODBC reserved keyword list, which is provided in the next section.

Note: The ISO standard reserved keyword 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 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 2 lists the SQL Server 2014 ODBC reserved keywords.

Table 2: 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	HOURL	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
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 Reserved 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. DBAs 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 3 lists keywords that could be reserved in future SQL Server releases.

Table 3: Future Reserved 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
CASCADE	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	SQL EXCEPTION
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://msdn.microsoft.com/en-us/library/ms189822.aspx) (<http://msdn.microsoft.com/en-us/library/ms189822.aspx>) in SQL Server 2014 Books Online.

Upgrade Tools

There are two main tools available to help identify potential problems before you upgrade to SQL Server 2014: the SQL Server 2014 Upgrade Advisor and the Best Practices Analyzer (BPA). For details about using these upgrade tools, see Chapter 1, "Upgrade Planning and Deployment."

SQL Server 2014 Upgrade Advisor

Perhaps the most useful upgrade tool is the SQL Server 2014 Upgrade Advisor. Upgrade Advisor lets you determine, in a test environment, how an application currently running on SQL Server 2005, 2008, 2008 R2, or 2012 will run on SQL Server 2014. 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 2005, SQL Server 2008, SQL Server 2008 R2, or SQL Server 2012 instances, including T-SQL database code and external scripts, before upgrading.

Note: You can run the SQL Server 2014 Upgrade Advisor only against instances of SQL Server SQL Server 2008, 2008 R2, and 2012. You cannot run it against instances of SQL Server 2005, 2000, or 7.0. For SQL Server 2005, use the SQL Server 2012 Upgrade Advisor.

For more information and download instructions, see [SQL Server 2014 Upgrade Advisor](http://msdn.microsoft.com/en-us/library/ee210467.aspx) (<http://msdn.microsoft.com/en-us/library/ee210467.aspx>).

SQL Server 2014 Best Practices Analyzer

You can use the SQL Server 2012 BPA (this is the latest version of the tool as of October 2014) to help identify bad practices in your T-SQL database code and T-SQL scripts. You should always run BPA on your T-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 are various versions of this tool, so make sure you get the one designed especially for SQL Server 2012. You can download the [SQL Server 2012 BPA](http://www.microsoft.com/en-us/download/details.aspx?id=29302) (<http://www.microsoft.com/en-us/download/details.aspx?id=29302>) from the Microsoft Download Center.

64-Bit Considerations

T-SQL queries are completely compatible between 32-bit and 64-bit editions of SQL Server 2014.

Known Issues and Workarounds

This section helps you focus on some key T-SQL-related known issues regarding upgrading from SQL Server 2005/2008/2008 R2/2012 to SQL Server 2014 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 2014. 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 or the `UPDATE STATISTICS` command to update statistics immediately to ensure optimal performance. For more information, see:

- [sp_updatestats \(Transact-SQL\)](http://msdn.microsoft.com/en-us/library/ms173804(v=sql.120).aspx)
([http://msdn.microsoft.com/en-us/library/ms173804\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms173804(v=sql.120).aspx))
- [UPDATE STATISTICS \(Transact-SQL\)](http://msdn.microsoft.com/en-us/library/ms187348(v=sql.120).aspx)
([http://msdn.microsoft.com/en-us/library/ms187348\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms187348(v=sql.120).aspx))

Upgrading from SQL Server 2005, SQL Server 2008, SQL Server 2008 R2, or SQL Server 2012

This section helps you focus on some key T-SQL-related known issues regarding upgrading from SQL Server 2005, SQL Server 2008, SQL Server 2008 R2, or SQL Server 2012 to SQL Server 2014 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.

In-Place Upgrade

The T-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 the "Post-Upgrade Tasks" section later in this chapter). In addition, external scripts on disk will be unaffected.

Side-by-Side Upgrade

In a side-by-side upgrade, any T-SQL objects stored in the database will be automatically moved to the new server, but their T-SQL code will remain unchanged. You will need to alter or re-create the objects directly to update them as required. In

addition, you might need to move any T-SQL external scripts to a new server or correct references within your database to those scripts.

Post-Upgrade Tasks

After upgrading to SQL Server 2014, work through the following checklist to ensure optimum performance:

- 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.
- Run `DBCC UPDATEUSAGE` to correct any incorrect page or row counts.
- Update statistics by using the `sp_updatestats` stored procedure to ensure that all statistics are up-to-date.
- Update revised database T-SQL objects such as stored procedures and functions.
- Update external T-SQL scripts.
- 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 2014 instance performs compared with your original SQL Server 2005, SQL Server 2008, SQL Server 2008 R2, or SQL Server 2012 instance. You can find a suite of tools for load testing, workload replay, and performance analysis at [RML Utilities for SQL Server](http://blogs.microsoft.co.il/yuval14/2014/04/26/rml-utilities-for-sql-server-2/) (<http://blogs.microsoft.co.il/yuval14/2014/04/26/rml-utilities-for-sql-server-2/>). For additional post-upgrade details, see Chapter 3, "Relational Databases."

Conclusion

Although SQL Server 2014 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 T-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 can help ease your transition from SQL Server 2005, SQL Server 2008, SQL Server 2008 R2, or SQL Server 2012 to SQL Server 2014.

Additional References

For an up-to-date collection of additional references for upgrading T-SQL queries to SQL Server 2014, see the following links:

- [Upgrade to SQL Server 2014](http://technet.microsoft.com/en-us/library/bb677622.aspx)
(<http://technet.microsoft.com/en-us/library/bb677622.aspx>)
- [SQL Server 2014 Web Site](http://www.microsoft.com/sqlserver)
(<http://www.microsoft.com/sqlserver>)
- [Books Online for SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms130214%28v=sql.120%29.aspx)
(<http://msdn.microsoft.com/en-us/library/ms130214%28v=sql.120%29.aspx>)
- [SQL Server Developer Center](http://msdn.microsoft.com/en-us/sqlserver)
(<http://msdn.microsoft.com/en-us/sqlserver>)
- [SQL Server TechCenter](http://technet.microsoft.com/en-us/sqlserver)
(<http://technet.microsoft.com/en-us/sqlserver>)

Chapter 11: Spatial Data

Introduction

SQL Server 2014 includes extensive support for spatial data. Although support for spatial data was introduced in SQL Server 2008, the new spatial features and enhancements since then provide a major milestone in the evolution of SQL Server spatial data support. For information about the features and enhancements, see [Spatial Data \(SQL Server\)](http://msdn.microsoft.com/en-us/library/bb933790.aspx) (<http://msdn.microsoft.com/en-us/library/bb933790.aspx>) in SQL Server 2014 Books Online.

Spatial data support is available for the SQL Server Database Services in relational databases. Spatial data is supported in all editions of SQL Server. In this chapter, we cover the breaking and behavior changes for spatial data that you need to know about when upgrading to SQL Server 2014 from pre-SQL Server 2012 versions.

Preparing to Upgrade

The first step for preparing to upgrade to SQL Server 2014 is to search for features that have been deprecated or discontinued. Then, you need to determine whether any deprecated or discontinued feature will affect your upgrade. You also need to search for changes that might prevent an upgrade (i.e., breaking changes) and changes in feature behavior that might require modifications after the upgrade (i.e., behavior changes).

For a complete list of depreciated features, discontinued features, breaking changes, and behavior changes, see [SQL Server Database Engine Backward Compatibility](http://msdn.microsoft.com/en-us/library/ms143532(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms143532\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143532(v=sql.120).aspx)).

Deprecated Features

There are no server-side spatial features that are deprecated. However, client-side code should use [IGeometrySink110](http://msdn.microsoft.com/en-us/library/microsoft.sqlserver.types.igeometrysink110.aspx) (<http://msdn.microsoft.com/en-us/library/microsoft.sqlserver.types.igeometrysink110.aspx>) and [IGeographySink110](http://msdn.microsoft.com/en-us/library/microsoft.sqlserver.types.igeographysink110.aspx) (<http://msdn.microsoft.com/en-us/library/microsoft.sqlserver.types.igeographysink110.aspx>) instead of [IGeometrySink](http://msdn.microsoft.com/en-us/library/microsoft.sqlserver.types.igeometrysink.aspx) (<http://msdn.microsoft.com/en-us/library/microsoft.sqlserver.types.igeometrysink.aspx>) and [IGeographySink](http://msdn.microsoft.com/en-us/library/microsoft.sqlserver.types.igeographysink.aspx) (<http://msdn.microsoft.com/en-us/library/microsoft.sqlserver.types.igeographysink.aspx>), respectively. This is needed because of circular arcs, which are new spatial objects introduced in SQL Server 2012.

Before you upgrade, see [Deprecated Database Engine Features in SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms143729.aspx) (http://msdn.microsoft.com/en-us/library/ms143729.aspx) for any last-minute changes.

Discontinued Functionality

When this guide was being written, there were no discontinued features for the spatial data type in SQL Server 2014. Before you upgrade, see [Discontinued Database Engine Functionality in SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms144262.aspx) (http://msdn.microsoft.com/en-us/library/ms144262.aspx) for any last-minute changes.

Breaking Changes

Before you upgrade, you need to be aware of several breaking changes if you are upgrading from SQL Server 2008 R2 or earlier to SQL Server 2014. This section will discuss the breaking changes brought about by improved precision in SQL Server 2012, SQL Server 2014, and SQL Azure. We will also discuss the introduction of new type of object named FullGlobe, which was added in SQL Server 2012. There was also an upgrade to the Microsoft.SqlServer.Types.dll assembly. In addition, you should check [Breaking Changes to Database Engine Features in SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms143179.aspx#SQL14) (http://msdn.microsoft.com/en-us/library/ms143179.aspx#SQL14) for any last-minute breaking changes.

Improved Precision

In SQL Server 2012 and SQL Server 2014, all constructions and relations are done with 48 bits of precision compared to 27 bits used in SQL Server 2008 and SQL Server 2008 R2. In addition, SQL Azure has been recently upgraded in all data centers to incorporate the new SQL Server 2012 spatial library. This upgrade, which is known as the "September 2011 Service Release," enables spatial operations in SQL Azure to be identical to those in SQL Server 2012 and SQL Server 2014.

The improved precision affects how individual coordinates (vertices) appear in spatial objects (i.e., rounding). It also affects the computational results produced in different versions of the database server for certain spatial operations.

Some results from spatial operations in SQL Azure (after the September 2011 Service Release), SQL Server 2012, and SQL Server 2014 can differ from the results produced from spatial operations in SQL Server 2008, SQL Server 2008 R2, and SQL Azure (before the September 2011 Service Release). In some cases, these changes will not be noticed and will not affect the results. However, there are potential cases where this might matter.

For example, consider the following coordinate, which was processed using the STUnion() method in SQL Server 2008 but was not involved in the resulting geometry:

Original Vertex Coordinate	➔	New Vertex Coordinate After Computation
82.339026 29.661245		82.339025999885052 29.662144999951124

In SQL Server 2012 and SQL Server 2014 the greater numerical precision assists in the preservation of original coordinates of input points in most cases. Here is the result of the same STUnion() method in SQL Server 2012 or SQL Server 2014:

Original Vertex Coordinate	➔	New Vertex Coordinate After Computation
82.339026 29.661245		82.339026 29.662145

FullGlobe

SQL Server 2012 and SQL Server 2014 support geography objects larger than a logical hemisphere. Restricted to slightly less than a logical hemisphere in SQL Server 2008 and SQL Server 2008 R2, geography features in SQL Server 2012 and SQL Server 2014 can now be as big as an entire globe.

A new type of object named FullGlobe can be constructed or received as a result of an operation. You can also construct objects because the interior for geography objects is defined by the orientation of its rings. There is no difference between exterior and interior rings in the geography data type.

For example, consider the three images in Figure 1. The first image shows an example of a "small hole" in a FullGlobe object. The second image shows a regular polygon. The third image shows the union of the two, yielding a spatial object that covers the Earth with three small holes.

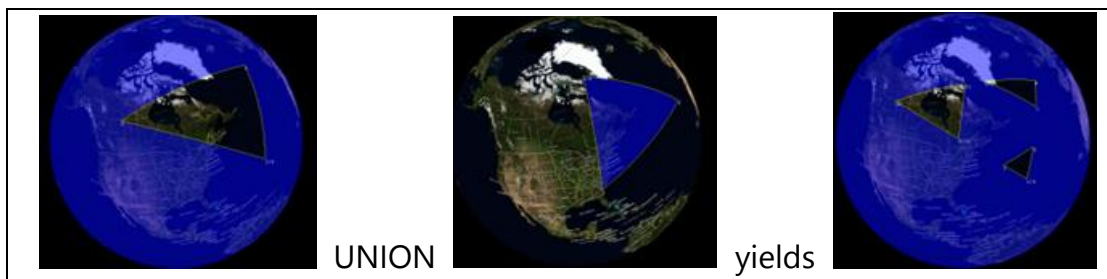


Figure 1: Constructing an object that has small holes

Here is sample Transact-SQL (T-SQL) code that constructs a new FullGlobe object and executes a method on that object:

```
DECLARE @g GEOGRAPHY = GEOGRAPHY::STGeomFromText('FULLGLOBE',4326);
SELECT @g.STArea()-- calculate the area of the WGS84 ellipsoid
--Result: 510,065,621,710,996 meters squared
```

Vertex order is critical for the geography type. In SQL Server 2008 and SQL Server 2008 R2, if you enter an incorrect coordinate order for a geography polygon, you receive the error: *“The specified input does not represent a valid geography instance because it exceeds a single hemisphere. Each geography instance must fit inside a single hemisphere. A common reason for this error is that a polygon has the wrong ring orientation.”*

For the SQL Server geography data type, ring order is defined by the *left-foot rule*. The left-foot rule specifies the interior region of the polygon. (When you “walk” the boundary of a polygon, your left foot is always inside.) Traditional outer ring/inner ring (hole) relationships can be modeled on the closed surface of the globe by using this definition.

Note: The term “left-foot rule” is used in deference to the term “left-hand rule,” which is already in use by physicists and mathematicians to describe phenomena other than polygon ring order.

The following examples demonstrate how the geography type deals with large (greater than a logical hemisphere) objects.

Here is sample T-SQL code that returns a small (less than a logical hemisphere) polygon:

```
DECLARE @R GEOGRAPHY;
SET @R = GEOGRAPHY::Parse('Polygon((-10 -10, 10 -10, 10 10, -10 10,
-10 -10))');
SELECT @R;
```

The resulting object is shown in Figure 2.

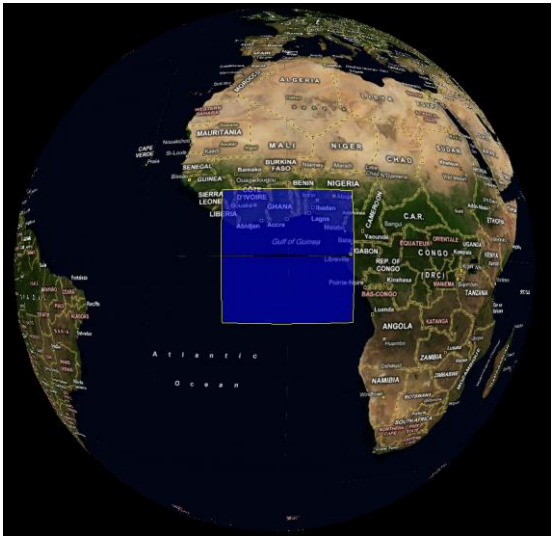


Figure 2: Resulting object is a small (less than a logical hemisphere) polygon

The following T-SQL code changes the coordinate order of the polygon ring to the opposite direction of the previous example:

```
DECLARE @R GEOGRAPHY;
SET @R = GEOGRAPHY::Parse('Polygon((-10 -10, -10 10, 10 10, 10 -10,
-10 -10))');
SELECT @R;
```

The resulting object is now the rest of the globe, as shown in Figure 3.

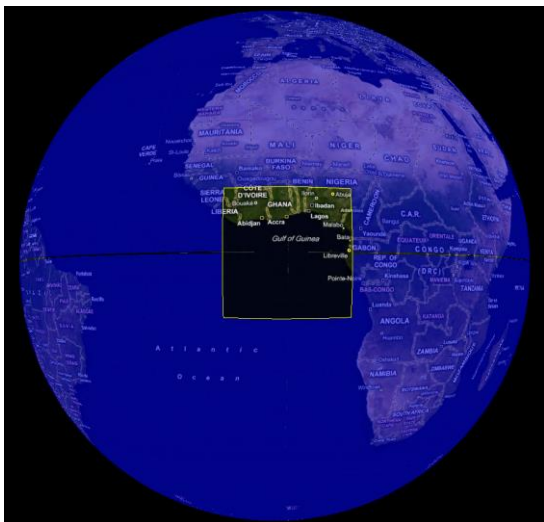


Figure 3: Resulting object is now the rest of the globe

There are several other considerations which need to be taken into account with the FullGlobe enhancements to the geography type:

- FullGlobe cannot be a member of GeometryCollection.

- Well-Known Text (WKT), Well-Known Binary (WKB), and Geography Markup Language (GML) and its schema support FullGlobe objects.
- Antipodal edges are not allowed. For example, LineString(0 90, 0 -90) is considered antipodal, whereas LineString(0 90, 0 0, 0 -90) is not.
- There are two types of arcs in geography: great circle arcs defined with two points and small circle arcs defined with three points.
- Degenerate arcs are treated as great circle arcs between the start and end points for the geography type. (For the geometry type, degenerate arcs are treated as straight edges.)
- EnvelopeAngle() will return 180 for objects larger than a logical hemisphere and less than 90 for objects smaller than a logical hemisphere.

Upgrade to the Microsoft.SqlServer.Types.dll Assembly

The Microsoft.SqlServer.Types.dll assembly has been upgraded from version 11.0 to version 12.0. You need to be careful with custom applications that have references to this assembly. If you have custom applications that use the SQL CLR data types (geometry, geography, and hierarchyid), you will need to make changes in them. You can get an error when the following conditions are true as shown in the following example, where the code references version 10.0 of this library:

1. When you move a custom application from a computer on which SQL Server 2008 R2 is installed to a computer on which only SQL Server 2014 is installed, the application will fail because the referenced version 10.0 of the SqlTypes assembly is not present. You can get this error message: *"Could not load file or assembly 'Microsoft.SqlServer.Types, Version=10.0.0.0, Culture=neutral, PublicKeyToken=89845dcd8080cc91' or one of its dependencies. The system cannot find the file specified."*
2. When you reference the SqlTypes assembly version 11.0 and version 10.0 is also installed, you can get the following error message: *"System.InvalidCastException: Unable to cast object of type 'Microsoft.SqlServer.Types.SqlGeometry' to type 'Microsoft.SqlServer.Types.SqlGeometry'."*
3. When you reference the SqlTypes assembly version 12.0 from a custom application that uses the .NET Framework 3.5, 4.0, or 4.5, your application will fail because SqlClient by design loads version 10.0 of the assembly. This failure occurs when your application calls one of the following methods:

- GetValue method of the SqlDataReader class
- GetValues method of the SqlDataReader class
- Bracket index operator [] of the SqlDataReader class
- ExecuteScalar method of the SqlCommand class

You can work around this issue by calling the [GetSqlBytes method](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldatareader.getsqlbytes.aspx) (http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldatareader.getsqlbytes.aspx) instead of the methods just listed. For more information about how to use the GetSqlBytes method, see the "SQL CLR Data Types (geometry, geography, and hierarchyid)" section in [Breaking Changes to Database Engine Features in SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms143179.aspx#SQL14) (http://msdn.microsoft.com/en-us/library/ms143179.aspx#SQL14).

Behavior Changes

You need to be aware of several behavior changes before upgrading to SQL Server 2014 from SQL Server 2008 R2 or earlier. This section will discuss the behavior changes resulting from enhancements to the STEnvelope method and the geography data type in SQL Server 2012 and SQL Server 2014. In addition, you should check [Behavior Changes to Database Engine Features in SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms143359.aspx) (http://msdn.microsoft.com/en-us/library/ms143359.aspx) for any last-minute behavior changes.

STEnvelope Method

In SQL Server 2012 and SQL Server 2014, the STEnvelope method is consistent with the behavior of other SQL Server Spatial methods.

If you call the STEnvelope method in SQL Server 2008 and SQL Server 2008 R2, you get the following results:

```
select geometry::Parse('POINT EMPTY').STEnvelope().ToString()
```

Result: POINT EMPTY

```
select geometry::Parse('LINESTRING EMPTY').STEnvelope().ToString()
```

Result: LINESTRING EMPTY

```
select geometry::Parse('POLYGON EMPTY').STEnvelope().ToString()
```

Result: POLYGON EMPTY

When you run the same Transact-SQL (T-SQL) code in SQL Server 2012 and SQL Server 2014, the results will be the following:

```
select geometry::Parse('POINT EMPTY').STEnvelope().ToString()
```

Result: GEOMETRYCOLLECTION EMPTY

```
select geometry::Parse('LINESTRING EMPTY').STEnvelope().ToString()
```

Result: GEOMETRYCOLLECTION EMPTY

```
select geometry::Parse('POLYGON EMPTY').STEnvelope().ToString()
```

Result: GEOMETRYCOLLECTION EMPTY

The best practice for determining when a spatial object is empty is to call the `STIsEmpty` (geometry data type) method. For more information about this method and others, review [geometry Data Type Method Reference](http://msdn.microsoft.com/en-us/library/bb933973(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/bb933973\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/bb933973(v=sql.120).aspx)).

Note: For more information about empty spatial types, see “Behavior of `STEnvelope()` Method Has Changed with Empty Spatial Types” in [Behavior Changes to Database Engine Features in SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms143359(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms143359\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143359(v=sql.120).aspx)).

Geography Data Type

In SQL Server 2012 and SQL Server 2014, the geography data type supports objects that are bigger than a hemisphere. In SQL Server 2008 R2 and earlier, the following methods will return NULL if the result is such object, but in SQL Server 2012 and SQL Server 2014, they will return the expected result. However, they are all controllable by DB COMPATIBILITY LEVEL and will continue to return NULL if the compatibility level is set to 100.

- Geography `STBuffer`(double distance): Returns a geometric object representing the union of all points whose distance from this instance is less than or equal to *distance*.
- Geography `BufferWithTolerance`(double distance, double tolerance, bool relative): This is equivalent to `STBuffer()` but allows for a user-defined tolerance.
- Geography `STUnion`(Geography other): Returns a geometric object representing the points in the union of this instance with *other*.
- Geography `STIntersection`(Geography other): Returns a geometric object representing the points in the intersection of this instance with *other*.

- Geography STDifference(Geography other): Returns a geometric object representing the points that are in this instance and not in *other*.
- Geography STSymDifference(Geography other): Returns a geometric object representing the points that are either in this instance or in *other*, but not in both. This method returns null if the spatial reference identifier (SRID) of this instance does not match the SRID of *other*. This method is not precise.

In addition, a new spatial type is added, called FullGlobe (represents entire globe), which may be returned by all the methods specified above or by methods that return a binary or text representation of a geography object, such as ToString() and STAsText().

Post-Upgrade Tasks

After you complete the upgrade to SQL Server 2014, review and make changes in your code to get the best performance from the new spatial data features.

For information on upgrading to SQL Server 2014, see Chapter 1, “Upgrade Planning and Deployment,” and Chapter 3, “Relational Databases,” in this guide. For information about the new spatial data features, see [Spatial Data \(SQL Server\)](http://msdn.microsoft.com/en-us/library/bb933790(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/bb933790\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/bb933790(v=sql.120).aspx)) in SQL Server 2014 Books Online.

Conclusion

When you upgrade to SQL Server 2014, the effort for spatial data is very minimal, especially if you are upgrading from SQL Server 2012. Just be sure to check for behavior changes, especially with the STEnvelope method.

The new spatial features and improvements introduced in SQL Server 2012 provide a major milestone in the evolution of SQL Server spatial data support. The ability to support full globe spatial objects and circular arcs on the ellipsoid are industry firsts for relational database systems.

After you have your databases upgraded to SQL Server 2014, all of these new features and improvements will be available to you.

Additional References

- [Spatial Data \(SQL Server\)](http://msdn.microsoft.com/en-us/library/bb933790(v=sql.120).aspx)
([http://msdn.microsoft.com/en-us/library/bb933790\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/bb933790(v=sql.120).aspx))

- [SQL Server TechNet Resources](http://technet.microsoft.com/en-us/sqlserver)
(<http://technet.microsoft.com/en-us/sqlserver>)
- [SQL Server Developer Center](http://msdn.microsoft.com/en-us/sqlserver)
(<http://msdn.microsoft.com/en-us/sqlserver>)

Chapter 12: XML and XQuery

Introduction

SQL Server 2014 includes extensive support for XML. This support includes creating XML from relational data with a query and shredding XML into relational tabular format. In addition, SQL Server has a native XML data type. You can store XML data, constrain it with XML schemas, index it with specialized XML indexes, and manipulate it using XML data type methods. All of T-SQL's XML data type methods accept an XQuery string as a parameter. XQuery (for XML Query Language) is the standard language used to query and manipulate XML data.

XML support is available in SQL Server Database Services in the relational database. The process of upgrading relational databases in SQL Server is covered in other chapters, such as Chapter 3, "Relational Databases," and Chapter 1, "Upgrade Planning and Deployment." Therefore, we are covering only changes to XML support inside SQL Server in this chapter. In addition, you do not have to worry about XML support in different editions of SQL Server; all editions fully support XML features.

Preparing to Upgrade

Before upgrading to SQL Server 2014, you should investigate which features are deprecated. These features will not affect your upgrade, but you might need to change your applications and queries. You also need to know what functionality cannot be upgraded because it is discontinued or because it has changed in SQL Server 2014. Plus, you need to be aware of some behavior changes in XML support between SQL Server 2005, 2008, 2008 R2, 2012, and 2014; otherwise, you could get unexpected results. Let's look at each of these categories of changes. For a complete reference of XML changes in SQL Server 2014, see [SQL Server Database Engine Backward Compatibility](http://msdn.microsoft.com/en-us/library/ms143532.aspx) (<http://msdn.microsoft.com/en-us/library/ms143532.aspx>) in SQL Server 2014 Books Online.

Deprecated Features

You can validate XML documents against a schema. An old XML schema language is XML Data Reduced (XDR) language. You can create XDR documents with the XMLDATA directive in the FOR XML clause of the SELECT T-SQL statement. The current validation language is called XML Schema, which is stored in XML Schema Document (XSD) XML files. SQL Server 2005 supports XSD generation. In SQL Server, you can validate XML data type values against XSD with XML Schema collections.

In SQL Server 2014, the XMLDATA directive to the FOR XML option is deprecated. Use the XMLSCHEMA directive to generate XSD schemas in RAW and AUTO modes. There is no replacement for the XMLDATA directive in EXPLICIT mode. This option is going to be still available in the next version of SQL Server; it is not going to be supported in one of the future versions. However, because there is not much reason to use XDR schemas inside SQL Server, you should start changing your code soon. For example, the following code, executed in the context of the AdventureWorksDW2014 database, generates XDR schema:

```
SELECT C.CustomerKey,
       C.FirstName + ' ' + C.LastName AS CustomerName,
       I.SalesAmount
FROM   dbo.DimCustomer AS C
       INNER JOIN dbo.FactInternetSales AS I
       ON C.CustomerKey = I.CustomerKey
WHERE  1 = 0
FOR XML AUTO, XMLDATA;
```

Note that AdventureWorksDW2014 is the 2014 version of the AdventureWorks demo data warehouse.

The following code generates XSD instead of XDR schema:

```
SELECT C.CustomerKey,
       C.FirstName + ' ' + C.LastName AS CustomerName,
       I.SalesAmount
FROM   dbo.DimCustomer AS C
       INNER JOIN dbo.FactInternetSales AS I
       ON C.CustomerKey = I.CustomerKey
WHERE  1 = 0
FOR XML AUTO, XMLSCHEMA;
```

Discontinued Functionality

There is no discontinued functionality for XML and XQuery support in SQL Server 2014. So, you do not have to worry about this.

Breaking Changes

In SQL Server 2014, XQuery string functions are surrogate pair-aware. SQL Server Unicode data types, including NCHAR, NVARCHAR, and XML, encode text in UTF-16 format. SQL Server allocates for each character a unique codepoint, a value in the range 0x0000 to 0x10FFFF. Most of the characters fit into a 16-bit word. Characters with codepoint values larger than 0xFFFF require two consecutive 16-bit words (i.e., two bytes). These characters are called supplementary characters, and the two consecutive 16-bit words are called surrogate pairs.

The standard World Wide Web Consortium (W3C) recommendation for XQuery functions and operators requires them to count a surrogate pair as a single character. In SQL Server versions prior to 2012, XQuery string functions did not recognize surrogate pairs as a single character. For example, string length calculations returned incorrect results.

The XML data type in SQL Server only allows well-formed surrogate pairs. However, it is possible to pass invalid or partial surrogate pairs to XQuery functions as string values. Below is an example of providing surrogate pairs to XML data type variable through string values. The value of the CustomerName element is “佻佻佻12”. The first three characters of the name are surrogate pairs. The length of this value is 5. However, in versions before SQL Server 2012, you would get a length of 8.

If you execute the following code in SQL Server 2005, 2008, or 2008 R2

```
DECLARE @x AS NVARCHAR(MAX), @y AS XML;
SET @x=
N'<C11003>
  <CustomerKey>11003</CustomerKey>
  <CustomerName>佻佻佻12</CustomerName>
  <SalesAmount>2294.9900</SalesAmount>
</C11003>';
SET @y = CAST(@x AS XML);
SELECT @y.query('
for $i in /C11003
return
  <NumberOfCharacters>
    { string-length($i/CustomerName[1]) }
  </NumberOfCharacters>
');
```

you get this result:

```
<NumberOfCharacters>8</NumberOfCharacters>
```

The same code executed in SQL Server 2012 and SQL Server 2014 returns:

```
<NumberOfCharacters>5</NumberOfCharacters>
```

This changed behavior affects the following XQuery functions, operators, and clauses:

- fn:string-length
- fn:substring
- fn:contains (only when the compatibility level is 110 or higher)
- fn:concat (only when the compatibility level is 110 or higher)

- Comparison operators including +, <, >, <=, >=, eq, lt, gt, le, and ge (only when the compatibility level is 110 or higher)
- Order by clause (only when the compatibility level is 110 or higher).

If you are upgrading from SQL Server 2005, you should also consider breaking changes in SQL Server 2008 and SQL Server 2008 R2.

In SQL Server 2005, the data types `xs:time`, `xs:date`, and `xs:dateTime` do not allow time zone support—data is always mapped to the Coordinated Universal Time (UTC) time zone. Starting with version 2008, SQL Server provides standard conformant behavior. This behavior includes:

- Values without time zones are validated.
- The time zone (or the absence of it) is preserved.
- The internal storage representation is modified. This does not have any influence on your applications.
- Resolution of stored values is increased.
- Negative years are disallowed.

The `sqltypes` schema namespace has been extended to include the date and time SQL Server data types introduced in SQL Server 2008, including `TIME`, `DATE`, `DATETIME2`, and `DATETIMEOFFSET`. You can also use the XML data type `value()` method to cast XML elements and attributes to the new SQL Server date and time data types. The following code does not run on SQL Server 2005; however, it runs on SQL Server 2008 and later. It shows usage of the new date and time data types.

```

DECLARE @myDoc AS XML;
DECLARE @OrderID AS INT;
DECLARE @OrderDate AS DATE;
DECLARE @OrderTime AS TIME;
DECLARE @OrderDateTime AS DATETIMEOFFSET;
SET @myDoc = '
<Root>
  <OrderDescription
    OrderID="1" OrderDate="2012-02-10"
    OrderTime="13:40:58.47786" OrderDateTime="1999-12-20 13:40:58.123-05:00">
  </OrderDescription>
</Root>';
SET @OrderID =
  @myDoc.value (' (/Root/OrderDescription/@OrderID) [1]', 'int');
SET @OrderDate =
  @myDoc.value (' (/Root/OrderDescription/@OrderDate) [1]', 'date');
SET @OrderTime =

```

```

    @myDoc.value (' (/Root/OrderDescription/@OrderTime) [1]', 'time');
SET @OrderDateTime =
    @myDoc.value (' (/Root/OrderDescription/@OrderDateTime) [1]',
'datetimeoffset');
SELECT @OrderID, @OrderDate, @OrderTime, @OrderDateTime;

```

Finally, in SQL Server 2005, steps in an XQuery or XML Path Language (XPath) expression that begin with a colon are allowed. This behavior does not conform to XML standards and is disallowed in all versions after SQL Server 2005. For example, the following code works on SQL Server 2005:

```

DECLARE @x AS XML;
SET @x=
N'<C11003>
  <CustomerKey>11003</CustomerKey>
  <CustomerName>Customer 1</CustomerName>
  <SalesAmount>2294.9900</SalesAmount>
</C11003>';
SELECT @x.query('
for $i in /C11003
return $i/:CustomerName[1]
');

```

However, this code does not work on SQL Server 2008 or later. You should remove the colon for these versions.

Behavior Changes

The XML data type value() method has changed internally in SQL Server 2012 and SQL Server 2014. In previous versions, the value method internally converted the source value to an xs:string, and then converted the xs:string to the target SQL Server data type. In SQL Server 2012 and later, the conversion to xs:string is skipped in some cases. This means that some queries are slightly faster. This conversion is skipped if:

- The source XML data type is byte, short, int, integer, long, unsignedByte, unsignedShort, unsignedInt, unsignedLong, positiveInteger, nonPositiveInteger, negativeInteger, or nonNegativeInteger, and the destination SQL data type is TINYINT, SMALLINT, INT, BIGINT, DECIMAL, or NUMERIC.
- The source XML data type is decimal, and the destination SQL data type is DECIMAL or NUMERIC.
- The source XML data type is float, and the destination SQL data type is REAL.
- The source XML data type is double, and the destination SQL data type is FLOAT.

In addition, if the XML data type exist() function returns NULL and is used in

comparison with 0, the comparison returns NULL in SQL Server 2012 and SQL Server 2014. Such a comparison returned TRUE in previous versions. Consider the following code:

```
DECLARE @test AS XML;
SET @test = NULL;
-- Returns 0 in SQL Server 2012 and 1 in earlier editions
SELECT COUNT(1) WHERE @test.exist('/whatever') = 0;
-- Returns 1 in all editions
SELECT COUNT(1) WHERE @test.exist('/whatever') IS NULL;
```

The first query returns 0 in SQL Server 2012 and SQL Server 2014, whereas it returns 1 in SQL Server 2008 R2 and earlier editions.

Post-Upgrade Tasks

After you upgrade SQL Server Database Services to version 2014, you should consider revising your code in order to benefit from the new features. There are not many new features regarding XML and XQuery in SQL Server 2014. However, some new features were added in SQL Server 2008. Therefore, if you are upgrading from SQL Server 2005, you might decide to make more code changes than when upgrading from SQL Server 2008 or SQL Server 2008 R2.

Features Added in SQL Server 2012

If you are upgrading from SQL Server 2008 R2 or earlier, you should also consider new XML and XQuery features added in SQL Server 2012. SQL Server 2012 adds support for supplementary characters in UTF-16 Unicode collation in the SQL Types XML schema and in other locations where SQL Server uses XML context. The XML schema for the [SQL to XSD types mapping](http://schemas.microsoft.com/sqlserver/2004/sqltypes) (http://schemas.microsoft.com/sqlserver/2004/sqltypes), which is version 1.2 in SQL Server 2012, exposes the supplementaryCharacters global attribute. In addition, the new global attribute is exposed in the schemas you generate with the XMLSCHEMA directive. Note that this directive is supported with FOR XML in RAW and AUTO modes; it is not supported in EXPLICIT or PATH modes. This new global attribute is exposed in the following catalog views:

- sys.xml_schema_components
- sys.xml_schema_attributes
- sys.xml_schema_component_placements

The following three queries return one row each in SQL Server 2012 and SQL Server 2014 and zero rows in previous versions:

```

SELECT xml_component_id, name
  FROM sys.xml_schema_components
  WHERE name = N'supplementaryCharacters';
SELECT xml_component_id, name
  FROM sys.xml_schema_attributes
  WHERE name = N'supplementaryCharacters';
SELECT P.xml_component_id, A.name
  FROM sys.xml_schema_component_placements AS P
  INNER JOIN sys.xml_schema_attributes AS A
  ON P.xml_component_id = A.xml_component_id
  WHERE P.xml_component_id = 373;

```

If you validate your XML data type values against an XML schema collection, the schema collection is upgraded when you upgrade your database as well. After the upgrade, the new supplementaryCharacters global attribute is exposed in your upgraded XML schema collection if the schemas in the collection import the SQL Types XML schema. Finally, the new supplementaryCharacters global attribute is added to XML column set values that represent UTF-16 collation sql_variant strings.

Features Added in SQL Server 2008

If you are upgrading from SQL Server 2005, you should consider new XML and XQuery features added in SQL Server 2008. The first feature we need to mention is relaxed (Lax in XML terminology) schema validation. Besides allowing the nodes defined in the schema that an XML instance is validated against, you can also allow additional elements and attributes in your XML instance that are not defined in your schema. This enables you to only partially validate an XML instance. In short, you enforce validation for elements that you can associate schema with them, and ignore elements for which you do not have or cannot control the schema.

From SQL Server 2008, you can define XML data types in your XML schemas that allow a limited set of values for multi-valued elements and attributes in your XML instances. For example, you can create an XML list type for product sizes, and then enumerate values like 48, 50, 52, and 54. Now suppose that you need to add values like S, M, L, and XL to the list. In the original list, the value type is integer; now you want to add strings. From SQL Server 2008, you can create union XML data types—types that can combine lists of enumerated elements of different data types.

The real power of XQuery lies in so-called FLWOR expressions. FLWOR is the acronym for *for, let, where, order by, and return*. A FLWOR expression is actually a *for each* loop. You can use it to iterate through a sequence returned by an XPath expression. Although you typically iterate through a sequence of nodes, you can use FLWOR expressions to iterate through any sequence. You can limit the nodes to be processed

with a predicate, sort the nodes, and format the returned XML. The parts of a FLWOR statement are:

- **For.** With the *for* clause, you can bind iterator variables to input sequences. Input sequences are either sequences of nodes or sequences of atomic values. You can create atomic value sequences using literals or functions.
- **Let.** With the optional *let* clause, you can assign a value to a variable for a specific iteration. The expression used for assignment can return a sequence of nodes or a sequence of atomic values.
- **Where.** With the optional *where* clause, you can filter the iteration.
- **Order by.** Using the *order by* clause, you can control the order in which the elements of the input sequence are processed. You control the order based on atomic values.
- **Return.** The return clause is evaluated once for each iteration, and the results are returned to the client in the iteration order.

In SQL Server 2005, the *let* clause is not supported. Consider the following example:

```
DECLARE @x AS XML;
SET @x = N'
<CustomersOrders>
  <Customer custid="1">
    <!-- Comment 111 -->
    <companyname>Customer NRZBB</companyname>
    <Order orderid="10692">
      <orderdate>2007-10-03T00:00:00</orderdate>
    </Order>
    <Order orderid="10702">
      <orderdate>2007-10-13T00:00:00</orderdate>
    </Order>
    <Order orderid="10952">
      <orderdate>2008-03-16T00:00:00</orderdate>
    </Order>
  </Customer>
  <Customer custid="2">
    <!-- Comment 222 -->
    <companyname>Customer MLTDN</companyname>
    <Order orderid="10308">
      <orderdate>2006-09-18T00:00:00</orderdate>
    </Order>
    <Order orderid="10952">
      <orderdate>2008-03-04T00:00:00</orderdate>
    </Order>
  </Customer>
</CustomersOrders>';
```

```

-- The following query works in SQL Server 2005 and later
SELECT @x.query('for $i in CustomersOrders/Customer/Order
  where $i/@orderid < 10900
  order by ($i/orderdate)[1]
  return
  <Order-orderid-element>
  <orderid>{data($i/@orderid)}</orderid>
  {$i/orderdate}
  </Order-orderid-element>')
  AS [1. Filtered, sorted and reformatted orders];
-- The following query does not work in SQL Server 2005
SELECT @x.query('for $i in CustomersOrders/Customer/Order
  let $j := $i/orderdate
  where $i/@orderid < 10900
  order by ($j)[1]
  return
  <Order-orderid-element>
  <orderid>{data($i/@orderid)}</orderid>
  {$j}
  </Order-orderid-element>')
  AS [2. Filtered, sorted and reformatted orders with let clause];

```

Both queries return the same result. In the *for* loop, the query iterates through all Order nodes using an iterator variable and returns those nodes. The resulting XML instance is filtered by the orderid attribute. The orderid attribute is converted to an element by creating the element manually and extracting only the value of the attribute with the data() function. The orderdate element is returned as well, both wrapped in the Order-orderid-element element. Note the braces around the expressions that extract the value of the orderid element and the orderdate element. XQuery evaluates expressions in braces; without braces, everything would be treated as a string literal and returned as such. The result is ordered by the orderdate element.

Note that in the first query, the orderdate element of the \$i iterator variable is referred twice, once in the order by clause and once in the return clause. You can spare some typing if you use the *let* clause to assign a name to the repeating expression. To name the expression, you have to use a variable different from \$i (e.g., \$j) in the second query.

Conclusion

You probably won't decide to upgrade to SQL Server 2014 only because of the new XML and XQuery features. However, there are many other reasons for the upgrade. And once you have your database upgraded, you can start exploiting XML and XQuery improvements as well.

Additional References

- [SQL Server 2014 Web Site](http://www.microsoft.com/en-us/server-cloud/products/sql-server/)
(<http://www.microsoft.com/en-us/server-cloud/products/sql-server/>)
- [SQL Server TechNet Resources](http://technet.microsoft.com/en-us/sqlserver)
(<http://technet.microsoft.com/en-us/sqlserver>)
- [SQL Server Developer Center](http://msdn.microsoft.com/en-us/sqlserver)
(<http://msdn.microsoft.com/en-us/sqlserver>)

Chapter 13: CLR

Introduction

The Common Language Runtime (CLR) is the heart of the Microsoft .NET Framework and provides the execution environment for all .NET Framework code. Code that runs within the CLR is referred to as managed code. The CLR provides various functions and services required for program execution, including just-in-time (JIT) compilation, allocating and managing memory, enforcing type safety, exception handling, thread management, and security.

With the CLR hosted in Microsoft SQL Server (called CLR integration or SQL CLR), you can author stored procedures, triggers, user-defined functions, user-defined types, and user-defined aggregates in managed code (both C# and VB.NET). Because managed code compiles to native code prior to execution, you can achieve significant performance increases in some scenarios.

SQL CLR was introduced with SQL Server 2005. Integration of the CLR in SQL Server also meant that SQL Server itself hosts the .NET Framework Runtime. Memory, threading, deadlock detection, and resolution services for .NET code are managed by the SQL Server Operating System (SQLOS) itself rather than the underlying Windows operating system.

This chapter will lead you through the CLR upgrade process.

Preparing to Upgrade

To prepare for the CLR upgrade process, you need review the changes that could block your upgrade or change the behavior of your applications after the upgrade.

Breaking Changes

Breaking changes are those changes that introduce the possibility of blocking an upgrade. There are no breaking changes between the SQL Server 2012 CLR and the SQL Server 2014 CLR. However, there are breaking changes that could block upgrades from SQL Server 2008 R2 or earlier to SQL Server 2014.

The .NET Framework

(Only Applicable When Upgrading from SQL Server 2008 R2 or Earlier)

The SQL Server 2014 CLR is based on the .NET Framework 4, which replaces the .NET Framework 2.0 used by SQL Server 2008 R2 and older versions. This means

compatibility issues between the previous versions of the .NET Framework (1.0, 2.0, 3.0, 3.5, and 3.5 SP1) and the .NET Framework 4 are also applicable to SQL CLR assemblies when upgrading from SQL Server 2008 R2 or older. For details on what's new in the .NET Framework 4, see:

- [What's New in the .NET Framework 4](http://msdn.microsoft.com/en-us/library/ms171868(v=vs.100).aspx)
([http://msdn.microsoft.com/en-us/library/ms171868\(v=vs.100\).aspx](http://msdn.microsoft.com/en-us/library/ms171868(v=vs.100).aspx))
- [Migration Guide to the .NET Framework 4](http://msdn.microsoft.com/en-us/library/ff657133(v=vs.100).aspx)
([http://msdn.microsoft.com/en-us/library/ff657133\(v=vs.100\).aspx](http://msdn.microsoft.com/en-us/library/ff657133(v=vs.100).aspx))

Note: As far as the upgrade to SQL Server 2014 is concerned, the .NET Framework is part of the installation process.

Behavior Changes

Behavioral changes are those changes that do not block the upgrade process but you need to be aware of them as they might affect the existing CLR objects. There are almost no behavior changes between the SQL Server 2012 CLR and the SQL Server 2014 CLR. However, there are a few behavior changes when upgrading from SQL Server 2008 R2 or earlier to SQL Server 2014 that could affect existing CLR objects the way the .NET Framework 4 works.

Managed Code Can No Longer Catch Exceptions from Corrupted Process States (Only Applicable When Upgrading from SQL Server 2008 R2 or earlier)

In version 4 of the CLR, CLR database objects no longer catch corrupted state exceptions (e.g., access violations) in catch blocks. These exceptions are now caught in the CLR integration hosting layer. The `<legacyCorruptedStateExceptionsPolicy>` setting in the .NET Framework 4 that would enable applications to catch exceptions for corrupted process states is not available in SQL Server 2014, so the only method to catch these exceptions is to apply the `System.Runtime.ExceptionServices.HandleProcessCorruptedStateExceptionsAttribute` to the method that contains the exceptions catch block. For details, see [HandleProcessCorruptedStateExceptionsAttribute Class](http://msdn.microsoft.com/en-us/library/system.runtime.exception.services.handleprocesscorruptedstateexceptionsattribute(v=vs.100).aspx) ([http://msdn.microsoft.com/en-us/library/system.runtime.exception.services.handleprocesscorruptedstateexceptionsattribute\(v=vs.100\).aspx](http://msdn.microsoft.com/en-us/library/system.runtime.exception.services.handleprocesscorruptedstateexceptionsattribute(v=vs.100).aspx)).

Here is an example of applying this attribute:

```

using System.Runtime.ExceptionServices

/*
Individual managed methods can override the defaults and catch these
exceptions by applying the HandleProcessCorruptedStateExceptions attribute to
the method.
*/

[HandleProcessCorruptedStateExceptions]
public static void HandleCorruptedState()
{
    // Write Exception Notification code here.
    // Can log or send mail to admin, etc.
}

```

Unsupported Format String with TimeSpan Throws an Error (Only Applicable When Upgrading from SQL Server 2008 R2 or Earlier)

In the .NET Framework 4, the System.TimeSpan structure implements the [IFormattable interface](http://msdn.microsoft.com/en-us/library/system.iformattable(v=vs.100).aspx) and supports formatting operations with standard and custom format strings. This means that if a parsing method supplies an unsupported format specifier or format string, a System.FormatException will be thrown. In the previous versions of the .NET Framework, the System.TimeSpan structure does not implement the IFormattable interface and does not support format strings. However, many developers mistakenly assumed that [TimeSpan](http://msdn.microsoft.com/en-us/library/system.timespan(v=vs.100).aspx) supported a set of format strings and used them in [composite formatting operations](http://msdn.microsoft.com/en-us/library/txafckwd(v=vs.100).aspx) with methods such as [String.Format](http://msdn.microsoft.com/en-us/library/system.string.format(v=vs.100).aspx) (http://msdn.microsoft.com/en-us/library/system.string.format(v=vs.100).aspx). The runtime ignored the format string and instead called the [TimeSpan.ToString method](http://msdn.microsoft.com/en-us/library/1ecy8h51(v=vs.100).aspx) (http://msdn.microsoft.com/en-us/library/1ecy8h51(v=vs.100).aspx). This means that the format strings had no effect on the formatting operations but they did not result in an exception. This legacy behavior can be preserved by setting the <TimeSpan_LegacyFormatMode> element's enabled attribute to true in the application's configuration file. For details, see [<TimeSpan_LegacyFormatMode> Element](http://msdn.microsoft.com/en-us/library/ee803802(v=vs.100).aspx) (http://msdn.microsoft.com/en-us/library/ee803802(v=vs.100).aspx).

Because of the special hosting considerations of SQL CLR compared to CLR itself, changing the application configuration file is not possible because there is no sqlservr.exe.config file for SQL CLR. The only way SQL Server 2014 allows you to enable <TimeSpan_LegacyFormatMode> is by changing the database compatibility level (dbcmtplevel) to 100 (which is the database compatibility level for SQL Server 2008) for

new databases created on SQL Server 2014. Databases upgraded from SQL Server 2008 or SQL Server 2005 keep their respective database compatibility levels and require no change. For information on how to change the database compatibility level, see [ALTER DATABASE Compatibility Level \(Transact-SQL\)](http://msdn.microsoft.com/en-us/library/bb510680.aspx) (http://msdn.microsoft.com/en-us/library/bb510680.aspx).

String Sorting Conforms to the Unicode 5.1 Standard (Only Applicable When Upgrading from SQL Server 2008 R2 or Earlier)

In the .NET Framework 4, string comparison, sorting, and casing operations performed by the [System.Globalization.CompareInfo class](http://msdn.microsoft.com/en-us/library/system.globalization.compareinfo(v=vs.100).aspx) (http://msdn.microsoft.com/en-us/library/system.globalization.compareinfo(v=vs.100).aspx) now conform to the Unicode 5.1 standard. This means that the results of string comparison methods such as [String.Compare\(String, String\)](http://msdn.microsoft.com/en-us/library/84787k22(v=vs.100).aspx) (http://msdn.microsoft.com/en-us/library/84787k22(v=vs.100).aspx) and [String.LastIndexOf\(String\)](http://msdn.microsoft.com/en-us/library/1wdsy8fy(v=vs.100).aspx) (http://msdn.microsoft.com/en-us/library/1wdsy8fy(v=vs.100).aspx) may differ from previous versions of the .NET Framework. If an application depends on legacy behavior, you can restore the string comparison and sorting rules used in the .NET Framework 3.5 and earlier versions by setting the <CompatSortNLSVersion> element's enabled attribute to 4096 (which is the locale ID representing sort order of the .NET Framework 3.5 and earlier versions) in your application's configuration file. For details, see [<CompatSortNLSVersion> Element](http://msdn.microsoft.com/en-us/library/ff469779(v=vs.100).aspx) (http://msdn.microsoft.com/en-us/library/ff469779(v=vs.100).aspx).

Because of the special hosting considerations of SQL CLR compared to CLR itself, changing the application configuration file is not possible because there is no sqlservr.exe.config file for SQL CLR. The only way SQL Server 2014 allows you to enable <CompatSortNLSVersion> is by changing the database compatibility level (dbcmtlevel) to 100 (which is the database compatibility level for SQL Server 2008) for new databases created on SQL Server 2014. Databases upgraded from SQL Server 2008 or SQL Server 2005 keep their respective database compatibility levels and require no change. For information on how to change the database compatibility level, see [ALTER DATABASE Compatibility Level \(Transact-SQL\)](http://msdn.microsoft.com/en-us/library/bb510680.aspx) (http://msdn.microsoft.com/en-us/library/bb510680.aspx).

SQL CLR Data Types (geometry, geography, hierarchyid)

The assembly Microsoft.SqlServer.Types.dll, which contains the spatial data types and the hierarchyid type, has been upgraded to version 12.0. Custom applications that reference this assembly may fail when the following conditions are true:

- When you move a custom application from a computer on which SQL Server 2012 or earlier was installed to a computer on which only SQL Server 2014 is installed, the application will fail because the referenced version (11.0 or 10.0) of the SqlTypes assembly is not present. You may see this error message: *"Could not load file or assembly 'Microsoft.SqlServer.Types, Version=10.0.0.0, Culture=neutral, PublicKeyToken=89845dcd8080cc91' or one of its dependencies. The system cannot find the file specified."*
- When you reference the SqlTypes assembly version 12.0 and version 10.0 is also installed, you may see this error message: *"System.InvalidCastException: Unable to cast object of type 'Microsoft.SqlServer.Types.SqlGeometry' to type 'Microsoft.SqlServer.Types.SqlGeometry'."*
- When you reference the SqlTypes assembly version 12.0 from a custom application that targets the .NET Framework 4, the application will fail because SqlClient by design loads version 10.0 of the assembly. This failure occurs when the application calls one of the following methods:
 - GetValue method of the SqlDataReader class
 - GetValues method of the SqlDataReader class
 - bracket index operator [] of the SqlDataReader class
 - ExecuteScalar method of the SqlCommand class

There are some possible workarounds, including the following:

Using an assembly redirection policy. The configuration file for an application can contain an assembly redirection policy. This workaround does not require code to be changed. However, the XML configuration file needs to be deployed with the application.

Here is an example of an assembly redirection policy for Microsoft.SqlServer.Types.dll:

```
<assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
  ...
  <dependentAssembly>
    <assemblyIdentity name="Microsoft.SqlServer.Types"
publicKeyToken="89845dcd8080cc91" culture="neutral" />
    <bindingRedirect oldVersion="12.0.0.0" newVersion="10.0.0.0" />
  </dependentAssembly>
  ...
</assemblyBinding>
```


Changing the connection string. In .NET 4.5 and later, client applications can use the Type System Version property of the connection string to set up the version of types used on the client. If *Types System Version = 2014* is specified, the client will load version 12.0.0.0 of Microsoft.SqlServer.Types.dll. If not, version 10.0.0.0 will be loaded by default.

Note: .NET 4.5 will deprecate the Type System Version = Latest option. That option will stay locked into SQL Server 2008 and will always load version 10.0.0.0.

Using the GetSqlBytes method. You can work around this issue by calling the GetSqlBytes method instead of the Get methods to retrieve CLR SQL Server system types, as shown in the following example:

```
string query = "SELECT [SpatialColumn] FROM [SpatialTable]";
using (SqlConnection conn = new SqlConnection("..."))
{
    SqlCommand cmd = new SqlCommand(query, conn);

    conn.Open();
    SqlDataReader reader = cmd.ExecuteReader();

    while (reader.Read())
    {
        // In version 11.0 only
        SqlGeometry g = SqlGeometry.Deserialize(reader.GetSqlBytes(0));

        // In version 10.0 or 11.0
        SqlGeometry g2 = new SqlGeometry();
        g.Read(new BinaryReader(reader.GetSqlBytes(0).Stream));
    }
}
```

Untested/Unsupported .NET Framework Assemblies

Beginning with SQL Server 2005, SQL Server has a list of supported .NET Framework libraries, which have been tested to ensure that they meet reliability and security standards for interaction with SQL Server. Supported libraries do not need to be explicitly registered on the server before they can be used in your code. SQL Server loads them directly from the Global Assembly Cache (GAC). For list of supported assemblies, see [Supported .NET Framework Libraries](http://msdn.microsoft.com/en-us/library/ms403279(v=sql.120).aspx) (http://msdn.microsoft.com/en-us/library/ms403279(v=sql.120).aspx).

Assemblies other than those listed in [Supported .NET Framework Libraries](http://msdn.microsoft.com/en-us/library/ms403279(v=sql.120).aspx) (http://msdn.microsoft.com/en-us/library/ms403279(v=sql.120).aspx) are considered unsupported assemblies by SQL Server. This means SQL Server won't automatically

recognize them or load them even if they exist in the GAC. You need to explicitly register these assemblies in SQL Server and mark them UNSAFE. If your CLR objects are referring to unsupported assemblies, you will have to drop them and recreate them after the upgrade to SQL Server 2014. For details, see [Support policy for untested .NET Framework assemblies in the SQL Server CLR-hosted environment](http://support.microsoft.com/kb/922672) (http://support.microsoft.com/kb/922672).

Constant Folding for CLR User-Defined Functions and Methods (Only Applicable When Upgrading from SQL Server 2008 R2 or Earlier)

In SQL Server 2014, the following user-defined CLR objects are foldable:

- Deterministic scalar-valued CLR user-defined functions
- Deterministic methods of CLR user-defined types

Foldable user-defined CLR objects enhance performance when these functions or methods are called more than once with the same arguments. However, this change may cause unexpected results when non-deterministic functions or methods have been incorrectly marked as deterministic in error. The determinism of a CLR function or method is indicated by the value of the `IsDeterministic` property of `SqlFunctionAttribute` or `SqlMethodAttribute`. For more information about constant folding, see [Troubleshooting Poor Query Performance: Constant Folding and Expression Evaluation During Cardinality Estimation](http://technet.microsoft.com/en-us/library/ms175933(v=sql.120).aspx) (http://technet.microsoft.com/en-us/library/ms175933(v=sql.120).aspx). For more information about deterministic and non-deterministic functions, see [Deterministic and Nondeterministic Functions](http://technet.microsoft.com/en-us/library/ms178091(v=sql.120).aspx) (http://technet.microsoft.com/en-us/library/ms178091(v=sql.120).aspx).

Dynamic Management View (DMV) Changes

The [sys.dm_clr_appdomains view](http://msdn.microsoft.com/en-us/library/ms187720(v=sql.120).aspx) (http://msdn.microsoft.com/en-us/library/ms187720(v=sql.120).aspx) has existed since SQL Server 2005. This view returns a row for each application domain (AppDomain) in SQL Server. AppDomain is a construct in the .NET Framework CLR that is a unit of isolation for an application. This view is used to understand and troubleshoot CLR integration objects that are executing in SQL Server. SQL Server 2014 and SQL Server 2012 have three additional columns that are not found in SQL Server 2008 R2 and earlier. Those three columns are shown in Table 1.

Table 1: Additional Columns in the sys.dm_clr_appdomains View

Column Name	Data Type	Description
total_processor_time_ms	bigint	Total processor time, in milliseconds, used by all threads while executing in the current application domain since the process started. This is equivalent to System.AppDomain.MonitoringTotalProcessorTime.
total_allocated_memory_kb	bigint	Total size, in kilobytes, of all memory allocations that have been made by the application domain since it was created, without subtracting memory that has been collected. This is equivalent to System.AppDomain.MonitoringTotalAllocatedMemorySize.
survived_memory_kb	bigint	Number of kilobytes that survived the last full, blocking collection and that are known to be referenced by the current application domain. This is equivalent to System.AppDomain.MonitoringSurvivedMemorySize.

Visual Studio 2010 Compatibility

Visual Studio 2010 does not natively support SQL CLR deployment to SQL Server 2014. However, this is supported by SQL Server Data Tools (SSDT). Existing Visual Studio 2010 SQL CLR database projects can be automatically migrated to SSDT database projects. For details on SSDT, refer to Chapter 9, “SQL Server Data Tools.”

Another alternative is to deploy the SQL CLR assemblies to SQL Server 2014 manually by using Create Assembly statements.

Additional References

For an up-to-date collection of additional references for upgrading CLR objects, refer to the following links:

- [Supported .NET Framework Libraries](http://msdn.microsoft.com/en-us/library/ms403279(v=sql.120).aspx)
(http://msdn.microsoft.com/en-us/library/ms403279(v=sql.120).aspx)
- [Migration Guide to the .NET Framework 4](http://msdn.microsoft.com/en-us/library/ff657133(v=vs.100).aspx)
(http://msdn.microsoft.com/en-us/library/ff657133(v=vs.100).aspx)
- [Supported .NET Framework Libraries](http://msdn.microsoft.com/en-us/library/ms403279(v=sql.120).aspx)
(http://msdn.microsoft.com/en-us/library/ms403279(v=sql.120).aspx)
- [Support policy for untested .NET Framework assemblies in the SQL Server CLR-hosted environment](http://support.microsoft.com/kb/922672)
(http://support.microsoft.com/kb/922672)
- [Breaking Changes to Database Engine Features in SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms143179.aspx)
(http://msdn.microsoft.com/en-us/library/ms143179.aspx)

Chapter 14: SQL Server Management Objects

Introduction

SQL Server Management Objects (SMO) is a set of objects designed for programmatic management of Microsoft SQL Server. You can use SMO to build customized SQL Server management applications.

The SMO object model extends and supersedes the SQL Server Distributed Management Objects (SQL-DMO) object model. Compared to SQL-DMO, SMO increases performance, control, and ease of use. Most SQL-DMO functionality is included in SMO, and there are various new classes that support new features in SQL Server.

Because SMO is fully compatible with SQL Server 2005 and later, you can easily manage a multi-version environment.

If you want to develop an application that uses SMO, you should select the Client Tools SDK when you install SQL Server. To install the Client Tools SDK without installing SQL Server, install Shared Management Objects from the [Microsoft SQL Server 2014 Feature Pack](http://www.microsoft.com/en-US/download/details.aspx?id=42295) (<http://www.microsoft.com/en-US/download/details.aspx?id=42295>). If you want to ensure that SMO is installed on a computer that will run your application, you can use the Shared Management Objects.msi in the SQL Server 2014 Feature Pack.

SMO uses the Microsoft System.Data.SqlClient object driver to connect to and communicate with instances of SQL Server. SMO clients require SQL Server Native Client (which is included with SQL Server) and the .NET Framework 2.0. To develop applications by using SMO, you must have a recent version of Microsoft Visual Studio installed.

Preparing to Upgrade

Before upgrading to SQL Server 2014, you should investigate which features are deprecated or discontinued.

Deprecated Features

Several features have been deprecated in previous version of SQL Server. When you upgrade to SQL Server 2014 SMO, you might have to modify the code by changing or removing deprecated functionalities. For more information about deprecated features, see [Deprecated Database Engine Features in SQL Server 2014](http://msdn.microsoft.com/en-us/library/cc707789(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/cc707789\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/cc707789(v=sql.120).aspx)) in SQL Server 2014

Books Online. In addition, by changing the target version of SQL Server using the *Other versions* hyperlink in this page, you can check if previous deprecated features are affecting your code.

There are also several deprecated objects in SMO libraries that will be removed in future SQL Server versions. Table 1 shows the deprecated SMO objects in SQL Server 2014. For more information, see [Deprecated Management Tools Features in SQL Server 2014](http://msdn.microsoft.com/en-us/library/cc879341(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/cc879341\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/cc879341(v=sql.120).aspx)) in SQL Server 2014 Books Online.

Table 1: Deprecated Objects in SMO

Feature	Deprecation Stage
Microsoft.SqlServer.Management.Smo.Information class	Announcement
Microsoft.SqlServer.Management.Smo.Settings class	Announcement
Microsoft.SqlServer.Management.Smo.DatabaseOptions class	Announcement
Microsoft.SqlServer.Management.Smo.DatabaseDdlTrigger.NotForReplication property	Announcement

Discontinued Functionality

When you upgrade to SQL Server 2014 SMO, you might have to modify the code by changing or removing discontinued functionalities. For more information about general database engine discontinued features, see [Discontinued Database Engine Functionality in SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms144262(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms144262\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms144262(v=sql.120).aspx)) in SQL Server 2014 Books Online.

Keep in mind that if you are still using SQL-DMO, you should be aware that this functionality is no longer included in SQL Server 2014 and must be converted to use SMO. For more information about SQL-DMO mapping to SMO, see [SQL-DMO Mapping to SMO](http://msdn.microsoft.com/en-us/library/ms162159(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms162159\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms162159(v=sql.120).aspx)) in SQL Server 2014 Books Online.

Upgrading from previous versions

SMO applications and tools are external to the SQL Server engine, so they can be upgraded at any time, without affecting database engine upgrade process. The Microsoft development team tried to maintain compatibility with earlier versions as much as possible. SMO applications that were written using previous versions of SQL Server can run “as is” using legacy features or can be recompiled by using SMO in SQL Server 2014.

SMO is implemented as a set of Microsoft .NET Framework assemblies. This means that the Common Language Runtime (CLR) from the .NET Framework 3.5 must be installed before using SMO. The SMO assemblies are installed by default into the Global Assembly Cache (GAC) with the SQL Server SDK installation option. The assemblies are located in C:\Program Files (x86)\Microsoft SQL Server\120\SDK\Assemblies\ with release number 12.0.2000.8, as Figure 1 shows.

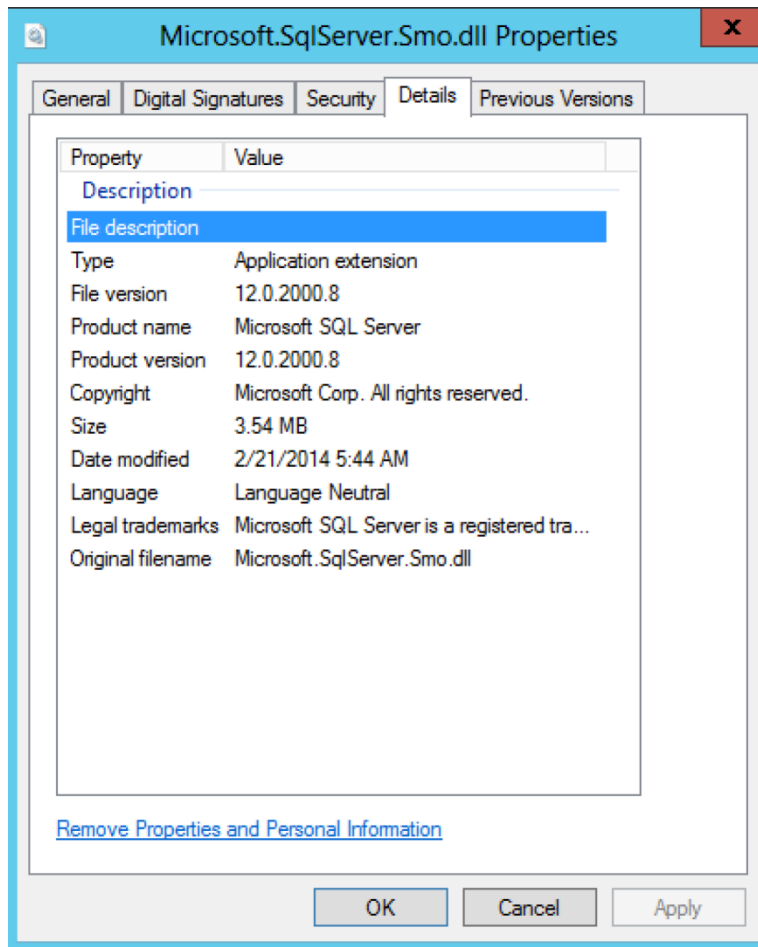


Figure 1: SMO assembly properties

SMO libraries included in SQL Server 2014 are built against the .NET Framework 3.5 but can be consumed by managed applications that target either the .NET 3.5 or .NET 4.x runtimes. It is the application developer's choice of which .NET runtime to target. This can be useful in scenarios where you have a managed application that currently targets .NET 3.5 and wants to use SQL Server 2014 SMO, without being forced to target the .NET 4.x runtime.

Before recompiling SMO projects, you must remove references to old SMO DLLs and substitute them with new SMO DLLs, provided with SQL Server 2014.

At the minimum, your SMO projects should reference the following:

- Microsoft.SqlServer.ConnectionInfo
- Microsoft.SqlServer.Smo
- Microsoft.SqlServer.Management.Sdk.Sfc

SmoEnum.dll has been removed in SQL Server 2012, so references to this DLL must be removed from the SMO project.

If your code uses Urn functionality, such as `Server.GetSqlSmoObject(Urn)`, you must also reference and link to the `Microsoft.SqlServer.Management.Sdk.Sfc` namespace. If your code uses the `Transfer` object directly, you will have to link to the `Microsoft.SqlServer.Management.SmoExtended` namespace. For more information about SMO assemblies, see [Files and Version Numbers](http://msdn.microsoft.com/en-us/library/ms162161(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms162161\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms162161(v=sql.120).aspx)).

SMO and PowerShell

SQL Server 2014 supports Windows PowerShell, which is a powerful scripting shell that lets administrators and developers automate server administration and application deployment. The PowerShell language supports more-complex logic than the Transact-SQL (T-SQL) language, giving SQL Server administrators the ability to build robust administration scripts. Windows PowerShell scripts can also be used to administer other Microsoft server products. This gives administrators a common scripting language across servers.

The SQL Server 2014 PowerShell components can be used to manage instances of SQL Server 2000 or later. Instances of SQL Server 2005 must be running SP2 or later. Instances of SQL Server 2000 must be running SP4 or later. When the SQL Server 2014 PowerShell components are used with earlier versions of SQL Server, they are limited to the functionality available in those versions.

To load SMO into PowerShell and execute the scripts you can:

- Use a module named `sqlps`
`Import-Module SQLPS -DisableNameChecking`
- Load the SMO assemblies using the PowerShell 2.0's `AddType` cmdlet
`Add-Type -AssemblyName "Microsoft.SqlServer.Smo"`

- Use the still reliable load assembly mode

```
[reflection.assembly]::LoadWithPartialName("Microsoft.SqlServer.Smo")
```

Existing PowerShell scripts should work without modifications because the PowerShell runtime loads the latest version of the SMO assemblies, which are fully compatible with older SQL Server versions.

Conclusion

Upgrading SMO projects to SQL Server 2014 can be a straightforward process. By referencing new DLLs and recompiling existing applications, it is possible to access new features exposed by new SMO libraries (to manage AlwaysOn Availability Groups, for example).

Additional References

For an up-to-date collection of additional references for upgrading SMO projects to SQL Server 2014, see the following links:

- [Overview \(SMO\)](http://msdn.microsoft.com/en-us/library/ms162557(v=sql.120).aspx)
(http://msdn.microsoft.com/en-us/library/ms162557(v=sql.120).aspx)
- [SQL Server 2014 Web Site](http://www.microsoft.com/sqlserver/en/us/default.aspx)
(http://www.microsoft.com/sqlserver/en/us/default.aspx)
- [Books Online for SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms130214(v=sql.120).aspx)
(http://msdn.microsoft.com/en-us/library/ms130214(v=sql.120).aspx)
- [SQL Server Developer Center](http://msdn.microsoft.com/en-us/sqlserver/)
(http://msdn.microsoft.com/en-us/sqlserver/)
- [SQL Server TechNet Resources](http://technet.microsoft.com/en-us/sqlserver/)
(http://technet.microsoft.com/en-us/sqlserver/)

Chapter 15: Business Intelligence Tools

Introduction

The business intelligence (BI) and data visualization tools in SQL Server 2014 provide an array of choices to meet different business and user reporting requirements. These tools are made available for users and IT professionals in various roles to plan, design, and manage data and report content. This chapter introduces all of the BI tools, including applications that are covered in greater detail in other chapters. In most cases, it will give you a high-level view of the tools and their features. As noted, please refer to the appropriate chapters for the background and details you will need to plan for and implement a comprehensive upgrade.

BI Tools Users

BI solutions require the participation of professionals in different business roles, with users and technical professionals having different perspectives and different reasons for using various tools. User roles for the BI tools fall into three categories, which generally include business information workers, software developers, and system administrators.

Business Information Workers

Business information workers typically want to explore information and find answers, rather than design complex reports. They need easy-to-use tools to browse data and create simple reports, quickly and with less technical expertise. They typically create a report to answer a specific question or address a particular need, and then they may discard the report or save it to a personal area for reuse. They tend to create a separate report for each task and may or may not share these reports with others who have similar needs.

The role of the business information worker is changing rapidly in many organizations that need to analyze information from diverse sources. Although it may be true that business information workers need to create reports and data visualizations without complex programming tools, many now take on the role of data analyst, BI specialist, or data scientist and have the expertise and desire to use sophisticated data modeling, visualization, and analysis tools found in modern self-service BI tools. Information workers will primary use tools like:

- Excel 2013 Professional Plus or Office 365 Professional Plus
- The Power Pivot for Excel 2013 add-in

- Report Builder 3.0
- Power View in SharePoint 2013 Enterprise Edition

Software Developers

Members of this role write complex queries and custom programming code to process business rules and give reports conditional formatting and behavior. Developers may prefer to work within the SQL Server Data Tools (SSDT) report designer environment because it is similar to familiar programming tools. SSDT used with different versions and editions of Visual Studio support multi-project solution management, version control, and team collaboration. Report design is not the same as application development, but reports can be integrated into custom dashboards and applications. Designing a report is faster and easier in some ways than developing software. Advanced report design can involve writing code and even developing custom components.

System Administrators

System administrators are typically concerned with the setup and ongoing maintenance of servers and the infrastructure to keep reporting solutions available and working. Administrators typically spend their time and energy managing security and optimizing the system for efficiency. Reporting Services has an administrative component that is especially important in large-scale implementations.

In smaller organizations, the same person may play the role of system administrator, developer, and report designer. As an administrator, your primary management tools for BI projects and components will be SQL Server Management Studio (SSMS), configuration tools, and SQL Server Profiler. For Reporting Services, you will use Report Manager in native mode and the SharePoint context menus for reports in SharePoint integrated mode. For other BI content in SharePoint, you will use the context menus and SharePoint configuration pages.

BI Tools Overview

The term “BI tools” refers to a collection of configuration and design applications that can be found in a few different places, including the Windows App menu, SharePoint, and Office application add-ins. Some of these tools are installed with different products (in addition to SQL Server 2014) such as Visual Studio and Office 2013 Professional Plus. These tools are all part of the SQL Server 2014 and Microsoft Business Intelligence ecosystem.

To get started, we will discuss these tools and show you where to find them. To locate many of the installed tools, a good place to start is the Windows Start screen (in Windows 8 or later and in Windows Server 2012 or later) or the Windows Start menu (in Windows 7 or earlier and in Windows Server 2008 R2 or later). Figure 1 shows a selection of tiles pinned to the Windows Server 2012 Start screen. These are tools used to design, develop, and test BI solutions.



Figure 1: Windows Server 2012 Start screen with SQL Server 2014 tools, Visual Studio 2013, and Excel 2013.

Referring to these menu items, the BI tools covered in this chapter include:

- Import and Export Wizard
- SQL Server Data Tools
- SQL Server Management Studio
- Analysis Services Deployment Wizard
- Reporting Services Configuration Manager
- Integration Services Deployment Wizard
- Integration Services Project Conversion Wizard
- SQL Server Profiler

Additional tools that do not appear on the Start menu in this configuration include:

- Report Builder
- Power Pivot for Excel 2013 add-in
- Power View add-in for Excel 2013
- Data Mining Add-ins for Office

Import and Export Wizard

The Import and Export Wizard is a simple interface that will create and utilize a SQL Server Integration Services (SSIS) package to read data from a data source and transform data into a destination. By utilizing the capabilities of SSIS, data can be imported from and exported to many file formats and database products supported by SSIS. Two versions of the wizard are optimized for 32-bit or 64-bit runtime environments and use corresponding data access components and data providers for some data sources that require platform-specific support.

The Import and Export Wizard can also be launched from SSMS when connected to a SQL Server relational instance. Separate Import and Export menu items in SSMS actually launch the same tool and set default values for either the source or destination adaptors.

The Import and Export Wizard user interface has not changed significantly from the previous version, but it does support updated data providers and utilizes the updated SSIS 2014 architecture, which was introduced in SQL Server 2012. After working through the wizard, the resulting SSIS package can be executed immediately and then

optionally saved to the file system or to a SQL Server instance. You can use the Import and Export Data wizard when you just need to import or export data. You can use the package as a starting point in an SSIS project. Details are available in [SQL Server Import and Export Wizard](http://msdn.microsoft.com/en-us/library/ms141209.aspx) (<http://msdn.microsoft.com/en-us/library/ms141209.aspx>).

Chapter 17, "Integration Services," covers the upgrade path for SSIS. Please refer to this chapter for specific steps and considerations for upgrading SSIS packages and projects.

SQL Server Data Tools for Business Intelligence

SQL Server Data Tools (SSDT) and SQL Server Data Tools for Business Intelligence (SSDT-BI) are separate add-ins for Visual Studio 2012 and Visual Studio 2013. Because SQL Server and Visual Studio are on separate release schedules, different versions of the add-ins are available and updates are provided as needed. Visual Studio must be obtained separately and no longer installs with the SQL Server Client Tools. After Visual Studio is installed, you should download and install the appropriate versions of the SSDT and SSDT-BI add-ins from the [Microsoft SQL Server Data Tools](http://msdn.microsoft.com/en-us/hh297027.aspx) web page (<http://msdn.microsoft.com/en-us/hh297027.aspx>). After initially installing SQL Server, you will need to restart your computer before installing the SSDT and SSDT-BI add-ins. The specific compatibility requirements for the current and future versions of Visual Studio have changed and are likely to change in the future so read the Microsoft SQL Server Data Tools web page thoroughly before installing the add-ins.

Running the setup for SSDT-BI launches the SQL Server 2014 Setup Wizard, which opens the SQL Server Installation Center. On the Installation page, choose the first option *New SQL Server stand-alone installation or add features to an existing installation*. On the Installation Type page, shown in Figure 2, choose *Perform a new installation of SQL Server 2014*. The wizard will check for available SQL Server updates. On the Feature Selection page, under Shared features, select the *SQL Server Data Tools – Business Intelligence for Visual Studio 2013* feature or the feature you want to install. Follow the prompts to complete the installation.

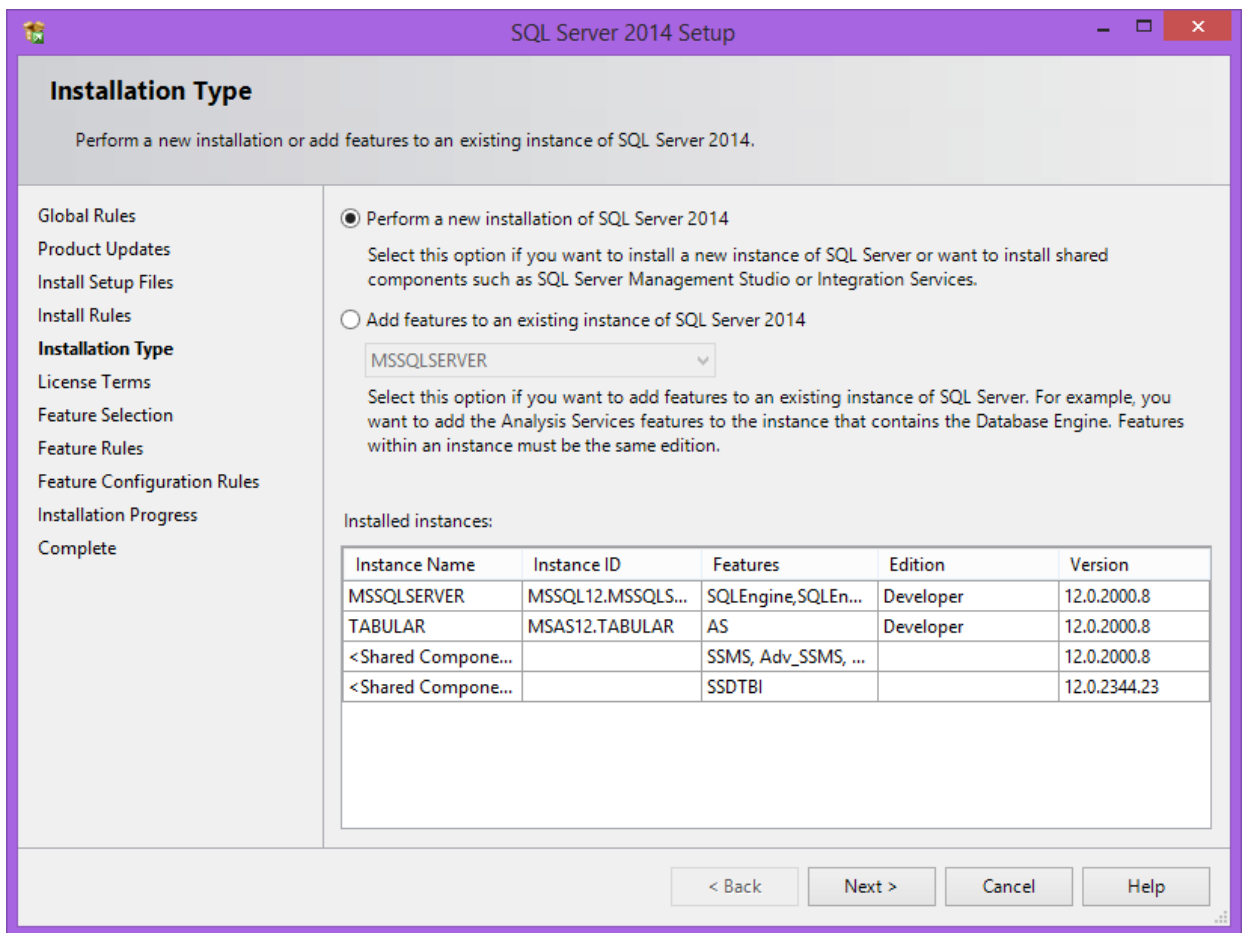


Figure 2: Installation Type page in the SQL Server 2014 Setup Wizard

This installation process can be time-consuming. You may be prompted to restart your computer when the Setup Wizard has completed.

The SSDT-BI add-in adds the following project templates to the Visual Studio project types for creating a new project, as shown in Figure 3:

- Analysis Services Multidimensional and Data Mining
- Integration Services
- Reporting Services
- Analysis Services Tabular

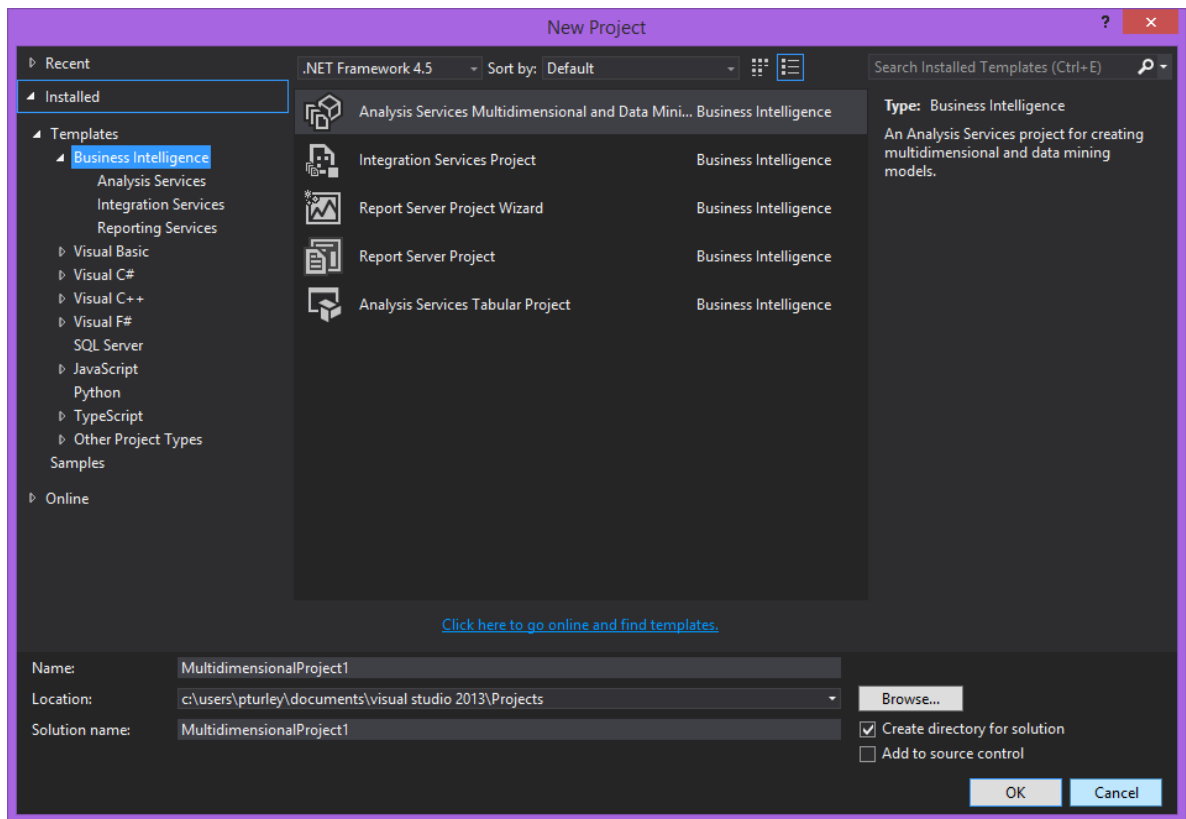


Figure 3: Visual Studio 2013 with SSDT-BI project templates

In addition, the New Project dialog box includes templates that will launch wizards, enabling you to create projects from imported objects. These include an Analysis Services Multidimensional project imported from a server database, an Analysis Services Tabular project imported from a server database, and an Analysis Services Tabular project imported from a Power Pivot Excel workbook file. A Reporting Services project can also be created using the Report Server Project Wizard.

SSDT Database Projects

SSDT Database Project allows you to develop SQL Server database Transact-SQL (T-SQL) scripts to create, synchronize, and manage database objects. SSDT now aligns the database development experience with familiar Visual Studio development and debugging tools, such as code-completion, IntelliSense, compare capabilities, and rich version control capabilities. The SSDT Database Project tools are extensive, allowing developers to generate database schema deployment scripts to create and update all database objects. SSDT Database Project can also create portable database backup packages, incorporating data with schema change management. This tool is covered in Chapter 9, “SQL Server Data Tools.” For developers and other database professionals, SSDT Database Project uses declarative, model-based tools to develop a database and

database objects online or offline for on-premises and SQL Azure databases. Visual Studio database projects can be converted to the newer SSDT database projects. For conversion steps and considerations, refer to [How to: Convert VS 2010 Database Projects to SSDT Database Projects and Re-target to a Different Platform](http://msdn.microsoft.com/en-us/library/hh272689(v=VS.103).aspx) ([http://msdn.microsoft.com/en-us/library/hh272689\(v=VS.103\).aspx](http://msdn.microsoft.com/en-us/library/hh272689(v=VS.103).aspx)).

Reporting Services Projects

Report definition files created for SQL Server 2008, 2008 R2, and SQL Server 2012 need not be upgraded to SQL Server 2014. Opening an earlier version project in SSDT 2014 will upgrade the project. Reports created for SQL Server 2008 can optionally be upgraded to the SQL Server 2008 R2 standard; these reports are supported in SQL Server Reporting Services (SSRS) 2014. SQL Server 2014 does not introduce a new report definition language (RDL) version and allows you to deploy reports in one of two compatibility modes: SSRS 2008 and SSRS 2008 R2. Refer to Chapter 18, "Reporting Services," for guidance on planning a Reporting Services project upgrade and for upgrading reports deployed to a report server.

Analysis Services Tabular Model Projects

The Analysis Services Tabular Model project type was introduced in SQL Server 2012. The designer experience in SSDT has many similarities to the designer experience in the Power Pivot for Excel 2013 add-in. A significant difference is that, even during the design stage, tabular model data and metadata is stored in a live SQL Server 2014 Analysis Services (SSAS) database instance configured for tabular storage mode. This means that a developer must have connectivity to the development SSAS database server or must have a local instance installed. Figure 4 shows the tabular model project designer in SSDT for Visual Studio 2013.

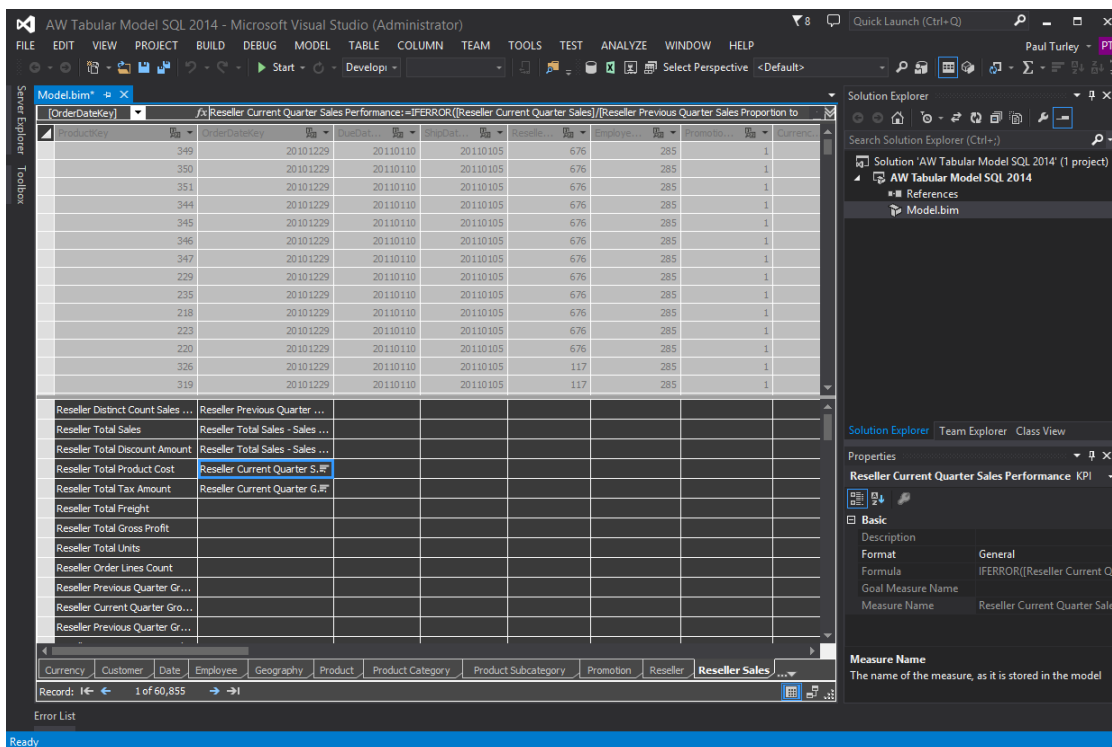


Figure 4: Tabular model project designer

SQL Server Management Studio

SSMS is not exclusively a BI tool but an essential utility for IT-developed BI solutions. It is the central management tool for all SQL Server services and objects. Upgrade considerations are minimal for SSMS with respect to SQL Server 2008, 2008 R2, and 2012 databases and services. Refer to Chapter 2, “Management Tools,” for details about using SSMS to upgrade earlier SQL Server versions and to manage new features in SQL Server 2014.

SQL Server 2014 includes many enhancements and user interface improvements for solution development and administration of the SQL Server Database Engine, Analysis Services, and Reporting Services. Like SSDT, SSMS utilizes the Visual Studio shell, providing more extensible support and compatibility. SSMS is a vital tool for upgrading to SQL Server 2014. Database objects from earlier SQL Server versions can be scripted and converted. Backward compatibility and the ability to upgrade databases are provided for versions back to SQL Server 2005 SP4. Be mindful that databases can only be upgraded and not converted, meaning that the upgraded database remains in the earlier version compatibility mode. For specific considerations and details regarding compatible editions and product versions for upgrade, see [SQL Server Management Studio](http://msdn.microsoft.com/en-us/library/hh213248.aspx) (<http://msdn.microsoft.com/en-us/library/hh213248.aspx>) in SQL Server 2014 Books Online.

Analysis Services Deployment Wizard

The Analysis Services Deployment Wizard is used to deploy XML database definition files and optional configuration files to an SSAS server. These files are created from an SSDT Analysis Services multidimensional project using the Build project option. The Deploy and Process actions also build deployment scripts and store these files in the bin folder under the project file folders.

The Analysis Services Deployment Wizard has not changed significantly from SQL Server 2008 R2 and SQL Server 2012. For more details and options, see [Deploy Model Solutions Using the Deployment Wizard](http://msdn.microsoft.com/en-us/library/ms176121.aspx) (<http://msdn.microsoft.com/en-us/library/ms176121.aspx>). Refer to Chapter 16, "Analysis Services," for specific guidance on how to upgrade Analysis Services projects.

Reporting Services Configuration Manager

Reporting Services Configuration Manager is used to configure and manage an SSRS instance in native mode. You do not use this tool for managing any settings when SSRS is integrated with SharePoint 2013. Additional information about the use of this tool may be found in [Reporting Services Configuration Manager \(Native Mode\)](http://msdn.microsoft.com/en-us/library/ms156305(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms156305\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms156305(v=sql.120).aspx)).

In SharePoint integrated mode, all configuration settings are managed using the SharePoint Central Administration site. To change a SharePoint-integrated SSRS instance to native mode, use the Configuration Manager to create a new report server content database in native mode. To migrate the instance from native to SharePoint integrated mode, use the Central Administrator. Note that existing report content will not be moved in either case. The recommended approach to migrate existing report server content is to save reports, shared data sources, and data sets to files in the file system and then upload them using SharePoint or an SSDT report project. More information about migrating report server instances and switching integration modes can be found in [Configure and Administer a Report Server \(SSRS Native Mode\)](http://msdn.microsoft.com/en-us/library/ms159624(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms159624\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms159624(v=sql.120).aspx)).

Integration Services Deployment Wizard

A significant enhancement to SQL Server Integration Services introduced in SQL Server 2012 is the ability to store packages in the SSISDB catalog. Previous versions of SSIS offered two deployment and storage options. You could store packages in the file system on the server or use SQL Server storage mode to deploy packages to the MSDB database. Although packages could be designed and managed using a Business

Intelligence Development Studio (BIDS) project, packages were stored and managed as separate objects. This paradigm changed in SQL Server 2012 and later by introducing the project deployment model. When the project is deployed from SSDT, the Convert to Package Deployment Model dialog box will help you determine whether the project and constituent packages are compatible with project or package deployment modes, and offers methods to ensure compatibility.

Integration Services Project Conversion Wizard

This section provides an overview of the Integration Services Project Conversion Wizard. Chapter 17, "Integration Services," provides comprehensive guidance about planning and implementing an SSIS project upgrade.

You can upgrade SSIS packages created with earlier product versions in three different ways: from SSMS, from SSDT, or from the file system or command prompt. To upgrade packages in an earlier-version SSIS project, open the existing project in SSDT. You can upgrade each package individually using the right-click menu. To upgrade all packages, right-click the SSIS Packages node in Solution Explorer and choose Upgrade All Packages from the menu.

To upgrade packages stored using the former SQL Server storage mode or file system storage mode, use SSMS. Connect to an SSIS instance and then expand the Stored Packages node. Right-click the File System or MSDB node, and then click Upgrade Packages.

You can also run the conversion wizard as a stand-alone application. To do this, use a command prompt, create a shortcut, or locate the SSISUpgrade.exe file located in the C:\Program Files (x86)\Microsoft SQL Server\120\DTS\Binn folder.

The Integration Services Project Conversion Wizard will allow you to save a backup copy of any converted package or SSIS project prior to converting the files to work with SQL Server 2014 Integration Services. The location of the converted files and backup folders created by the wizard may be different depending on the method used to launch the wizard. For details, see [Upgrade Integration Services Packages Using the SSIS Package Upgrade Wizard](http://msdn.microsoft.com/en-us/library/cc280547(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/cc280547\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/cc280547(v=sql.120).aspx)).

SQL Server Profiler

SQL Server Profiler is an essential tool for optimizing BI solutions and troubleshooting issues. By tracing a query execution, you can see how the SQL Server relational engine

or the multidimensional engine breaks the query into steps and how it manages the query internally. This can also be useful to debugging connection and security configuration issues.

The SQL Server Profiler tool itself is largely unchanged in SQL Server 2014 but several new events and counters have been added in several services that can be used to monitor performance and audit events.

Report Builder

The Report Builder design tool in SQL Server 2014 is nearly identical to Report Builder in SQL Server 2012 and includes the same feature set. The original Report Builder tool, available in SQL Server 2005 (later called Report Builder 1.0) is deprecated in SQL Server 2014, along with the first-generation semantic report models and report model projects. These will run on a SQL Server 2014 SSRS instance but are not supported.

Report Builder is available by default when Reporting Services is installed in native or SharePoint integrated mode. Report Builder can be launched directly from the web browser using Report Manager in native mode or from a document library in SharePoint integrated mode. The first time a user opens Report Builder from a report server or SharePoint site collection, it will be installed to their desktop as a .NET ClickOnce application. Note that if a user were to navigate to a different site collection or report server, they will be prompted to install Report Builder multiple times. This is expected behavior.

To open Report Builder in native mode and create a new report, navigate to Report Manager (typically *http://<server name>/Reports* by default), and in any report folder, click the Report Builder button on the toolbar.

To edit an existing deployed report with Report Builder, hover the mouse pointer over the report name in Report Manager, click the down arrow, and select Edit in Report Builder. In SharePoint, the context menus work exactly the same way.

To create a new report with Report Builder in SharePoint integrated mode, a document library must be configured to include the Report Builder Report content type. For information on how to add a content type to a library, see [Add Report Server Content Types to a Library \(Reporting Services in SharePoint Integrated Mode\)](http://msdn.microsoft.com/en-us/library/bb326289(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/bb326289\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/bb326289(v=sql.120).aspx)).

To open Report Builder, navigate to the document library and choose the Documents tab under the Library Tools group. In the left-side ribbon, select the Report Builder

Report option on the New Document menu.

Report Builder can also be installed to run as a standalone application. It can then be launched from the Windows Start menu.

The highest compatibility level for reports created with either Report Builder or SSDT in SQL Server 2014 is SQL Server 2008 R2. Both Report Builder and SSDT can be used to deploy reports with SQL Server 2008 compatibility. Note that certain features may be lost in 2008 compatibility mode that cannot be recovered if they are saved again to the newer version.

Power Pivot for Excel 2013 Add-In

The Power Pivot for Excel 2013 add-in is used to author Power Pivot data models that are stored within an Excel 2013 workbook document. Note that there was a subtle name change from "PowerPivot" to "Power Pivot." The space was added to align the branding with all of the other "Power..." named BI products and tools. Both names are commonly used. Power Pivot is a data storage and aggregation technology that allows you to manage structured data in multiple, related tables, and to perform sophisticated calculations and analysis using in-memory aggregation. Using highly efficient data compression, Power Pivot can store and analyze tens or hundreds of millions of rows very quickly and with relatively few computer resources. The Power Pivot for Excel 2013 add-in is a client-side technology that stores data locally, compressed in a data model stored within the Excel document. When a workbook is published to a SharePoint 2013 Enterprise Edition site with the Power Pivot add-in for SharePoint configured, Power Pivot becomes a server-side technology. Using Excel Services in SharePoint 2013, many users can browse and interact with the data in a web browser concurrently without using Excel 2013 on their workstations.

Power Pivot is a central component in Power BI for Office 365. A Power Pivot data model authored in Excel 2013 for Office 365 can be searched, published, analyzed, and visualized with Power BI Q&A and Power View in a Power BI site. Power BI includes other integrated tools such as Power Query and Power Map. For more information about Power BI for Office 365, visit the [Power BI for Office 365 site](http://www.microsoft.com/en-us/powerbi/default.aspx) (<http://www.microsoft.com/en-us/powerbi/default.aspx>).

The Power Pivot add-in is installed with Office 2013 Professional Plus and is no longer a separate download as it was for Excel 2010. The Power Pivot and Power View add-ins must be enabled before they can be used. To do this, open the Excel Options window from the File tab. On the Add-Ins page, use the Manage drop-down list at the bottom

of the page to select COM Add-ins and click Go. On the COM Add-Ins window, enable Power Pivot and any other add-ins you would like to use, and then click OK, as Figure 5 shows. This will add a tab for each add-in to the Excel ribbon tabs.

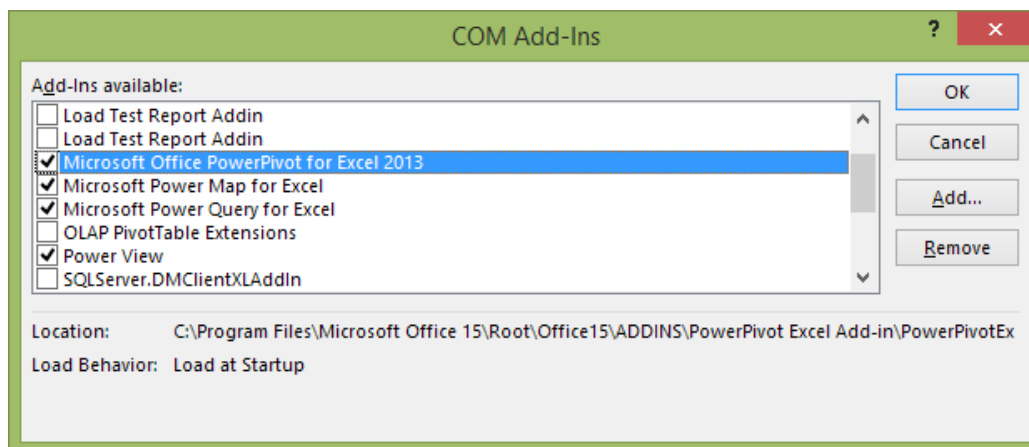


Figure 5: Enabling Power Pivot

Power Pivot data models can be visualized using an Excel PivotTable or PivotChart, Power View in Excel 2013 Professional Plus Edition, or Power View in SharePoint Server 2013. The volume of data in a Power Pivot data model is limited by the available memory on the computer and the hardware architecture when running the 32-bit build of Microsoft Office. Excel 2013 is no longer limited to 2 GB per file as it was in previous versions. Some data providers that are used to import and load data into a Power Pivot model use temporary disk space and have specific size limitations. The document size restriction in SharePoint is 2 GB due to the data type used to store list and library data in the SharePoint SQL Server content database. Note that there are restrictive default file size and upload size settings that may need to be modified by a SharePoint administrator.

You should store only text, date, and numeric values in a Power Pivot model and not binary, image, XML, geospatial, and other structured data types. Following these guidelines, a Power Pivot workbook can be used to store a large volume compressed data. To learn more about Power Pivot, visit the [Power Pivot: Powerful data analysis and data modeling in Excel page](http://office.microsoft.com/en-us/excel-help/powerpivot-powerful-data-analysis-and-data-modeling-in-excel-HA102837110.aspx) (<http://office.microsoft.com/en-us/excel-help/powerpivot-powerful-data-analysis-and-data-modeling-in-excel-HA102837110.aspx>).

The Power Pivot for Excel 2013 add-in was updated with the same storage and query engine used in SQL Server 2012 tabular models, and the design features are very similar to those in SSDT-BI. Figure 6 shows Power Pivot's graphical relationship designer. Another improvement is that there is no dependence on having any installed SQL Server instance for desktop users.

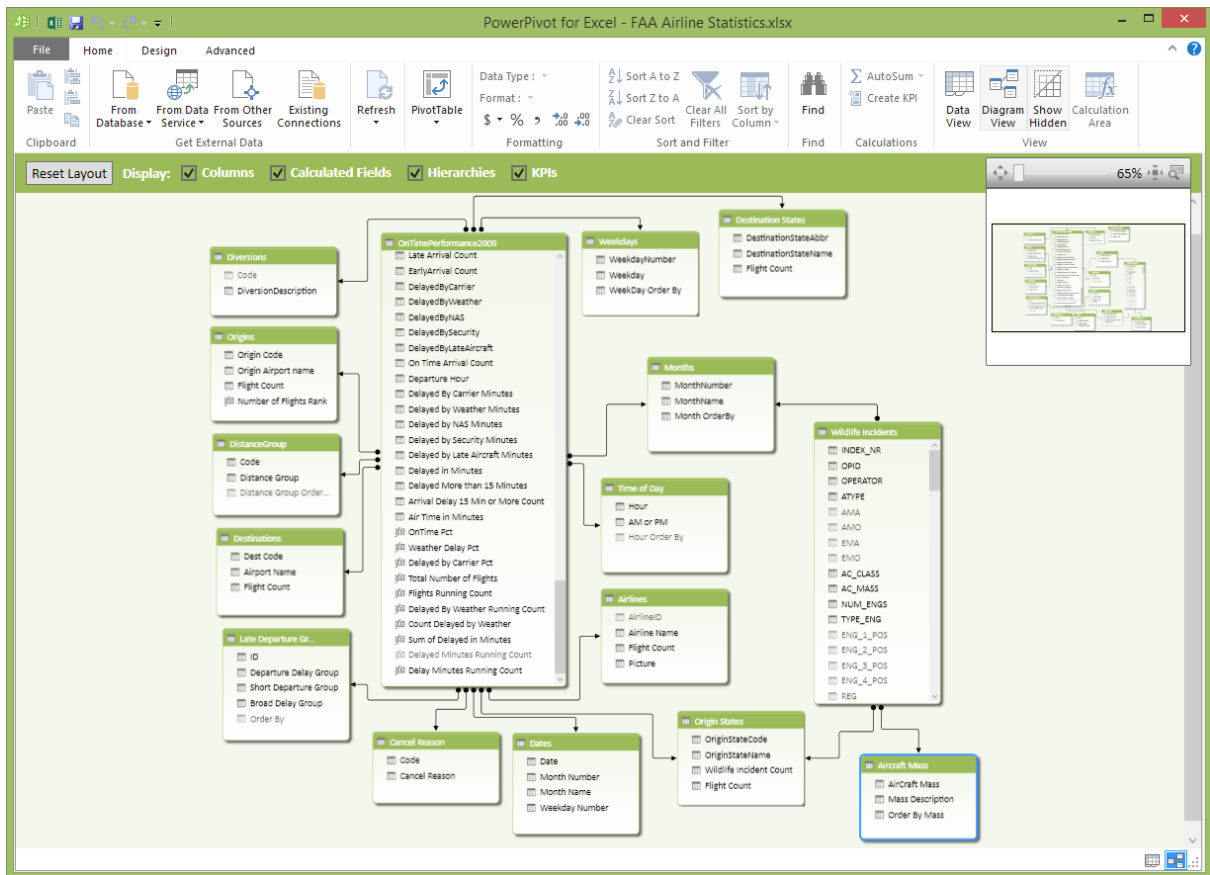


Figure 6: The Power Pivot graphical model designer in Diagram view

For upgrade purposes, a Power Pivot workbook authored using the Power Pivot for Excel 2010 add-in will be upgraded when opened using the 2013 Excel add-in. A Power Pivot model cannot be downgraded to work with the earlier version of the Excel add-in.

A Power Pivot workbook model can be upgraded to an Analysis Services Tabular Model by importing the workbook file into a tabular project in SSDT but an SSAS tabular model cannot be converted back to a Power Pivot model in Excel.

Power View

Power View was one of the most notable additions to the SQL Server 2012 BI and SharePoint arsenal. This tool enables a powerful data visualization experience that was designed with business users in mind. Power View may be easy to use, but it is not light on capabilities and useful features for serious business data analysis. Don't think of Power View as a report design tool like SSDT report designer or Report Builder. Power View is a data analysis and presentation tool that you can use to browse and visualize information in order to make it more graphical and meaningful. With it, business users can gain insights and make impactful presentations with data in prepared semantic models.

Data sources are either a Power Pivot workbook published to SharePoint Server 2013 or an Analysis Services Tabular Model, stored in an Analysis Services Tabular instance. In SharePoint 2013, you can launch Power View directly from a published Power Pivot workbook or from a BISM or RSDS connection file defined in a SharePoint connection library. A connection can provide connectivity to a Power Pivot workbook or tabular model.

The tool is highly interactive. Figure 7 shows a Power View report with multiple related visuals. When the name of an airline is clicked in the bar chart on the right side, this acts a slicer, filtering the highlighted data displayed in the stacked column series chart on the left.

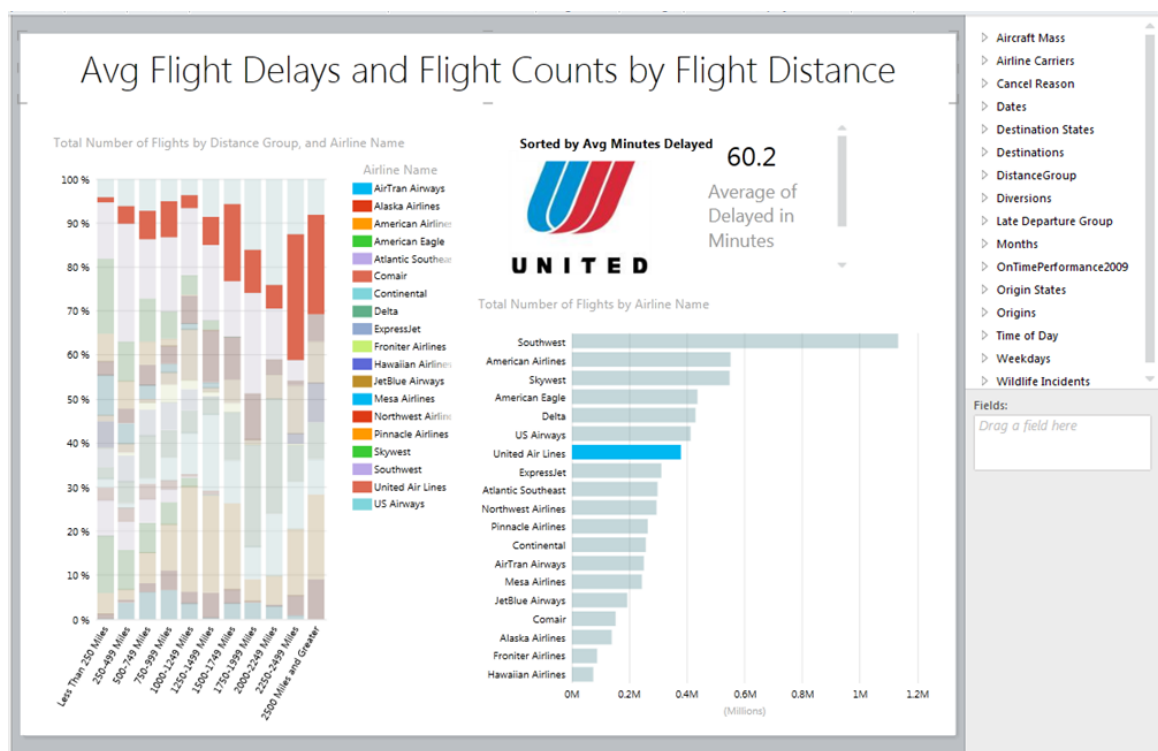


Figure 7: Multiple related visuals and slicers

Power View is installed and enabled in SharePoint Server 2013 Enterprise Edition with the Reporting Services add-in for SharePoint 2013. To use Power View with published Power Pivot workbooks, the Power Pivot add-in for SharePoint 2013 must also be installed and configured.

Data Mining Add-ins for Office

The SQL Server 2012 SP1 Data Mining Add-ins for Office will work with SQL Server 2014, Office 2010, and Office 2013. The download package includes installers for both

32-bit and 64-bit Office. You will need to install the package that matches the architecture of your Office installation. The previous version of the Data Mining Add-ins for Office 2007 works only with the 32-bit version of Excel. An SSAS 2014 or SSAS 2012 instance in Multidimensional storage mode is required for the Excel add-in to function. An Excel data mining user will need to be granted rights to create mining models on the SSAS instance. This is considered to be an elevated privilege, so it may be easier and more secure to install a separate SSAS Multidimensional instance with administrative rights granted for data mining users.

If you have existing data mining content created in previous versions of Office, no steps are necessary to use those models with the latest version of the Data Mining Add-ins for Office. The Data Mining Add-ins for Office can be downloaded and installed from the [Microsoft Download Center](http://www.microsoft.com/en-us/download/details.aspx?id=35578) (<http://www.microsoft.com/en-us/download/details.aspx?id=35578>), where you will find additional information about the add-ins' system requirements and installation. For supporting information, updates, tutorials, and samples, see Microsoft's [SQLServerDataMining.com](http://www.microsoft.com/sqlserver/datamining) web site.

Preparing to Upgrade BIDS to SSDT Projects

The SSDT project types haven't changed between SQL Server 2012 and SQL Server 2014, and don't require upgrades between these two product versions. The steps provided to upgrade to SQL Server 2012 SSDT projects also apply to SQL Server 2014. The steps and process for upgrading SQL Server 2008 and 2008 R2 BIDS projects to SQL Server 2012 SSDT projects are summarized as follows:

- Identify projects to be upgraded.
- Create backup copies of project folders and files.
- Backup existing databases and deployed objects as needed.
- Use the Visual Studio Conversion Wizard to convert project files.
- Review the conversion report and validate successful conversion.
- Convert individual object files as needed.
- Deploy new objects from converted SSDT projects.

As with any production solution, an upgrade should be tested and validated on a non-production server using a backup copy of the databases and deployed objects. The Visual Studio Conversion Wizard and each of the project type-specific conversion utilities perform backups, but it is always a good idea to create a master backup of

these project files on separate media. The most reliable way to back up a project is to copy the entire directory structure, starting with the solution folder, including all projects and subfolders, to a network share or external storage device.

Any configuration change comes with an inherent risk so it's advisable to proceed with caution and have a comprehensive recovery plan. Some product features in SQL Server 2014 have changed more than others and if you plan to take advantage of these improvements, you should consider the potential impact of making a change to an existing solution. Table 1 provides some considerations for upgrading each project type.

Table 1: Upgrade Considerations for the Types of Projects

Project Type	Changes from SQL Server 2008 R2 to 2012 (no changes in 2014)	Considerations
Integration Services	Many new features and architectural changes.	Convert the BIDS project to SSDT. Convert and test each package in the SSDT designer. Redeploy and thoroughly test all packages on a test server and then in production. Use the Convert to Package Deployment Model to check compatibility.
Analysis Services Multidimensional	Service architecture and design environment are similar.	Perform an in-place upgrade on the existing database or re-deploy from SSDT after backing up the BIDS project files.
Analysis Services Tabular	New capability in 2012.	Project must be designed in SSDT or imported from a Power Pivot workbook.
Reporting Services	Service architecture and design environment are similar.	Report definitions haven't changed. Re-deploy from SSDT or perform an in-place upgrade after backing up the BIDS project.
Database Project	Project type didn't exist in BIDS. Significant changes from prior Visual Studio 2010 database projects.	Backup the original project and database. In SSDT, use the Convert to SQL Server Database Project wizard to upgrade and then test the new project before deployment.

When planning to upgrade multiple projects of different types that were created using SQL Server 2008 or SQL Server 2008 R2, be mindful that the upgrade story for these

different SQL Server services is somewhat different. Each project type (SSIS, SSAS, and SSRS) may be upgraded to the SQL Server 2014 project type, but the upgrade path for the individual files within those projects may be different. Integration Services object definition files have changed significantly compared to SSAS and SSRS object definition files, and in some cases, these files do not need to be upgraded. For those that do require an upgrade, SSDT will prompt you to upgrade files created in BIDS for SQL Server 2008 and 2008 R2 and will update those files to the appropriate version.

To convert older BIDS or SSDT projects, open them from Visual Studio. You can't depend on Windows to open projects for conversion from the file system. When a BIDS/Visual Studio 2008 project or solution is opened from Windows Explorer, it will simply open in that version of the BIDS designer. To upgrade the project, start SSDT and then open the solution or project from the Visual Studio File menu. Figure 8 shows the Visual Studio project conversion window displayed when an SSRS 2008 R2 report project is opened in SSDT for Visual Studio 2013.

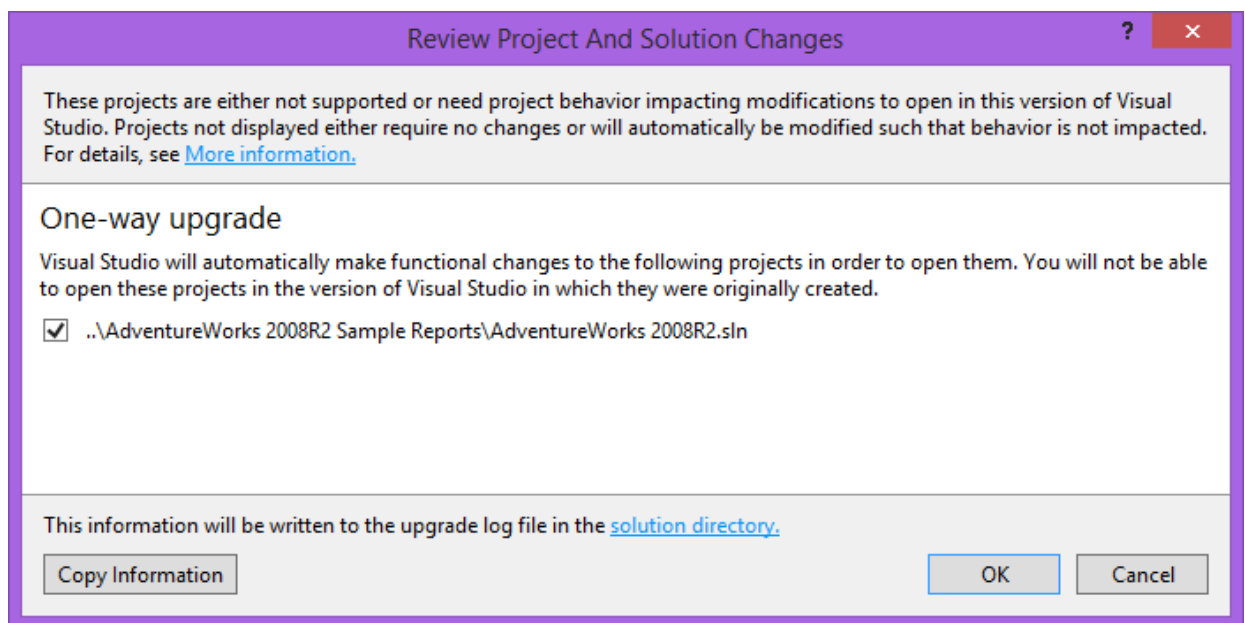


Figure 8: Review Project And Solution Changes window in Visual Studio

After conversion is complete, the wizard adds the UpgradeLog.XML file to the project folder. You can open this file in Internet Explorer to view details about the project upgrade, as shown in Figure 9.

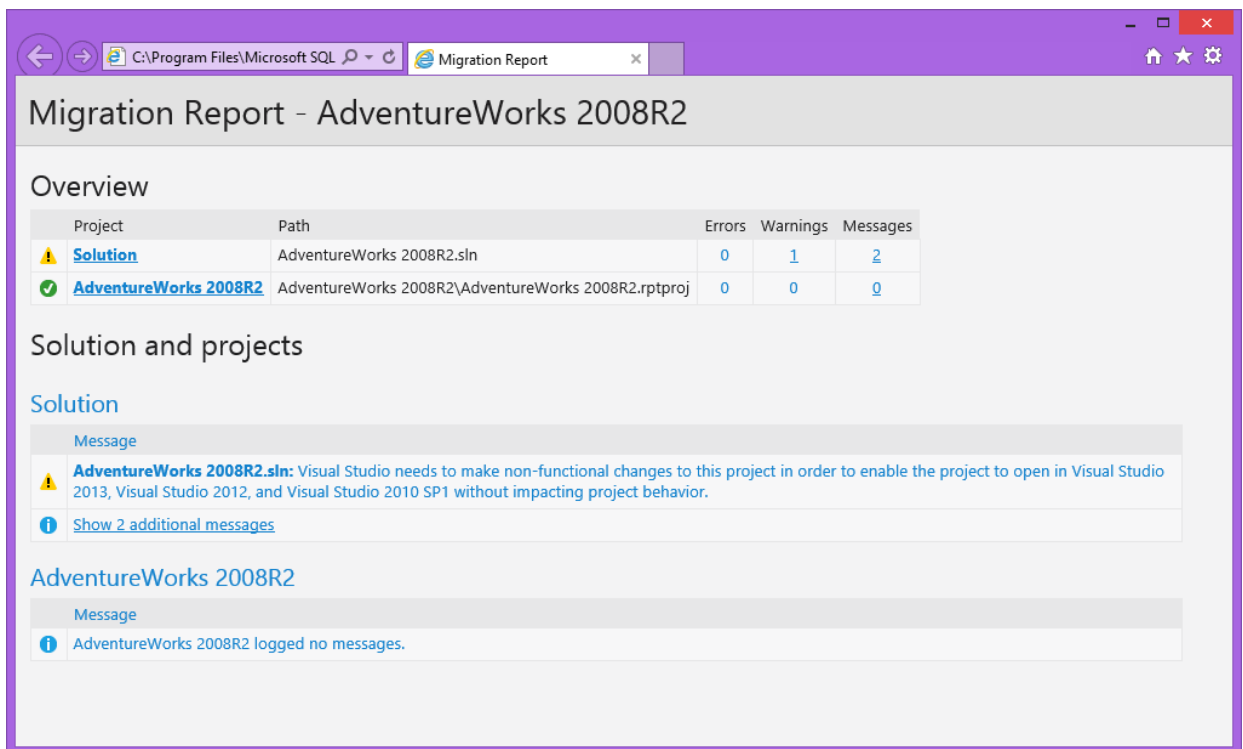


Figure 9: Conversation report

Post Upgrade Tasks and Breaking Changes

The specific post upgrade tasks for each project type and breaking changes for these projects are all covered in the respective chapters for each tool. In summary, consider the following restrictions and considerations:

- Visual Studio 2010 database projects converted to SSDT 2012 database projects will not include:
 - Properties such as Server.sqlsettings and SQLCMD variables defined in .sqlcmd files
 - Partial projects
 - Unit test projects
 - Data generation files
 - Extensibility files
- Some SSIS packages may contain properties that will not be compatible with the new project deployment model. When converting these packages, the Convert to Package Deployment Model dialog box will analyze compatibility and inform you about any restrictions.

- SSAS Tabular Model projects don't have all of the capabilities of SSAS Multidimensional Model projects. The differences between these tools and architectures are extensive. Consider these differences if you plan to replace multidimensional cube projects with tabular model projects.
- No conversion options exist for migrating multidimensional projects to tabular projects.

Additional References

The following resources may be helpful to plan and configure the tools for your SQL Server 2014 upgrade:

- [Checklists for Installing BI Features with SharePoint](http://msdn.microsoft.com/en-us/library/hh231668(v=sql.120).aspx)
([http://msdn.microsoft.com/en-us/library/hh231668\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/hh231668(v=sql.120).aspx))
- [Supported Version and Edition Upgrades](http://msdn.microsoft.com/en-us/library/ms143393(v=sql.120).aspx)
([http://msdn.microsoft.com/en-us/library/ms143393\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143393(v=sql.120).aspx))

Configuration guides, troubleshooting advice, and other in-depth articles and papers related to SQL Server and SharePoint BI tools and architecture are available from the [SQL Server Customer Advisory Team web site](http://sqlcat.com) (<http://sqlcat.com>).

Chapter 16: Analysis Services

Introduction

SQL Server Analysis Services (SSAS) provides a powerful query and calculation engine for building sophisticated business intelligence 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 expanded 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.

SQL Server 2008 R2 did not place the focus on SSAS specifically. There were not a significant number of new SSAS features in SQL Server 2008 R2 compared to SQL Server 2008.

With SQL Server 2012, SSAS was significantly upgraded with the addition of the tabular cube model. SSAS 2012 also introduced the Business Intelligence Semantic Model (BISM), subsuming the earlier Unified Dimensional Model (UDM).

With few exceptions, SQL Server 2014 Analysis Services is unchanged from the previous release. One notable exception is that Power View (surfacing in SharePoint or Excel) now supports SSAS multidimensional models.

When upgrading to SSAS 2014 from versions prior to SQL Server 2012 (i.e., SSAS 2005, 2008, or 2008 R2), you are able to choose whether to continue using the multidimensional model or to switch to the tabular model. You cannot specifically upgrade a multidimensional instance to tabular mode, but you can install a new instance of SSAS that supports tabular model solutions and run both instances side by side. In other words, you can upgrade an instance of the multidimensional model, or you can add a new instance of the tabular model. For more information, see [Install Analysis Services in Tabular Mode](http://msdn.microsoft.com/en-us/library/hh231722(v=sql.120).aspx) (http://msdn.microsoft.com/en-us/library/hh231722(v=sql.120).aspx).

For information on how to upgrade data mining models, see Chapter 19, "Data Mining."

For information on how to upgrade Power Pivot, see:

- Chapter 15, "Business Intelligence Tools"
- [Upgrade PowerPivot for SharePoint](http://msdn.microsoft.com/en-us/library/ee210646.aspx) (http://msdn.microsoft.com/en-us/library/ee210646.aspx)
- [Upgrade Power Pivot Data Models to Excel 2013](http://office.microsoft.com/en-us/excel-help/upgrade-power-pivot-data-models-to-excel-2013-HA103356104.aspx) (http://office.microsoft.com/en-us/excel-help/upgrade-power-pivot-data-models-to-excel-2013-HA103356104.aspx).

Preparing to Upgrade: In-Place Upgrade vs. Side-by-Side Upgrade

You can upgrade SSAS 2005/2008/2008 R2/2012 to SSAS 2014 in multidimensional mode using either the in-place or side-by-side method. The SQL Server 2014 Upgrade Advisor inspects all versions of SQL Server from 2005 through 2012 for any potential upgrade issues.

In-Place Upgrade

With an in-place upgrade, the SSAS 2005/2008/2008 R2/2012 engine and associated tools are removed and replaced by SSAS 2014. During the upgrade process, the older SSAS database metadata is moved to SSAS 2014, and the upgraded databases do not need to be reprocessed.

After the in-place upgrade is complete, only SSAS 2014 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 an earlier version of SSAS after an upgrade to SSAS 2014, you have to uninstall SSAS 2014, restart, reinstall the older version of SSAS, and restore the SSAS databases. Downtime because of upgrade problems can be significant and backups of the existing SSAS databases are critical.

Side-by-Side Upgrade

With a side-by-side upgrade, you install an instance of SSAS 2014 alongside the existing version of SSAS, which remains until separately uninstalled. During the upgrade process, users can continue to access the databases in the older versions of SSAS, which are unaffected by the upgrade process. Once a side-by-side upgrade has completed, both the previous version of SSAS and SSAS 2014 are installed. You can move and test database metadata without affecting the previous SSAS installation. After SSAS 2014 is fully tested, the older version of SSAS 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 2014 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.

Be aware that you cannot do a side-by-side upgrade of Power Pivot for SharePoint on the same machine. It has a hardcoded instance name, so if you tried to install a new instance, you would get an error. You can put it on a different machine as a new instance.

Both upgrade options will result in SSAS 2014 versions of the databases from a given instance of SSAS 2005/2008/2008 R2/2012.

Preparing to Upgrade: 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 2005/2008/2008 R2/2012 to SSAS 2014, there are potential issues that you might face during an upgrade. Before you start an upgrade, you should run the SQL Server 2014 Upgrade Advisor to obtain a report that identifies many of these potential issues. Upgrade Advisor will analyze databases on an existing instance of SSAS 2005, 2008, 2008 R2, or 2012.

If Upgrade Advisor reports any issues, follow its recommendations and guidance for possible mitigation options and strategies. For more information about how to install and run this tool, see Chapter 1, “Upgrade Planning and Deployment.” Chapter 1 also lists those issues that cannot be detected by Upgrade Advisor or whose detection would result in too many false-positive results.

The following sections discuss the most important upgrade issues. For a complete list of backward-compatibility issues, breaking changes, and behavior changes when moving to SSAS 2014, see [Analysis Services Backward Compatibility](http://technet.microsoft.com/en-us/library/ms143479(v=sql.120).aspx) ([http://technet.microsoft.com/en-us/library/ms143479\(v=sql.120\).aspx](http://technet.microsoft.com/en-us/library/ms143479(v=sql.120).aspx)).

Note: For a list of discontinued, deprecated, behavior, and breaking changes when upgrading from SSAS 2005, see Chapter 11, “Analysis Services,” in the [SQL Server 2008 R2 Upgrade Technical Reference Guide](http://download.microsoft.com/download/3/0/D/30DB8D46-8ACF-442A-99A2-0F4CE74AE14D/SQL_Server_2008_R2_Upgrade_Technical_Reference_Guide.docx) (http://download.microsoft.com/download/3/0/D/30DB8D46-8ACF-442A-99A2-0F4CE74AE14D/SQL_Server_2008_R2_Upgrade_Technical_Reference_Guide.docx).

Deprecated Features

Table 1 describes the SSAS objects and settings that are deprecated in SQL Server 2014, which means that they will not be supported in future releases of SQL Server.

Table 1: Deprecated Objects and Settings

Deprecated Feature/Functionality	Comments
CalculationPassValue function	This function is no longer needed, so there is no replacement. The OLAP engine manages the calculation pass.
CalculationCurrentPass function	This function is no longer needed, so there is no replacement. The OLAP engine manages the calculation pass.
NON_EMPTY_BEHAVIOR query optimizer hint turned on by default	The NON_EMPTY_BEHAVIOR query optimizer hint will be turned off by default in a future release. This is an MDX optimization hint that can produce incorrect results when not used correctly.
CELL_EVALUATION_LIST intrinsic cell property	Originally provided a list of evaluated formulas that apply to a cell. It is blank in this release of SSAS. Solve order is now specified in MDX script. For more information, see Understanding Pass Order and Solve Order (MDX) (http://technet.microsoft.com/en-us/library/ms145539.aspx).
COM assemblies	COM assemblies can pose a security risk. Support for COM assemblies will be removed in a future release.
Remote partitions	There is no replacement. Use local partitions instead. See Create and Manage a Local Partition (Analysis Services) (http://technet.microsoft.com/en-us/library/ms175611.aspx) for more information.
Remote linked measure groups	There is no replacement. Use local linked measure groups instead. See Linked Measure Groups (http://technet.microsoft.com/en-us/library/ms174899.aspx) for more information.
Dimensional writeback	There is no replacement. Use partition writeback if you require writeback capability. See Set Partition Writeback (http://technet.microsoft.com/en-us/library/ms174823.aspx) for more information.
Linked dimensions	There is no replacement. Consider copying dimensions to additional models rather than linking to a dimension in another model.

For more information about deprecated features in SSAS 2014, see [Deprecated Analysis Services Functionality in SQL Server 2014](http://technet.microsoft.com/en-us/library/ms143346.aspx) (http://technet.microsoft.com/en-us/library/ms143346.aspx).

Discontinued Functionality

Table 2 lists the features no longer available in SSAS 2014.

Table 2: Discontinued Functionality

Discontinued Feature	Comments
InsertInto connection string property (Local cubes)	The original connection string syntax for populating local cubes is replaced by the Create Global Cube statement. For more information, see CREATE GLOBAL CUBE Statement (MDX) (http://technet.microsoft.com/en-us/library/ms145581.aspx).
CreateCube connection string property (Local cubes)	The original connection string syntax for populating local cubes is replaced by the Create Global Cube statement. For more information, see CREATE GLOBAL CUBE Statement (MDX) (http://technet.microsoft.com/en-us/library/ms145581.aspx).
SQL Server 2000 PMML (Data Mining)	The SQL Server 2000 PMML feature produced a form of PMML that had proprietary extensions to support unique features provided by data mining algorithms that were not available in the PMML specification. In SQL Server 2005, SSAS updated the PMML feature to the newer PMML 2.1 standard. As a result, the proprietary extensions added in SQL Server 2000 are no longer needed, although they are still supported in this release.
Create Action statement (MDX)	This statement is included for backward compatibility. It is replaced by the Action object. For more information about how to create actions in recent versions of SSAS, see Actions (Analysis Services - Multidimensional Data) (http://technet.microsoft.com/en-us/library/ms174515.aspx).

For more information about these discontinued features in SSAS 2014, see [Discontinued Analysis Services Functionality in SQL Server 2014](http://technet.microsoft.com/en-us/library/ms143229.aspx) (http://technet.microsoft.com/en-us/library/ms143229.aspx).

Breaking Changes

There are no breaking changes identified when upgrading from SSAS 2012 to SSAS 2014 at the time of this writing. However, remember to review breaking changes from versions prior to SSAS 2012 if applicable. For more information about breaking changes when upgrading to SSAS 2014, see [Breaking Changes to Analysis Services Features in SQL Server 2014](http://technet.microsoft.com/en-us/library/ms143742.aspx) (http://technet.microsoft.com/en-us/library/ms143742.aspx).

Behavior Changes

There are no behavior changes identified when upgrading from SSAS 2012 to SSAS 2014 at the time of this writing. With respect to upgrading to SSAS 2012 and later, there are only two areas of behavior changes in SSAS 2012. The first area is for the multidimensional mode of SSAS, and the second area is for Power Pivot for SharePoint.

Multidimensional Mode

In SQL Server Management Studio (SSMS) and in Cube Designer, the Cube browser has been removed because it was based on a control that was an Office Web Control (OWC) component. OWC was deprecated by Office and is no longer available. In its place, a new browser is available that flattens out queries to only rows, similar to the MDX query designer in the SQL Server Reporting Services Report Designer.

Power Pivot for SharePoint

Higher permission requirements for using a Power Pivot workbook as an external data source are required with SSAS 2012 and SSAS 2014. With these releases, permission requirements have changed for Excel workbooks that render Power Pivot data from an external file. You must have Read permissions (or more specifically, the Open Items permission) to connect to an external Power Pivot workbook from a client application. The additional permissions specify that a user has download rights to view the source data embedded in the workbook. The additional permissions reflect the fact that model data is wholly available to the client application or workbook that links to it, resulting in a better alignment between permission requirements and the actual data connection behavior.

To continue using a Power Pivot workbook as an external data source, you must increase SharePoint permissions for users who connect to external Power Pivot data. Until you change the permissions, users will get the following error if they try to access Power Pivot workbooks in a data source connection: *"PowerPivot Web service returned an error (Access denied. The document you requested does not exist or you do not have permission to open the file.)"*

For more information about the behavior changes in SSAS 2012 and SSAS 2014, see [Behavior Changes to Analysis Services Features in SQL Server 2014](http://technet.microsoft.com/en-us/library/ms143682.aspx) (<http://technet.microsoft.com/en-us/library/ms143682.aspx>).

64-bit Considerations

SSAS 2005/2008/2008 R2/2012/2014 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. Accordingly, the 64-bit edition of SSAS 2012 or earlier should be upgraded to the 64-bit edition of SSAS 2014.

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 32-bit edition of SSAS 2005 to the 64-bit edition of SSAS 2014.

Upgrading from SSAS 2005, 2008, 2008 R2, or 2012

Upgrading to SSAS 2014 from SSAS 2005 through SSAS 2012 is typically a straightforward process. Although one of the goals of the SSAS 2014 team was to avoid any breaking changes when upgrading from previous versions, check the list of changes in this chapter to ensure that no changes will adversely affect an upgraded database. If your current SSAS 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 Best Practices Analyzer (BPA) for SQL Server to aid in upgrade preparation. You run this diagnostic utility against the SQL Server 2005/2008/2008 R2/2012 instances being upgraded. For more information about these upgrade tools, see Chapter 1, "Upgrade Planning and Deployment."

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 SQL Server upgrade, automatically upgrading associated metadata for each database found in SSAS 2005/2008/2008 R2/2012.

If a failed upgrade occurs, frequently the easiest resolution is to reinstall the previous version of SSAS and restore the installation to its state before the upgrade process was started. Back up all databases by using the Back Up command in SSMS. To do this,

open SSMS, right-click each database that is listed, and select Back Up. Provide a unique filename 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.

Side-By-Side Upgrade

As mentioned previously, in a side-by-side upgrade, you install an instance of SSAS 2014 alongside the existing version of SSAS. After installing SSAS 2014, users can continue to access the databases in the older versions of SSAS. After installing the multidimensional SSAS 2014 engine as a new named instance, you can move SSAS databases to it using one of the following methods:

- Back up the database from the older version of SSAS and restore it to the SSAS 2014 instance.
- Detach the database from the older version of SSAS and attach it to the SSAS 2014 instance.
- Open the project files in a supported version of Visual Studio and deploy the database to SSAS 2014. In this case, the cube will need to be processed, unlike the previous two options.

In-Place Upgrade

To start the in-place upgrade, start the Setup application for SQL Server 2014, selecting Installation and then *Upgrade from SQL Server 2005, SQL Server 2008, SQL Server 2008 R2 or SQL Server 2012*. The Setup program will run a system configuration check, collect system information, and prompt for a product key. After it installs any necessary setup files, the Setup application will then prompt you to select the instance to upgrade.

The next screen shows the features to be installed. All the installed components are selected and no changes can be made. If it is impossible to select different options, you will need to run Setup again to add features.

After you have selected the components to install, the Setup application will prompt you for an instance name for the newly installed SQL Server 2014 components. To upgrade an existing installation of SSAS, 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 is selected, and continue.

After you select the instance name, Setup will check for the necessary disk space. The Setup application will then check the server configuration, check the full-text search upgrade option, and prompt the user to turn on Error and Usage Reporting. This is an optional setting. Enabling it 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. When there are no failures, 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 along with any notes that are relevant to the upgrade process.

After the upgrade process is finished, you should perform a short set of post-installation tasks to ensure that the upgrade completed successfully. See the “Post-Upgrade Tasks” topic later in this section 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\120\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\120\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 server to SSAS 2014 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 necessary.

Review upgraded databases. Each database that was upgraded by the SQL Server 2014 Setup application should be reviewed to ensure that the upgrade process completed successfully. Using SSMS, connect to SSAS on the upgraded server. If the workstation components were installed as part of the upgrade, SSMS should be available on the upgraded server; otherwise, SSMS will have to be started on another server or workstation that has the workstation components for SQL Server 2014 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. The cubes that are upgraded from SSAS 2005/2008/2008 R2/2012 do not have to be processed to be browsed.

In SSAS, the database CompatibilityLevel property determines whether certain data operations are available based on a particular version of the server. For example, setting the database compatibility level property to 1100 enables the use of the scalable string storage feature in SQL Server 2014.

Valid values for the CompatibilityLevel property include the following:

- 1050. This value is set when you attach or restore a SSAS database that was created in SQL Server 2005/2008/2008 R2/2012 on a SQL Server 2014 SSAS server. The CompatibilityLevel property is automatically added when you attach or restore the database.
- 1100. This is the default value for new SSAS databases that you create in SQL Server 2012 or SQL Server 2014. You can also specify it for databases created in earlier versions of SSAS to enable the use of features that are supported only at this compatibility level (namely, scalable string storage for dimension attributes or distinct count measures that contain string data).

Setting the database compatibility to a higher level is irreversible. After you increase the compatibility level to 1100, you must continue to run the database on newer servers. You cannot rollback to 1050, nor can you attach or restore an 1100 database on a server version that is earlier than SQL Server 2012. Be sure to back up the database in the event you want to use the previous version on an earlier version of SSAS. For more information, see [Set the Compatibility Level of a Multidimensional Database \(Analysis Services\)](http://msdn.microsoft.com/en-us/library/gg471593.aspx) (<http://msdn.microsoft.com/en-us/library/gg471593.aspx>).

BIDS/SSDT Project Upgrades

SSAS projects from Business Intelligence Development Studio (BIDS) 2005, BIDS 2008, and BIDS 2008 R2 can be opened in Visual Studio 2012 or Visual Studio 2013. Keep in mind that the SSIS/SSRS/SSAS 'Templates' are now a separate download. For more information, see [Microsoft SQL Server Data Tools - Business Intelligence for Visual Studio 2013](http://www.microsoft.com/en-us/download/confirmation.aspx?id=42313) (<http://www.microsoft.com/en-us/download/confirmation.aspx?id=42313>). This process will require a Visual Studio project upgrade, so it is strongly recommended that you first back up the BIDS 2005/2008/2008 R2 projects if they will need to be subsequently re-opened in BIDS 2005/2008/2008 R2.

Conversely, SQL Server Data Tools (SSDT) projects based on SSAS 2012 can be opened with the SQL Server 2014 templates in Visual Studio 2012 or Visual Studio 2013 without upgrading. Note that since the release of Visual Studio 2013, Microsoft no longer offers updates for SSDT through the Visual Studio 2010 shell. To get updates, users will need to move to Visual Studio 2012 or Visual Studio 2013.

Conclusion

Upgrading to SSAS 2014 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 provides for two versions of SSAS run at the same time, with SSAS 2014 being a named instance. When the earlier version of SSAS is removed, the SSAS 2014 version can be changed to the default instance on the server.

Moving from SSAS 2005/2008/2008 R2/2012 to SSAS 2014 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 from versions prior to SSAS 2012, a redesign is not necessary, and the overall design of multi-dimensional cubes and dimensions has not changed.

Additional References

For an up-to-date collection of additional references for upgrading to SQL Server 2014, see the following links:

- [Business intelligence](http://www.microsoft.com/sqlserver/en/us/solutions-technologies/business-intelligence/analysis-services.aspx)
(<http://www.microsoft.com/sqlserver/en/us/solutions-technologies/business-intelligence/analysis-services.aspx>)
- [SQL Server 2014 Web Site](http://www.microsoft.com/sqlserver/en/us/default.aspx)
(<http://www.microsoft.com/sqlserver/en/us/default.aspx>)
- [Books Online for SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms130214(v=sql.120).aspx)
([http://msdn.microsoft.com/en-us/library/ms130214\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms130214(v=sql.120).aspx))
- [SQL Server Developer Center](http://msdn.microsoft.com/en-us/sqlserver)
(<http://msdn.microsoft.com/en-us/sqlserver>)
- [SQL Server TechNet Resources](http://technet.microsoft.com/en-us/sqlserver)
(<http://technet.microsoft.com/en-us/sqlserver>)

Chapter 17: Integration Services

Introduction

This chapter is addressed to existing SQL Server customers who are interested in upgrading SQL Server Integration Services (SSIS) solutions developed in SQL Server 2005, SQL Server 2008, SQL Server 2008 R2, and SQL Server 2012 to SQL Server 2014.

Although SSIS 2014 is unchanged from the previous release, SSIS 2012 provided new functionality for both for developers and administrators. This chapter will cover some of those new features as well as provide pointers to SSIS 2014 content on the web.

However, the primary focus is to assist existing customers with their upgrade to SSIS 2014. To that end, this chapter is organized into the following sections:

- “Preparing to Upgrade to SSIS 2014.” This section discusses the considerations and steps required prior to an SSIS 2014 upgrade.
- “SSIS Sample Package Overview.” This section provides an overview of the sample packages referenced in this chapter.
- “Running Upgrade Advisor.” This section shows you how to use the SQL Server 2014 Upgrade Advisor to find out if there are any issues you need to address prior to your upgrade.
- “Installing SSIS 2014.” This section discusses in-place upgrades, side-by-side upgrades, and new installation options.
- “Project Conversion Wizard.” This section shows you how to use the SSIS 2014 Project Conversion Wizard to upgrade SQL Server packages. It also introduces you to the configuration, deployment, and management features in the project deployment model, which was introduced in SSIS 2012.
- “Additional References.” This section contains useful online links for this topic.

To learn more about the features introduced in SSIS 2012, see:

- [What's New \(Integration Services\)](http://msdn.microsoft.com/en-us/library/bb522534(v=sql.110).aspx) (http://msdn.microsoft.com/en-us/library/bb522534(v=sql.110).aspx)
- [SSIS MSDN web site](http://msdn.microsoft.com/en-us/sqlserver/cc511477.aspx) (http://msdn.microsoft.com/en-us/sqlserver/cc511477.aspx)

Preparing to Upgrade to SSIS 2014

Customers upgrading to SSIS 2014 from SSIS 2005, SSIS 2008, SSIS 2008 R2, or SSIS 2012 have multiple installation options available, including upgrading in-place,

performing a side-by-side installation, and performing a new installation. These are covered later in the “Installing SSIS 2014” section.

In preparation for the upgrade, plan on performing the following steps for your SSIS upgrade:

1. Migrate all of your Data Transformation Services (DTS) packages to SSIS packages.
2. Run Upgrade Advisor, which will alert you to any issues that may need to be addressed prior to the upgrade.
3. Perform the SQL Server installation/upgrade process of which SSIS is a component.
4. Run the SSIS 2014 Project Conversion Wizard to upgrade your existing SSIS packages to SSIS 2014.
5. Perform post upgrade steps, optionally converting to SSIS 2014's project deployment model (introduced in SSIS 2012).

Even though converting to the project deployment model is optional, it is highly recommended that you at least review the capabilities of this new feature. The SSIS 2014 server leverages the project deployment model, which makes it easier to configure, deploy, and manage SSIS packages.

In preparation for your upgrade, it is first useful to review SSIS 2014 support for backward compatibility, especially the discontinued features, which include DTS.

SSIS Backward Compatibility

As mentioned earlier, SSIS 2014 is unchanged from the previous release. However, if you are upgrading from a version prior to SSIS 2012, it is highly recommended that you review SSIS's backward compatibility.

[Integration Services Backward Compatibility](http://msdn.microsoft.com/en-us/library/ms143706(v=sql.110).aspx) ([http://msdn.microsoft.com/en-us/library/ms143706\(v=sql.110\).aspx](http://msdn.microsoft.com/en-us/library/ms143706(v=sql.110).aspx)) contains the most recent information on SSIS deprecated features, discontinued features, breaking changes, and behavior changes. The key items to mention here are two discontinued features: DTS and ActiveX scripting.

DTS has been discontinued and is no longer supported in SSIS 2012 and SSIS 2014. DTS packages will need to be migrated to SSIS packages prior to upgrading to SQL Server 2014. For more information on this topic, see [Considerations for Upgrading Data](#)

[Transformation Services](http://msdn.microsoft.com/en-us/library/ms143716(v=SQL.105).aspx) ([http://msdn.microsoft.com/en-us/library/ms143716\(v=SQL.105\).aspx](http://msdn.microsoft.com/en-us/library/ms143716(v=SQL.105).aspx)).

The [SQL Server 2008 R2 Upgrade Technical Reference Guide](http://download.microsoft.com/download/3/0/D/30DB8D46-8ACF-442A-99A2-0F4CE74AE14D/SQL_Server_2008_R2_Upgrade_Technical_Reference_Guide.docx) (http://download.microsoft.com/download/3/0/D/30DB8D46-8ACF-442A-99A2-0F4CE74AE14D/SQL_Server_2008_R2_Upgrade_Technical_Reference_Guide.docx) also contains information on how to migrate DTS packages to SSIS, including how to use the DTS Package Migration Wizard. In addition, [Migrating Data Transformation Services Packages](http://msdn.microsoft.com/en-us/library/ms143501(v=SQL.105).aspx) ([http://msdn.microsoft.com/en-us/library/ms143501\(v=SQL.105\).aspx](http://msdn.microsoft.com/en-us/library/ms143501(v=SQL.105).aspx)) in SQL Server 2008 R2 Books Online has more information on the DTS Package Migration Wizard.

Finally, there are third-party products such as [DTS xChange](http://pragmaticworks.com/Products/Business-Intelligence/DTSxChange/Default.aspx) (<http://pragmaticworks.com/Products/Business-Intelligence/DTSxChange/Default.aspx>) that can be used to convert your DTS packages to SSIS packages.

ActiveX scripting has also been discontinued with the SSIS 2012 and SSIS 2014 releases. Existing ActiveX scripting tasks will need to be rewritten using a combination of SSIS tasks (including the Script task) and task control flow.

Note that you can choose a side-by-side installation if you still have some DTS packages that haven't been converted yet. However, at some point in time, these DTS packages will need to be migrated to SSIS. In addition, a side-by-side installation creates a more complex environment with multiple versions of SQL Server, SQL Server Data Tools (SSDT), and/or Business Intelligence Development Studio (BIDS) residing on one server.

64-Bit Considerations

As the computing world moves to 64-bit systems, 64-bit vs. 32-bit considerations are becoming less relevant. However, there are still some scenarios where running SSIS in 32-bit mode is required.

The most common scenario is when Microsoft Excel is either a source or destination within a dataflow. For more information on this topic, see the following:

- [Importing Data from 64-bit Excel in SSIS](http://hrvoje.piasevoli.com/2010/09/01/importing-data-from-64-bit-excel-in-ssis) (<http://hrvoje.piasevoli.com/2010/09/01/importing-data-from-64-bit-excel-in-ssis>)
- [Excel Error 64-bit version of SSIS](https://social.msdn.microsoft.com/Forums/sqlserver/en-US/289e29ad-26dc-) (<https://social.msdn.microsoft.com/Forums/sqlserver/en-US/289e29ad-26dc->

4f90-bad4-ffb86c76e5f9/excel-error-64bit-version-of-ssis?forum=sqlintegrationservices)

- [Quick Reference: SSIS in 32- and 64-bits](http://toddmcdermid.blogspot.com/2009/10/quick-reference-ssis-in-32-and-64-bits.html)
(<http://toddmcdermid.blogspot.com/2009/10/quick-reference-ssis-in-32-and-64-bits.html>)

For more information on SSIS and 64-bit considerations, see [64-bit Considerations for Integration Services](http://msdn.microsoft.com/en-us/library/ms141766(v=sql.105).aspx) ([http://msdn.microsoft.com/en-us/library/ms141766\(v=sql.105\).aspx](http://msdn.microsoft.com/en-us/library/ms141766(v=sql.105).aspx)).

The next step is to run Upgrade Advisor. Before you do that, though, it is useful to know about some of the details of the sample SSIS solution and packages used in the remainder of this chapter.

SSIS Sample Package Overview

The scenario for this sample is an SSIS solution whose packages combine to build the Geography and Sales Territory dimensions from the [AdventureWorks2008R2 OLTP sample database](http://msftdbprodsamples.codeplex.com/releases/view/59211) (<http://msftdbprodsamples.codeplex.com/releases/view/59211>). This sample consists of a master package and a series of execution packages both layered on a custom Extraction, Transformation, and Loading (ETL) Framework.

ETL Frameworks

Custom ETL Frameworks are common in existing SSIS production solutions, and this sample uses a lightweight version of the [Stonemeadow Solutions ETL Framework](http://etlframework.codeplex.com) (<http://etlframework.codeplex.com>) found on CodePlex. Another example of a custom SSIS ETL Framework is the [BI Monkey SSIS ETL Framework](http://ssisetlframework.codeplex.com) (<http://ssisetlframework.codeplex.com>).

One common pattern for an ETL Framework is to have two types of packages: master packages and execution packages. The master package is responsible for larger deployment and workflow responsibilities. The execution packages contain the ETL code. Note that this pattern fits well within the project deployment model for SSIS 2012 and SSIS 2014 (i.e., a project can be based on one master package and multiple execution packages).

In addition, SSIS 2012 and SSIS 2014 have significant improvements in configurations, logging, and deployments to the point where third-party or custom ETL Frameworks may no longer be necessary. If you are upgrading to SSIS 2014, you should study these new capabilities to decide whether to replace an existing ETL Framework capability

(e.g., configurations) with the SSIS 2014 equivalent. If you decide to continue to using your existing ETL Framework, you can still use the SSIS 2014 Project Conversion Wizard. For new SSIS implementations, you should strongly consider using SSIS 2014's features to meet your ETL Framework needs.

The SQL Server Worldwide Users Group (SSWUG) article [SQL Server 2012 ETL Framework Features in the New SSIS Catalog](http://www.sswug.org/articles/viewarticle.aspx?id=59199)

(<http://www.sswug.org/articles/viewarticle.aspx?id=59199>) provides a list of what types of features are in the SSIS Catalog out-of-the-box. Although this is an unofficial list, it provides a good categorization of ETL Framework features along with what is supported in SSIS 2012 and SSIS 2014.

Master Package

The master package is responsible for:

- **Configuration.** The master package initializes source and destination connection strings and file directories.
- **Batch creation and logging.** The batch represents one instance of the master package invocation and is used for logging as well as creating execution lineage identifiers.
- **Error logging.** The master package has an OnError event that logs error information for every master package error as well as unhandled execution package errors.
- **Workflow.** The sequencing of execution packages, including upstream dependencies (from child packages), occurs in the master package.

Figure 1 shows the task flow for a master package. Five out of the seven tasks in this task flow are Execute Package tasks. The other two tasks are responsible for creating a batch record in the ETL Framework logging database and then updating this batch's status after the execution packages complete.

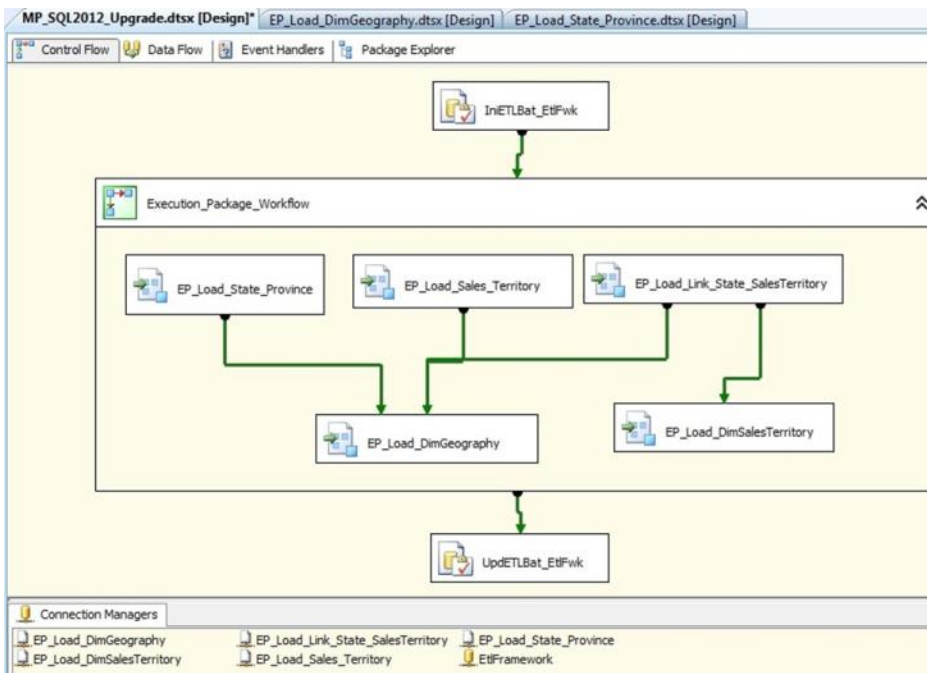


Figure 1: Master package task flow

Note that custom ETL Frameworks may use SSIS Script tasks to populate SSIS variables from configuration information stored in a file or database.

This master package uses SSIS package configurations, shown in Figure 2, to initialize all source, destination, and logging connection strings used by the execution packages as well as the file directory used by expressions to initialize each execution package file connection.

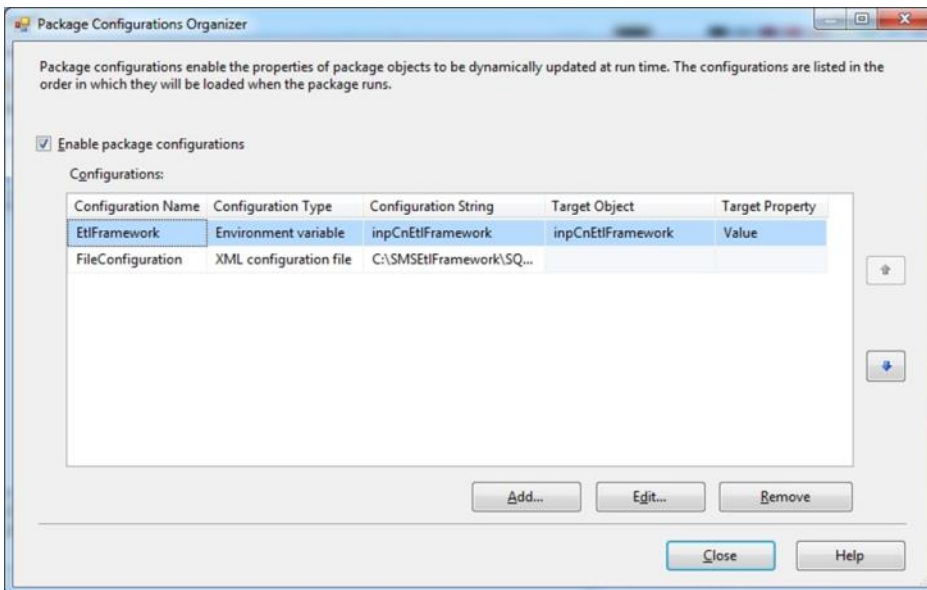


Figure 2: Master package configuration

Note that two different configuration types are used. The ETL Framework logging database is populated by an environment variable. The source and database connection strings as well as the DTSX package directory are loaded by an XML configuration file.

SSIS expressions are heavily used in this sample in support of runtime configurations. These allow the solution to be moved across environments (e.g., development, test, quality assurance, staging, production), without having to open and edit the package. Figure 3 shows the properties for one of the file connections, each of which point to a DTSX package file.

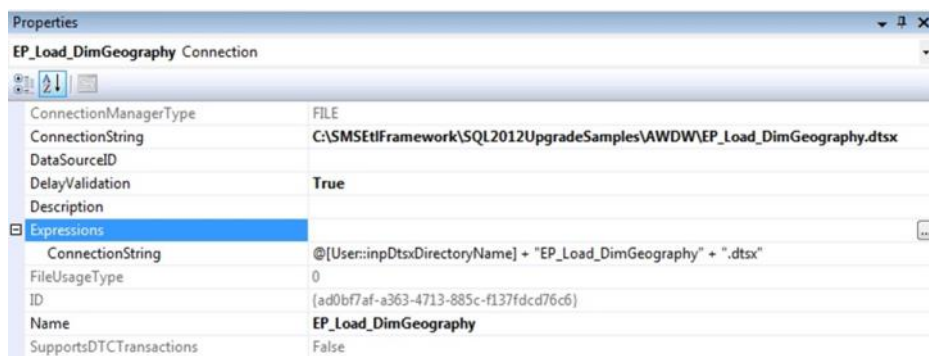


Figure 3: Execute package file connection

Note that the above connection string is created by combining the `inpDtsxDirectoryName` configuration value with the execution package's name. Having SSIS expressions leveraging variables loaded by configurations is a common technique in existing SSIS implementations. Note that this technique also requires that the `DelayValidation` property be set to `True`, as shown in Figure 3.

Execution Packages

The execution packages are responsible for the real ETL work. Each of the five execution packages in this sample follows a similar pattern for both their task flow and data flow. Figure 4 shows an example of this task flow.

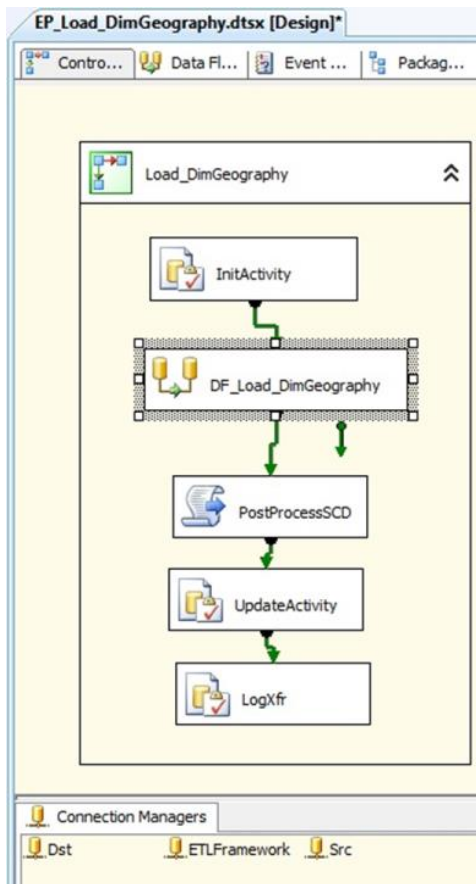


Figure 4: Execute package task flow

Each execution package task flow has:

- **Three logging tasks.** InitActivity, UpdateActivity and LogXfr are SQL tasks that call a stored procedure to insert one or more records into an ETL Framework logging table. Note that the InitActivity task is also responsible for creating the execution lineage ID stored in all destination tables.
- **One data flow.** This data flow implements either a Slowly Changing Dimension I (SCD1) or Slowly Changing Dimension II (SCD2) data flow. Simply put, SCD2 is about versioning every change, while SCD1 updates one record.
- **One SCD Post Process task.** This is a Script task that builds and executes SQL statements used in SCD post-processing activities. For example, an SCD2 (versioned table) will need to update the previous version with an End Date and change the Record status from Active to Inactive.

Note that each of these execution package tasks has the same source, destination, and logging database configurations. This is a common pattern, which is nicely supported by SSIS 2012 and SSIS 2014 Shared Connections.

Each execution packages uses parent package configurations to load configuration parameters at runtime. Figure 5 shows the package configurations common to all execution packages. Notice how all the connection strings and batch instance values are consumed at runtime by the execution packages after they are configured by the master package.

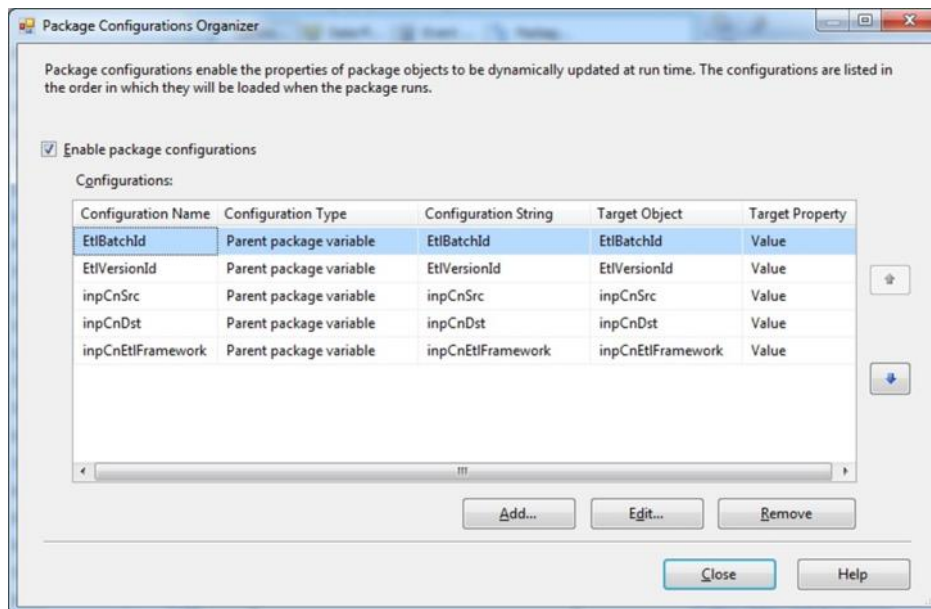


Figure 5: Execute package configuration

Each execution package uses expressions to set their database connection strings at runtime. Figure 6 is a screenshot of the Destination (Dst) database connection's properties. Notice how the inpCnDst variable initialized by a parent package variable is used to set the ConnectionString attribute.

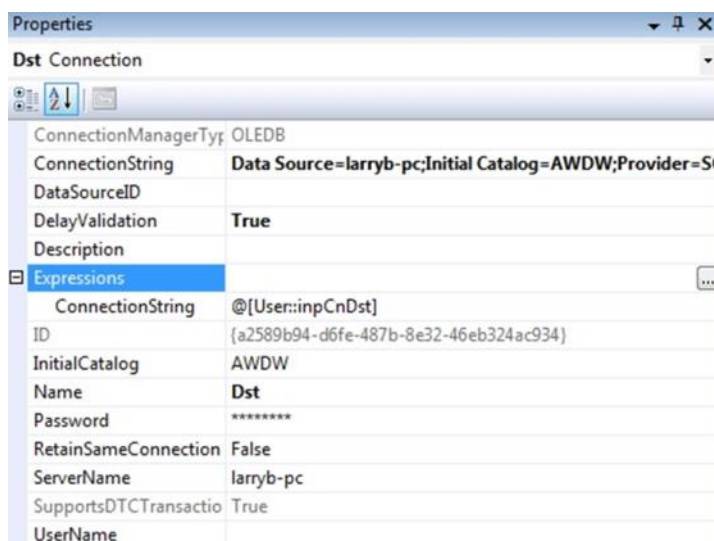


Figure 6: Database connection

Each execution package also has one of two patterns, depending upon whether the destination table is an SCD1 or SCD2. Figure 7 shows an example of an SCD1 data flow.

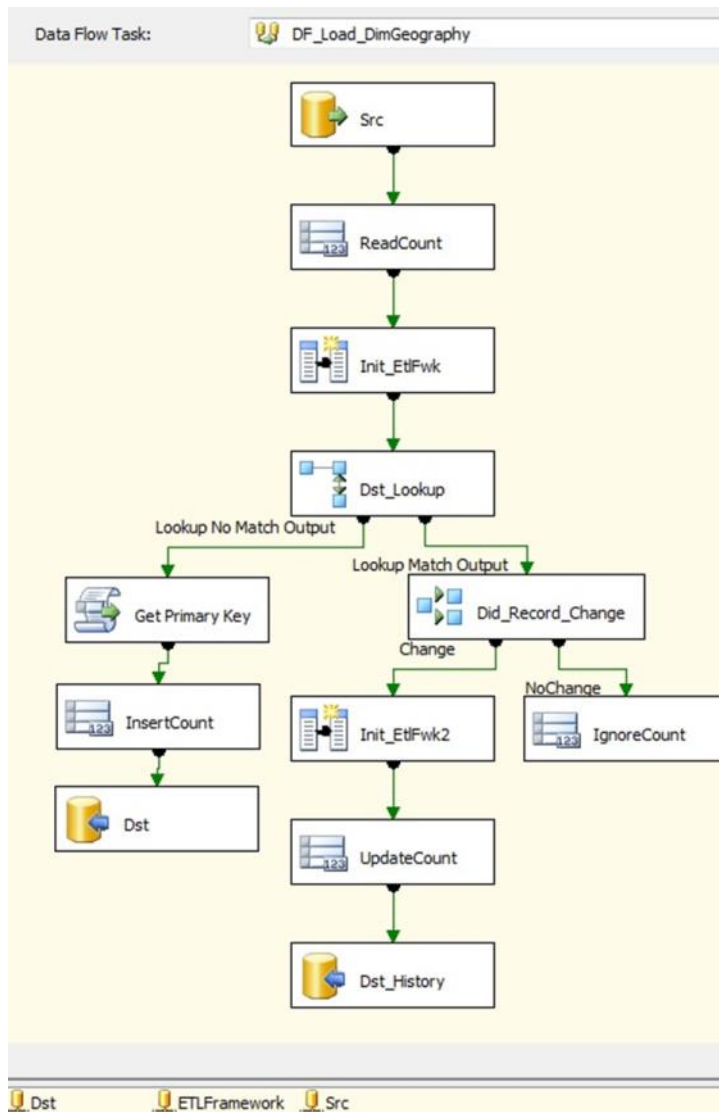


Figure 7: SCD1 data flow

In this data flow, the package does the following:

- The package reads data from the source.
- The package adds record count instrumentation. Note that logging and reporting on record counts is a useful component in an ETL Framework.
- The package issues a lookup to see if the record exists.
- If the record doesn't exist, the package inserts it. Before inserting, the package generates a primary key if SQL Server IDENTITY columns are not in place.

- If the record does exist, the package checks to see if anything has changed in the record.
- If the record has not changed, the package ignores the record.
- If the record changed, the package inserts a record into the History table.

Note that the SCD1 post processing logic will build and execute the UPDATE statement that is applied to the destination SCD1 table.

That concludes the review of the SSIS sample solution that will be converted to SSIS 2014. The first step after planning has completed is to run the Upgrade Advisor wizard.

Running Upgrade Advisor

It is recommended that you run Upgrade Advisor prior to starting the actual SSIS 2014 upgrade. Upgrade Advisor will inspect your SSIS files and inform you if there are any issues that you need to address prior to your upgrade. Note that before you run Upgrade Advisor, you need to install it.

This can be done by selecting the Install Upgrade Advisor option within the SQL Server Installation Center screen, which is invoked by running the Setup.exe application on the SQL Server installation media. Note that you may get the error shown in Figure 8.

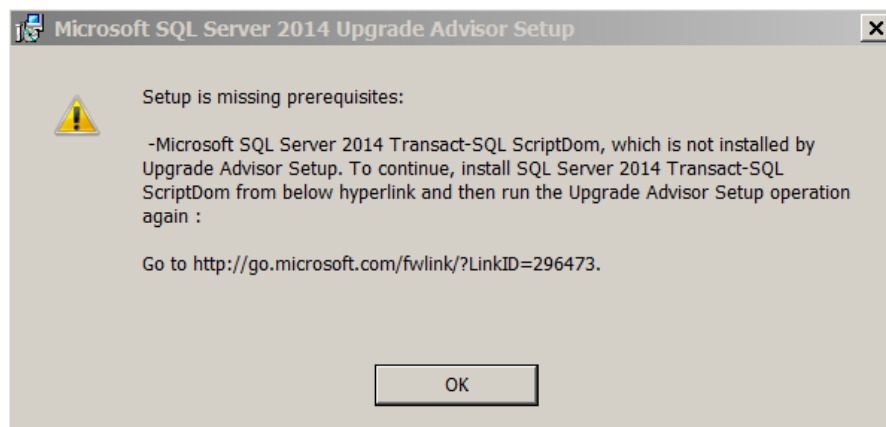


Figure 8: Missing setup prerequisites

The SQL Server Transact-SQL ScriptDom prerequisite can be obtained from the [Microsoft SQL Server 2014 Feature Pack](http://www.microsoft.com/en-us/download/details.aspx?id=42295) (http://www.microsoft.com/en-us/download/details.aspx?id=42295). The link is available in the "Microsoft® SQL Server® 2014 Transact-SQL ScriptDom" section. The file is SQLDOM.MSI.

You can proceed with installing Upgrade Advisor after you install SQLDOM.MSI. For more information on installing Upgrade Advisor, see Chapter 1, "Upgrade Planning and Deployment" in this guide.

After the Upgrade Advisor installation completes, it is available within the SQL Server 2014 program group. You can navigate to this program group by pressing the Start button, and then All Programs. Clicking the Launch Upgrade Advisor Analysis Wizard button and then the Next button gets you to the SQL Server Components screen shown in Figure 9.

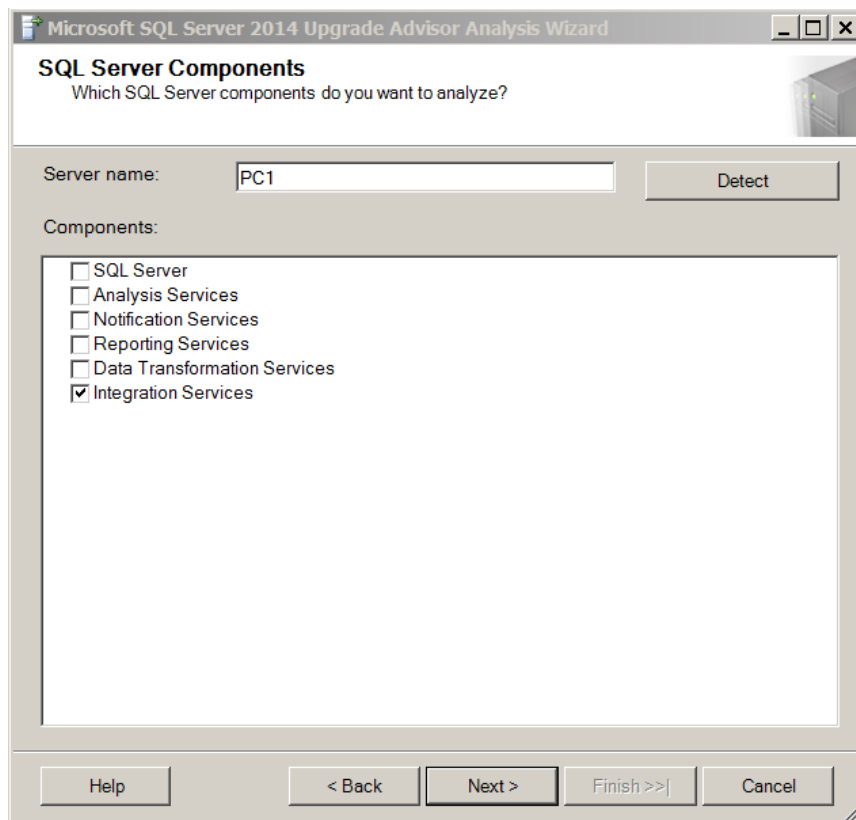


Figure 9: Selecting the SSIS component

Select Integration Services and click the Next button to navigate to the Connection Parameters screen shown in Figure 10. This screen prompts you for the database instance that you are targeting for the conversion.

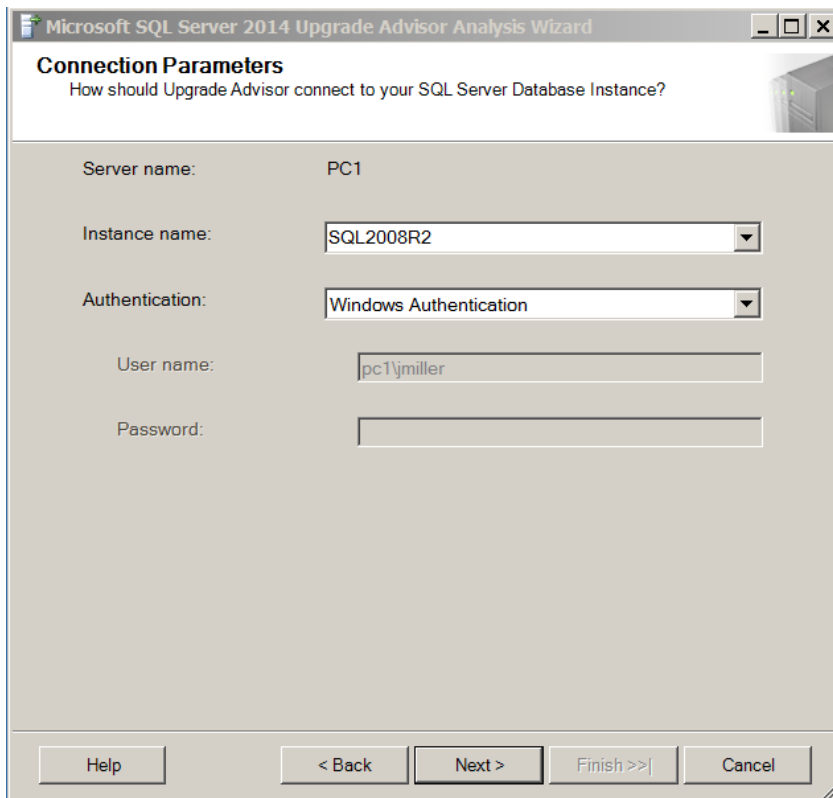


Figure 10: Connecting to the SQL Server instance

Clicking Next brings you to the SSIS Parameters screen shown in Figure 11. This is where you provide the location of your SSIS packages.

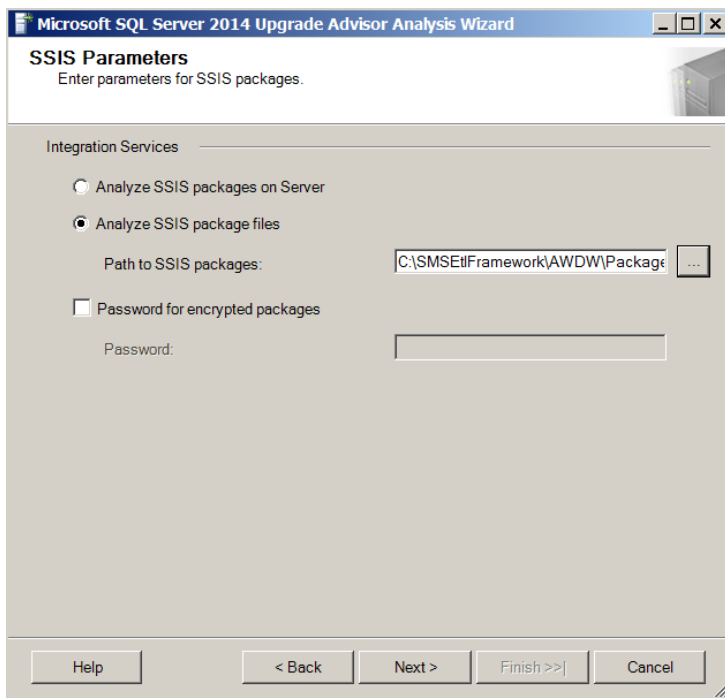


Figure 11: Choosing the location of the SSIS packages

Clicking the Next button gets you to the Upgrade Advisor Settings screen. In this screen, you can verify that the SSIS package location is correct. You should also make a note of the locations of the Upgrade Advisor log and report files.

Clicking Next starts the Upgrade Advisor analysis process. It reports on the progress of the analysis and will indicate the status of the upgrade, as shown in Figure 12.

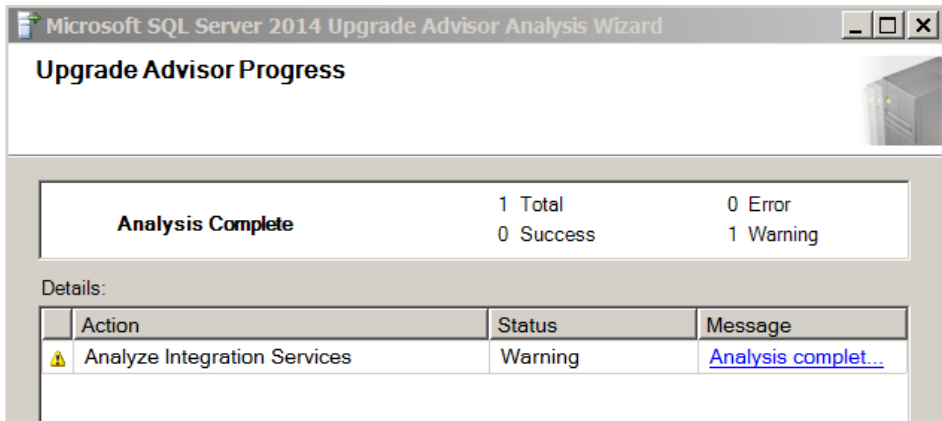


Figure 1: Checking the progress of Upgrade Advisor

Clicking the Launch Report button displays the View Report screen shown in Figure 13.

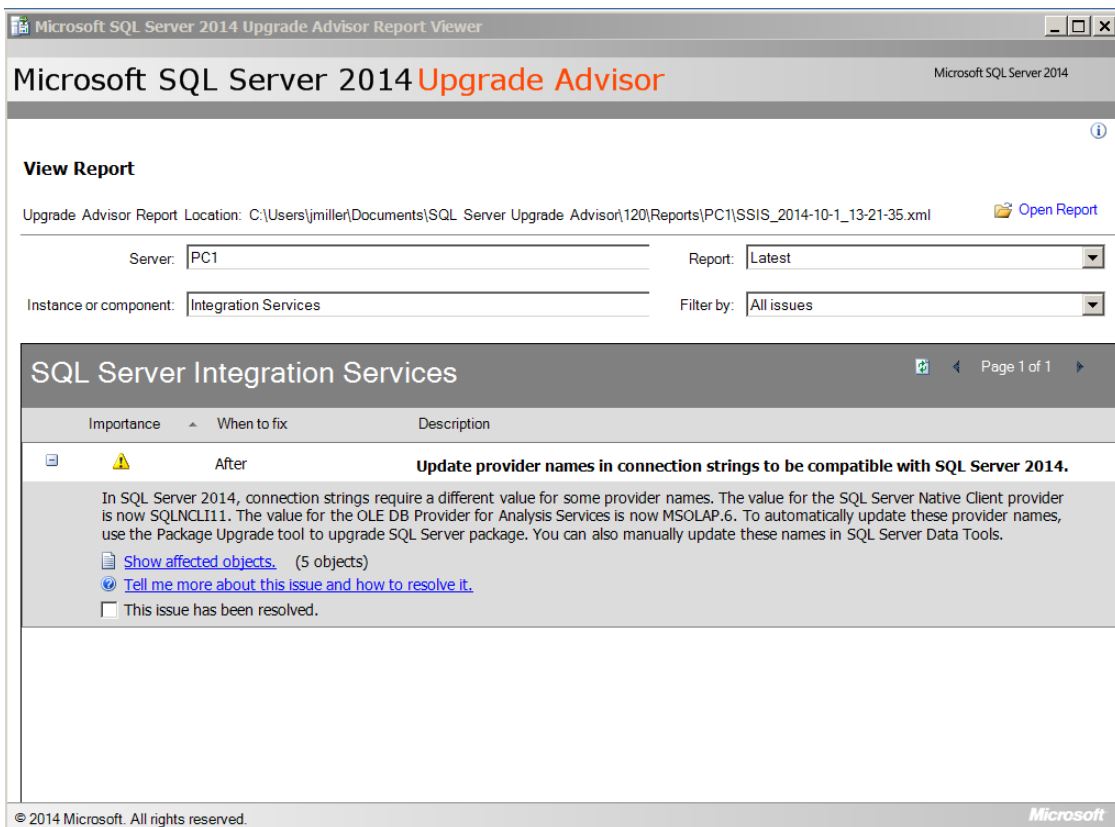


Figure 13: Viewing the Upgrade Advisor report

This concludes the walkthrough for running Upgrade Advisor for SSIS. In summary, it is a good practice to run Upgrade Advisor for your SSIS packages to ensure that you will not have an issue with the post upgrade steps.

After addressing the Upgrade Advisor warnings and errors, the next step is to install SQL Server 2014.

Installing SSIS 2014

You can choose to upgrade the current installation or leave the current installation as is and install either a new instance on the same server or a new server. Figure 14 shows the SSIS related components within the SQL Server 2014 Feature Selection screen. Note that most SSIS 2014 installations will also install the Database Engine at minimum.

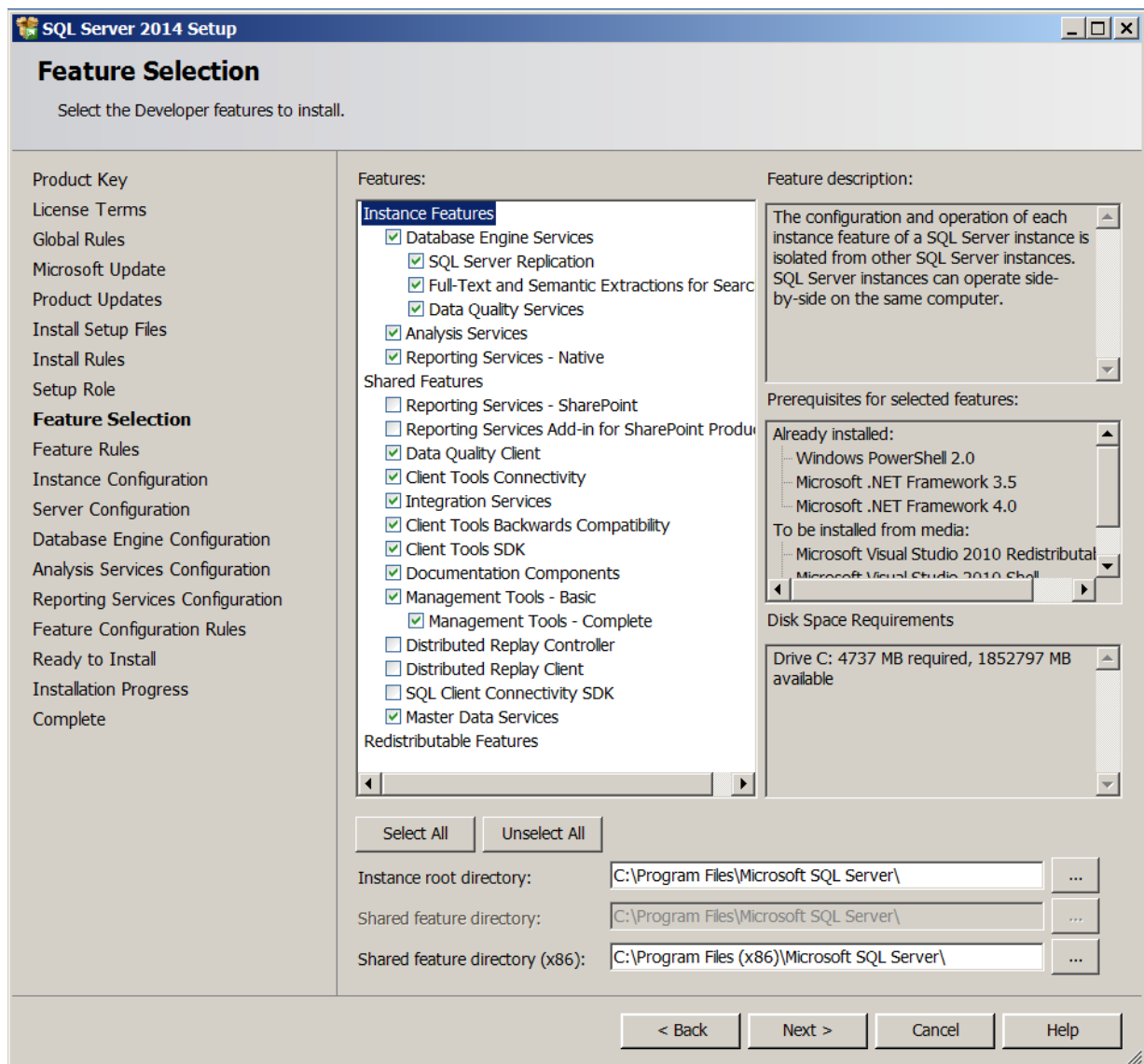


Figure 14: SQL Server 2014 feature selection

Select Integration Services to install the Integration Services service and to run packages outside the design environment.

For a complete installation of Integration Services, together with the tools and documentation for developing and managing packages, select both Integration Services and the following Shared Features:

- **SQL Server Data Tools** to install the tools for designing packages
- **Management Tools - Complete** to install SQL Server Management Studio (SSMS) for managing packages
- **Client Tools SDK** to install managed assemblies for SSIS programming
- **Data Quality Client** to install the Data Quality Services (DQS) client objects. Note that this feature isn't required by SSIS 2014, but provides data scrubbing, cleansing, and transformations that are core to many ETL solutions.

For more information on SSIS 2014 installations, see [Install Integration Services](http://technet.microsoft.com/en-us/library/ms143731(v=sql.120).aspx) ([http://technet.microsoft.com/en-us/library/ms143731\(v=sql.120\).aspx](http://technet.microsoft.com/en-us/library/ms143731(v=sql.120).aspx)).

SSIS 2014 Server Overview

The pre-SSIS 2012 Service, supported in SSIS 2012 and SSIS 2014 for backward compatibility, is different than the SSIS 2012 and SSIS 2014 server models. The SSIS 2012 and SSIS 2014 models store objects, settings, and operational data in a SQL Server database. It is accessed within SSMS by connecting to the instance of the SQL Server *Database Engine* hosting the Integration Services database.

Figure 15 is a screenshot in SSMS 2014 of the Integration Services Catalogs container:

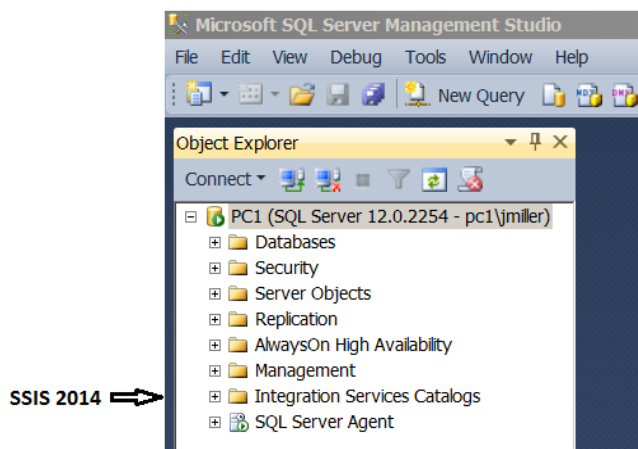


Figure 15: Integration Services Catalogs container

For more information on the SSIS 2014 server, see [Integration Services \(SSIS\) Server](http://technet.microsoft.com/en-us/library/gg471508(v=sql.120).aspx) (http://technet.microsoft.com/en-us/library/gg471508(v=sql.120).aspx).

The SSIS 2012 and SSIS 2014 servers leverage the project deployment model, which was introduced in SSIS 2012. This will be covered later in the “Convert to Project Deployment Model” section in this chapter.

In-Place Upgrade

To upgrade in place, first navigate to the SQL Server Installation Center screen, select Installation, and then select the *Upgrade from SQL Server 2005, SQL Server 2008, SQL Server 2008 R2, or SQL Server 2012* option. This selection will upgrade all of your existing SQL Server 2005/2008/2008 R2/2012 components and does not allow you to selectively upgrade SQL Server components.

Also note that you cannot use an in-place upgrade to perform the following actions:

- Reconfigure an existing installation of SSIS.
- Move from a 32-bit to a 64-bit version of SQL Server or from a 64-bit version to a 32-bit version.
- Move from one localized version of SQL Server to another localized version.

For more information, see [Upgrade Integration Services](http://technet.microsoft.com/en-us/library/cc879336(v=sql.120).aspx) (http://technet.microsoft.com/en-us/library/cc879336(v=sql.120).aspx).

The server configuration step is the only SSIS-specific option within the in-place upgrade process. It will prompt you for the account name and password for the SQL Server Integration Services service. This is the legacy SSIS service that existed in SQL Server 2005/2008/2008 R2. For more information on this service, see [Administration \(Integration Services\)](http://technet.microsoft.com/en-us/library/ms141799(SQL.105).aspx) (http://technet.microsoft.com/en-us/library/ms141799(SQL.105).aspx).

Side-by-Side Upgrade

A side-by-side upgrade requires you to select the SQL Server 2014 features shown earlier in Figure 14. You will need to provide a new instance name. Note that an instance name is optional for new installations.

Other areas that you need to consider with a side-by-side upgrade involve:

- Designing SSIS packages. You will need the version-specific instance of SQL Server Data Tools (SSDT) or Business Intelligence Development Studio (BIDS) to work with packages stored in earlier versions of SSIS.
- Managing SSIS packages. You will not be able to mix and match SSDT and BIDS with different versions of the SQL Server database. This means that the SQL Server package formats (i.e., 2005, 2008, 2012, and 2014) must be stored in the SQL Server msdb database of the same version, and must be accessed by the same version of SSMS.
- Running packages. The SQL Server 2014 version of dtexec will convert SSIS 2005/2008/2008 R2/2012 packages to the SQL Server 2014 package version prior to executing.

For more information on side-by-side installations, see [Interoperability and Coexistence \(Integration Services\)](http://technet.microsoft.com/en-us/library/bb522577(v=sql.120).aspx) ([http://technet.microsoft.com/en-us/library/bb522577\(v=sql.120\).aspx](http://technet.microsoft.com/en-us/library/bb522577(v=sql.120).aspx)).

It is natural for existing SSIS shops to be risk-averse and delay the upgrading of packages to the most recent version. However, given the strong tools and wizards in SSIS 2014, it is highly recommended that SSIS packages be upgraded to the SSIS 2014 format, especially for SSIS 2008 and SSIS 2008 R2. The upgrade wizard is designed to handle version differences.

SQL Server 2005 may be more difficult given the Script task migration from Visual Studio for Applications (VSA) to Visual Studio Tools for Applications (VSTA) and the changes in the Lookup transformations. However, SSIS shops do not want to be in the position where their packages are in a discontinued format, which is the case for DTS in SQL Server 2012 and SQL Server 2014. For more information on converting to VSTA, see [Migrate Scripts to VSTA](http://technet.microsoft.com/en-us/library/bb522527(v=sql.120).aspx) ([http://technet.microsoft.com/en-us/library/bb522527\(v=sql.120\).aspx](http://technet.microsoft.com/en-us/library/bb522527(v=sql.120).aspx)).

New Installations

New installations require you to select the SQL Server 2014 features previously shown in Figure 14. You will also want to upgrade existing SSIS packages to SSIS 2014. As part of this process, make sure that all of your package configurations are moved to this new instance of SQL Server, and that these package configurations reference the correct files, directories and database connections.

In summary, upgrading to SSIS 2014 requires the installer to choose the SSIS related components covered in this section. Side-by-side installations result in the most complexity on the server but allow you to keep your SSIS packages in the downstream format. Upgrading in place and new installations reduce the complexity on the server but require a packages upgrade.

After the install completes, the next step is to upgrade your SSIS packages.

Project Conversion Wizard

The SSIS 2014 Project Conversion Wizard upgrades your SSIS packages from previous formats (i.e., SSIS 2005/2008/2008 R2/2012) to SSIS 2014. The following example will show how to convert packages stored in a Visual Studio project accessed from a file directory. Converting packages managed by the legacy SSIS service will be covered in a later section.

Figure 16 below shows all the SSIS-related files for our upgrade sample.

Documents library
AWDW_SQL2008R2

Name ^	Date modified	Type	Size
bin	10/1/2014 9:56 AM	File folder	
AWDW	2/9/2012 3:01 PM	Analysis Services Database	2 KB
AWDW	10/1/2014 10:45 AM	Integration Services project file	3 KB
AWDW.dtproj	10/1/2014 10:59 AM	Visual Studio Project User Options file	1 KB
AWDW	12/3/2010 2:01 PM	Microsoft Visual Studio Solution	1 KB
EP_Load_DimGeography	10/1/2014 10:32 AM	Integration Services Package	414 KB
EP_Load_DimSalesTerritory	10/1/2014 10:37 AM	Integration Services Package	348 KB
EP_Load_Link_State_SalesTerritory	10/1/2014 10:31 AM	Integration Services Package	349 KB
EP_Load_Sales_Territory	10/1/2014 10:30 AM	Integration Services Package	354 KB
MP_SQL2012	10/1/2014 10:49 AM	Integration Services Package	94 KB

Figure 16: SSIS 2008 R2 sample files

The first step is to open a version of Visual Studio that supports the SQL Server 2014 BI templates. The following example is based on the Visual Studio 2013 Shell.

Note: Unlike earlier versions of BIDS and SSDT, SQL Server 2014 now requires a separate download for the Visual Studio Shell and BI templates. For more information see [Microsoft SQL Server Data Tools](http://msdn.microsoft.com/en-us/hh297027.aspx) (<http://msdn.microsoft.com/en-us/hh297027.aspx>).

After installing Visual Studio 2013, SSDT can be found under either:

- The Visual Studio 2013 program group:
(Start, All Programs, Visual Studio 2013)

- The SQL Server 2014 program group
(Start, All Programs, Microsoft SQL Server 2014, SQL Server Data Tools for Visual Studio 2013)

Once inside the shell, you see the Open Project link, as shown in Figure 17.

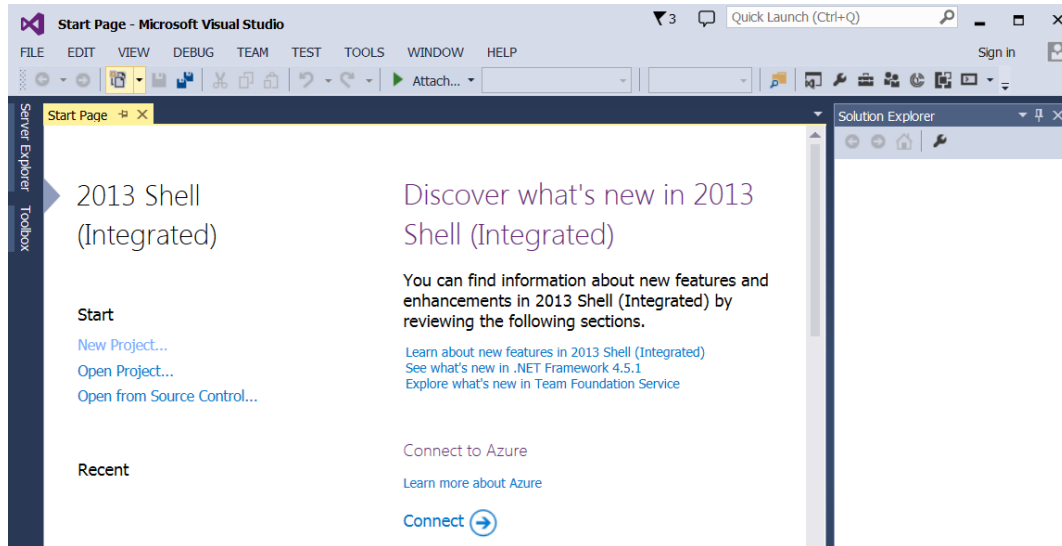


Figure 17: Opening a project in the Visual Studio 2013 Shell

The next step is to select the SSIS project, as shown in Figure 18.

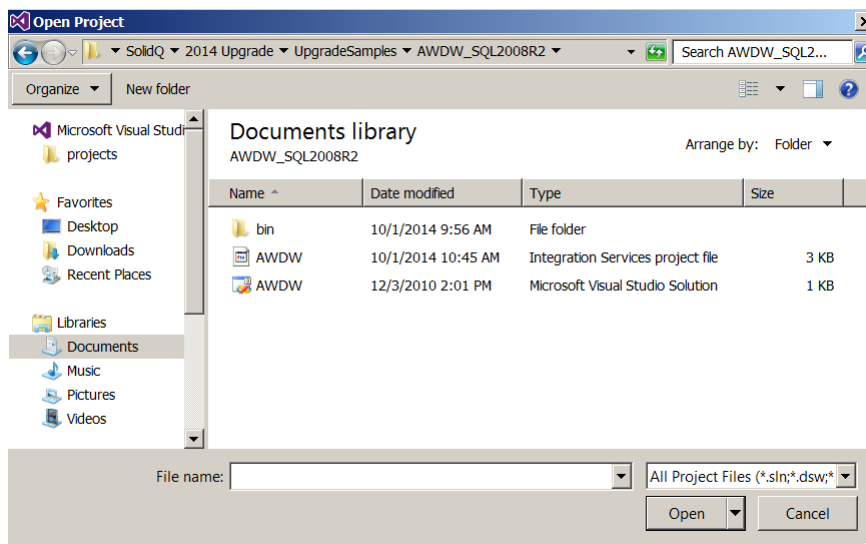


Figure 18: Selecting the SSIS project

The first page, shown in Figure 19, is the Review Project And Solution Changes page. It indicates the project files that will require changes and warns that this is a *one-way* change.

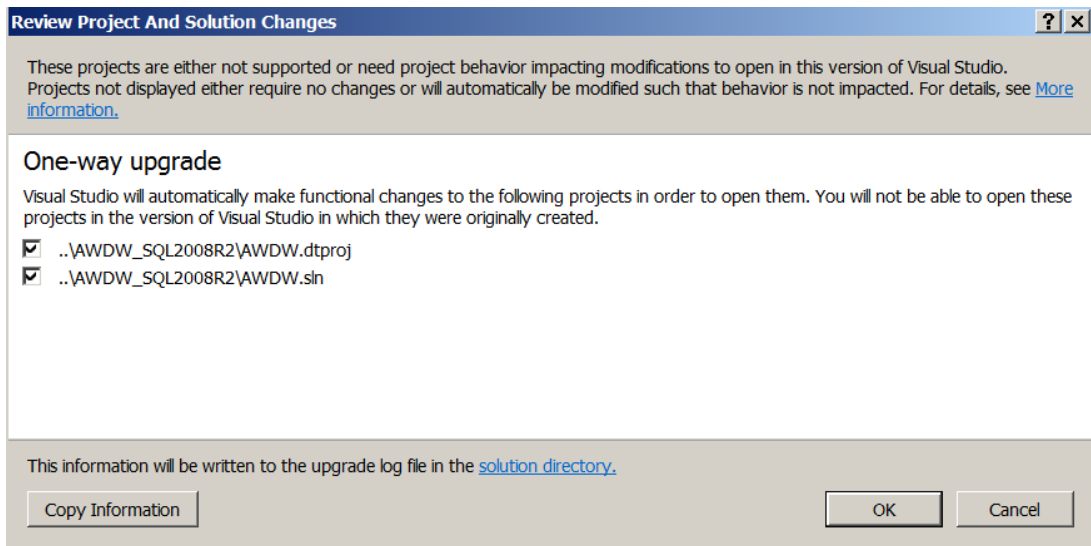


Figure 19: Viewing the Review Project And Solution Changes page

Click the OK button to start the conversion. The wizard then makes a backup of the original files, performs the upgrade, and displays the report shown in Figure 20.

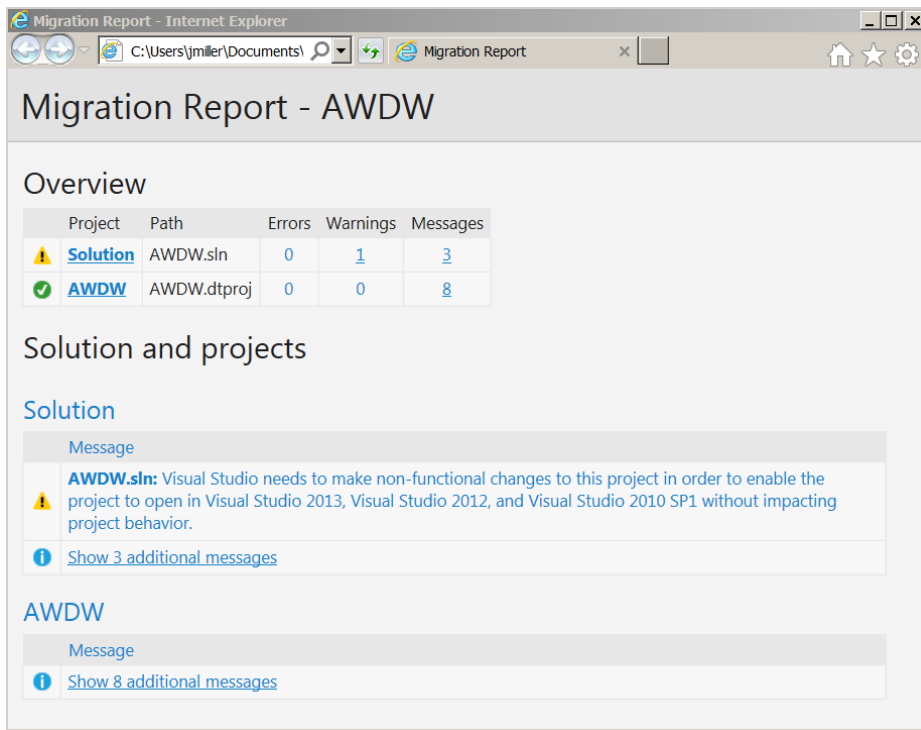


Figure 20: SSIS project migration report

At the same time, a second wizard is launched for upgrading the individual packages, as shown in Figure 21.

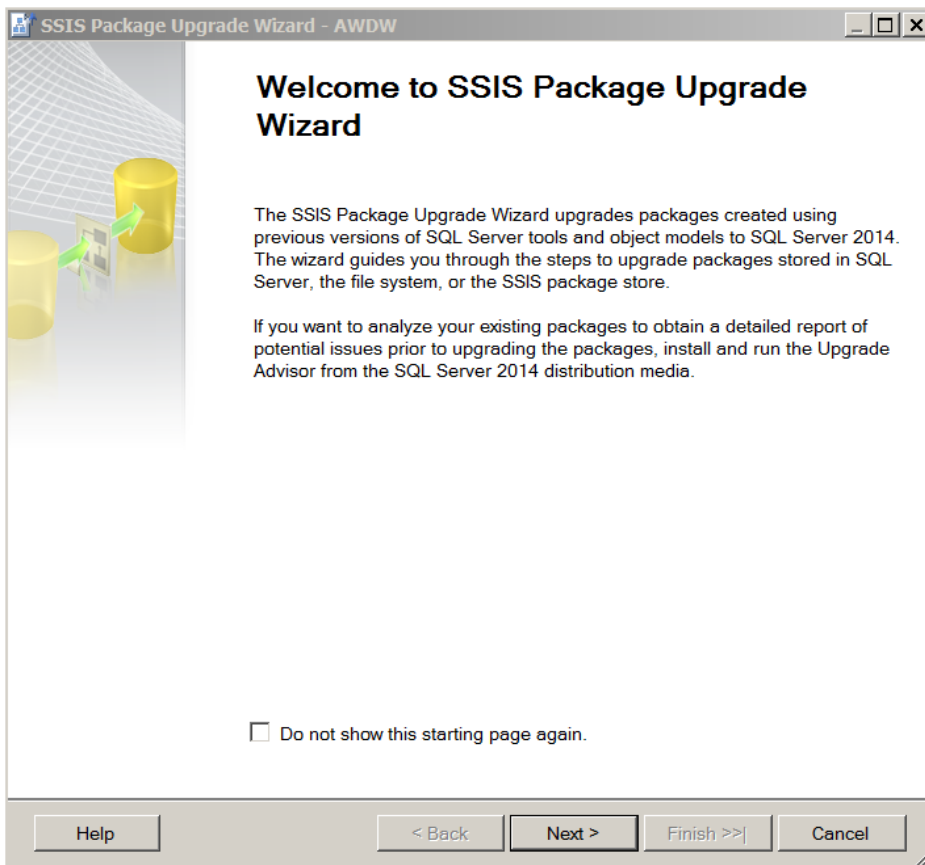


Figure 21: SSIS Package Upgrade Wizard

Clicking Next brings you to the Select Packages page, which Figure 22 shows.

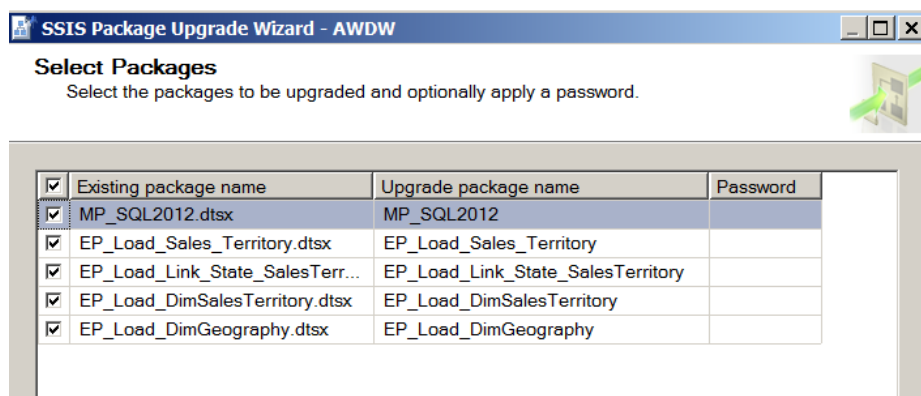


Figure 22: Select Packages page

Choose the packages that you would like to upgrade, and supply any package level passwords if needed. Note that you can also change package name(s) at this stage if desired. Click Next to go to the Management Options page shown in Figure 23.

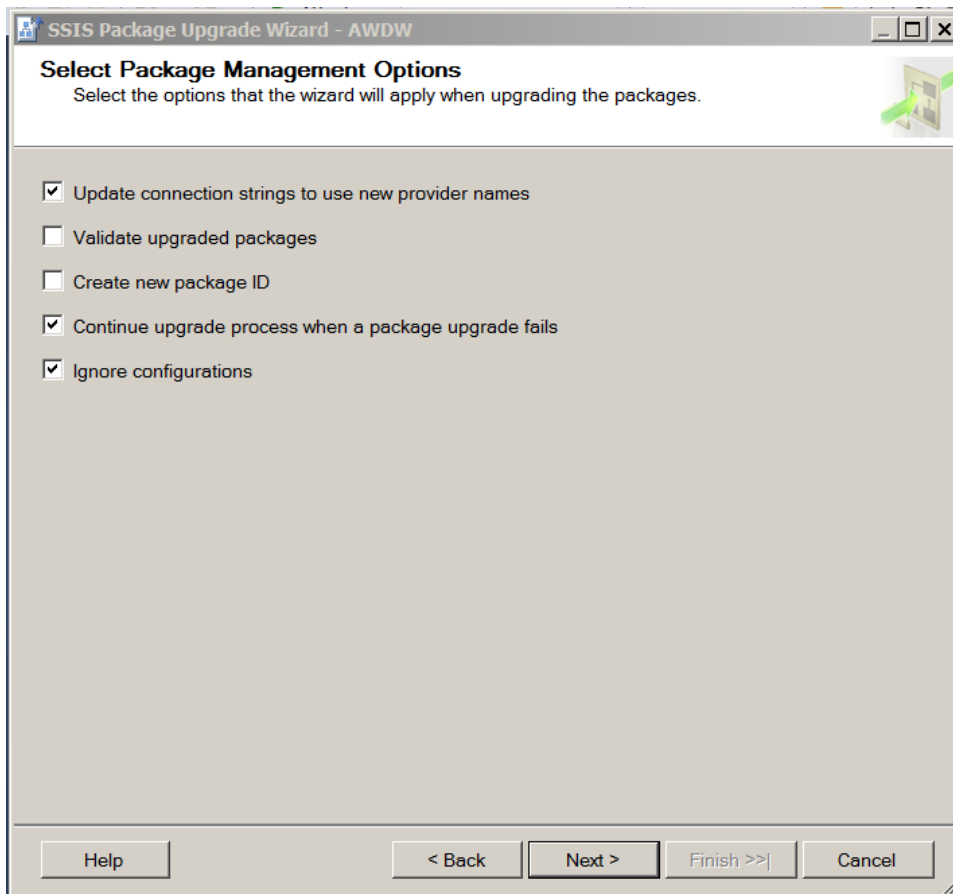


Figure 23: Select Package Management Options page

In the Select Package Management Options page, you set the package upgrade options. Your options include:

- Updating connection strings so that they use new provider names (default)
- Validating the upgraded packages
- Creating new package IDs
- Continuing the upgrade process if a package upgrade fails (default)
- Ignoring configurations (default)

Once these options have been confirmed, click Next. The summary page, shown in Figure 24, is then displayed. This is a good time to make sure that everything is ready to be converted, especially if the SSIS project and packages are under source control. Click the Finish button to start the conversion.

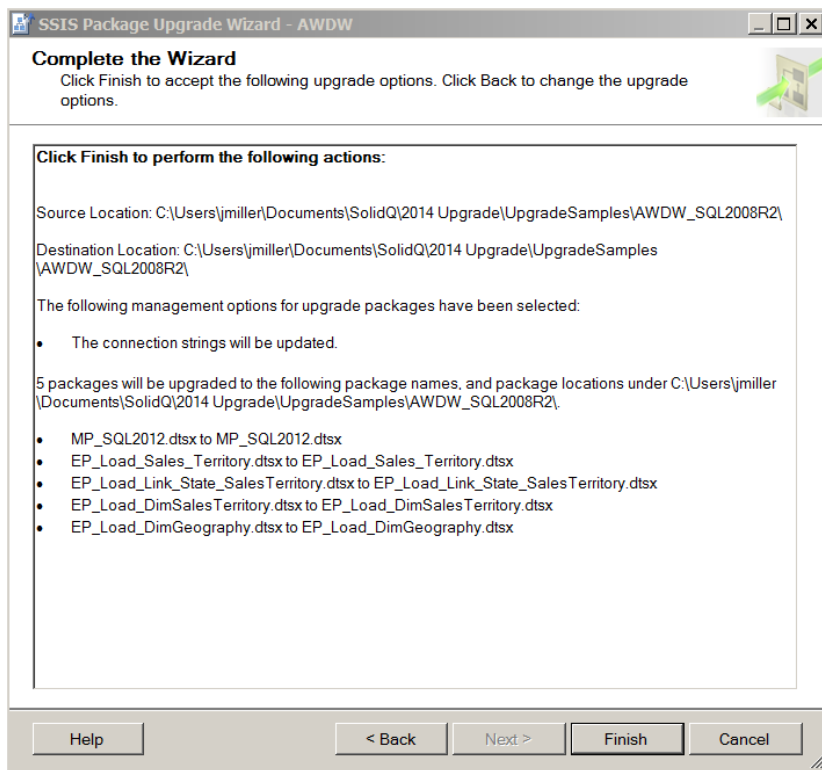


Figure 24: Reviewing the SSIS upgrade actions to be performed

Once complete, the wizard's results are displayed, as Figure 25 shows.

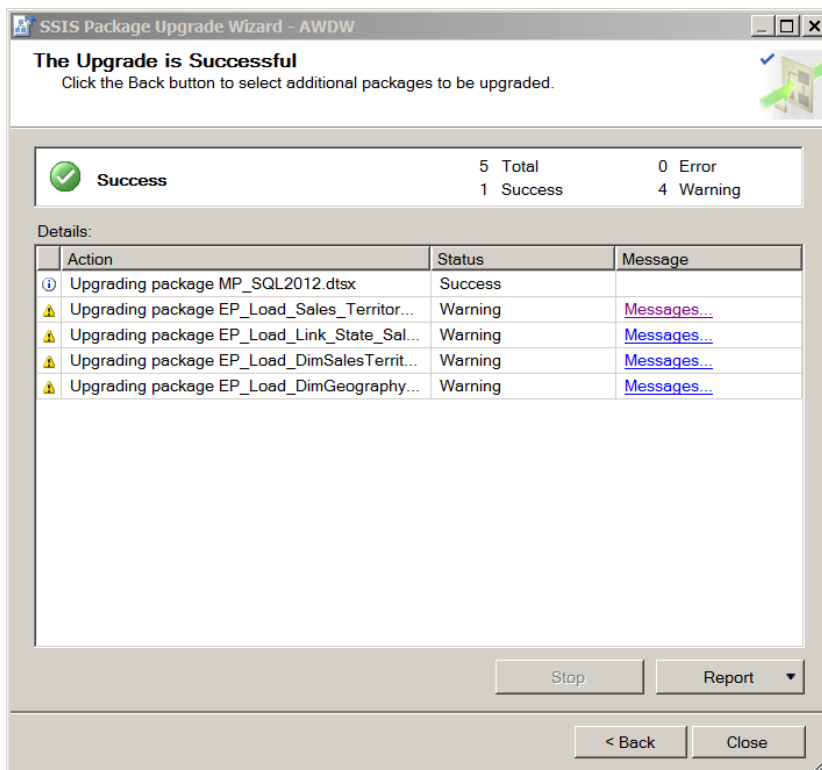


Figure 25: SSIS Package Upgrade Wizard results

Note that you are provided with a link for upgraded packages providing informational, warning, and error message that were captured during the upgrade. Figure 26 shows sample messages.

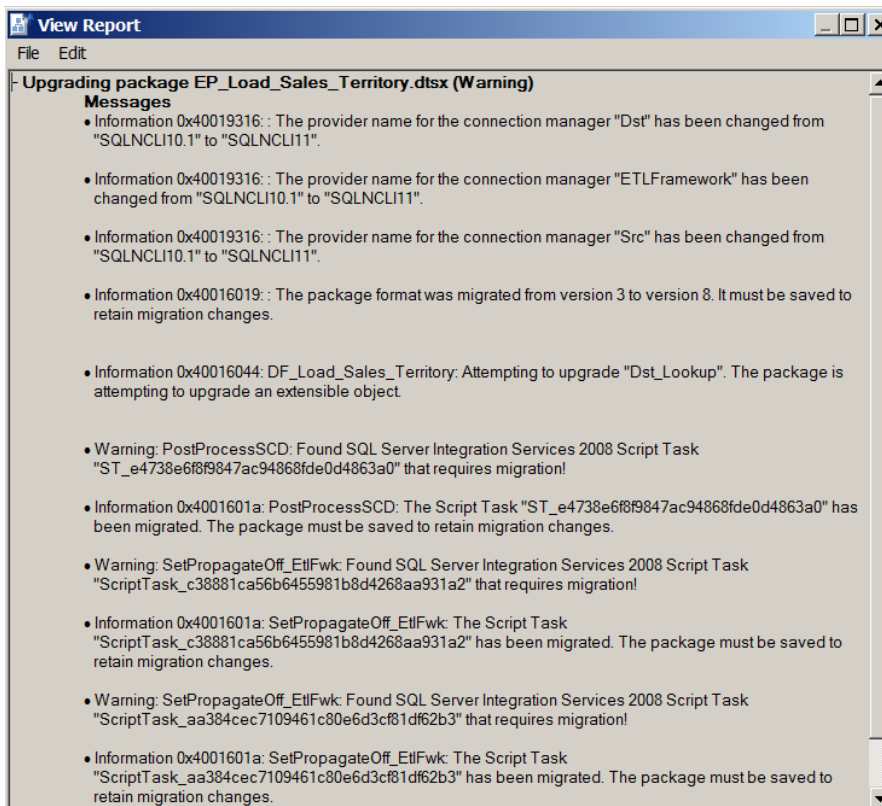


Figure 26: SSIS Package Upgrade Wizard messages

For this upgrade, messages were generated for the following:

- **Changed connection provider names.** Selecting the *Update connection strings to use new provider names* option instructed the SSIS Package Upgrade Wizard to convert all database connections to the SQL Server 2014 provider (i.e. SQLNCLI11). Note that this step does not modify the connection strings that are stored in SSIS configuration files
- **DTSX package format change.** The wizard automatically upgraded the package to SSIS 2014's new package format. This new format makes it easier: to read the XML package file, to work with source control and diff/merge software and to share transforms between data flows.
- **Script migration.** The wizard automatically upgraded the Script tasks. Note that this is displayed as a warning but should not be viewed as one since the scripts will continue to work after the upgrade without user modification.

Figure 27 shows the file directory after the SSIS Package Upgrade Wizard completes. Note that the original solution, project, and packages have been copied into the Backup directory.

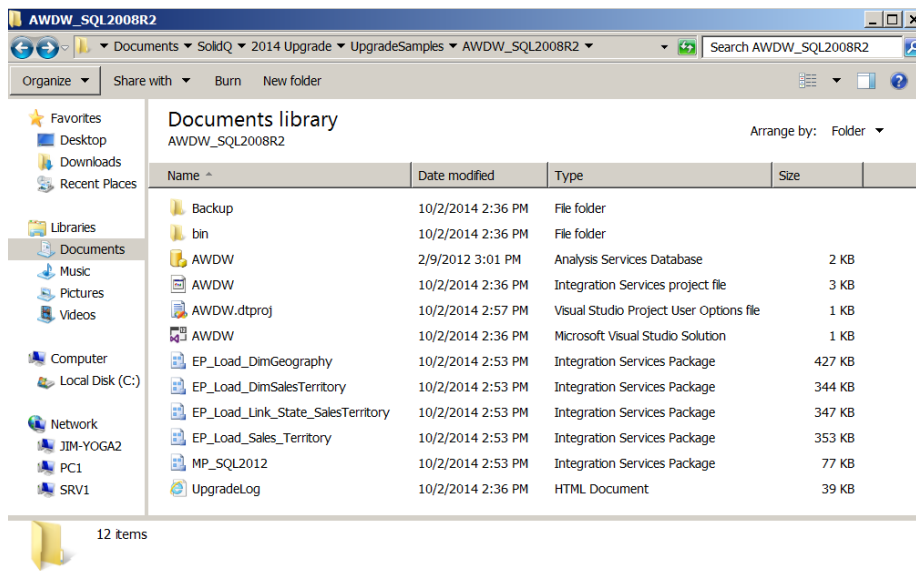


Figure 27: Viewing the file directory after the upgrade

This next example demonstrates how you can upgrade packages stored on the server. In this example, the sample packages are stored within the MSDB database in the AWDW directory. Figure 28 shows how you can highlight the directory and select the Upgrade Packages option from its context menu.

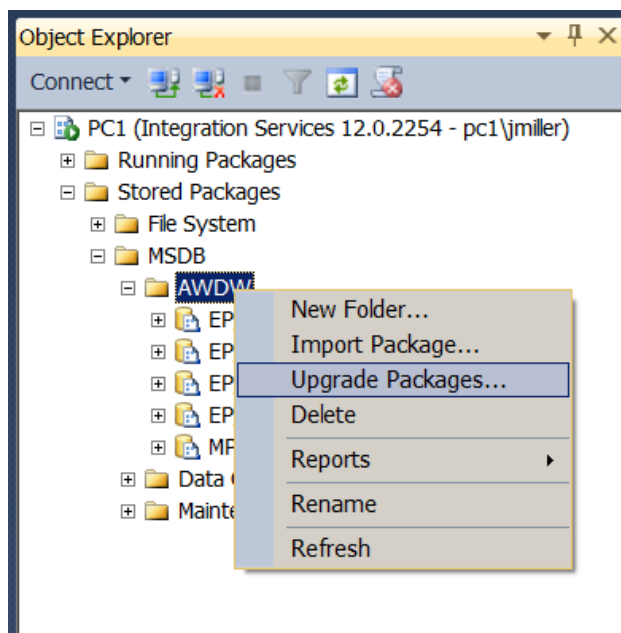


Figure 28: Upgrading packages stored on the server

After selecting Upgrade Packages, the welcome screen is displayed, as shown in Figure 29.

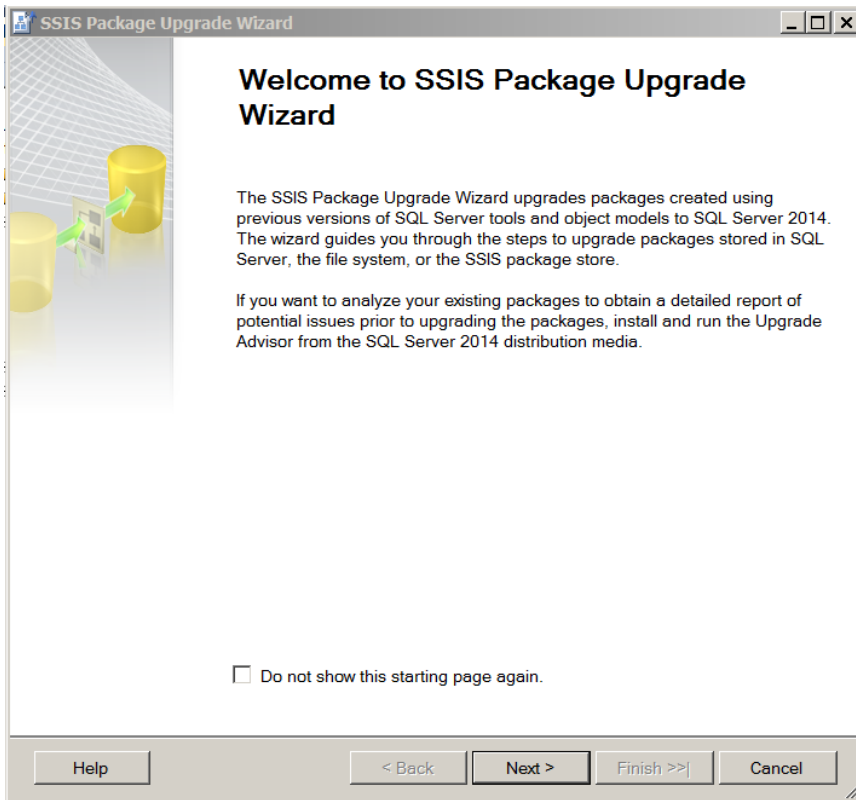


Figure 29: Launching the SSIS Package Upgrade Wizard from SSMS

Clicking the Next button brings up the Select Source Location screen shown in Figure 30.

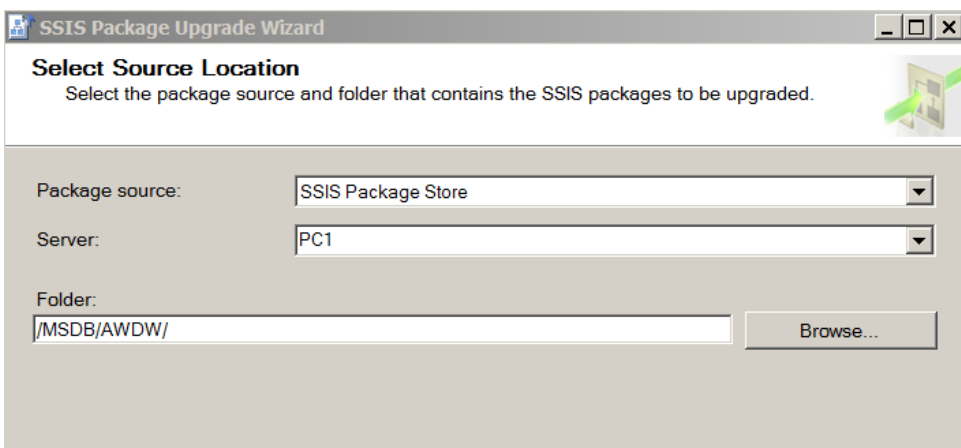


Figure 30: Select Source Location page

Select the AWDW directory and then click Next. At this point, the upgrade process is the same as the process documented earlier in this chapter.

Now that the packages have been upgraded, the next step is to test the converted packages within SSIS 2014. Figure 31 is a modified screenshot of the upgraded master package.

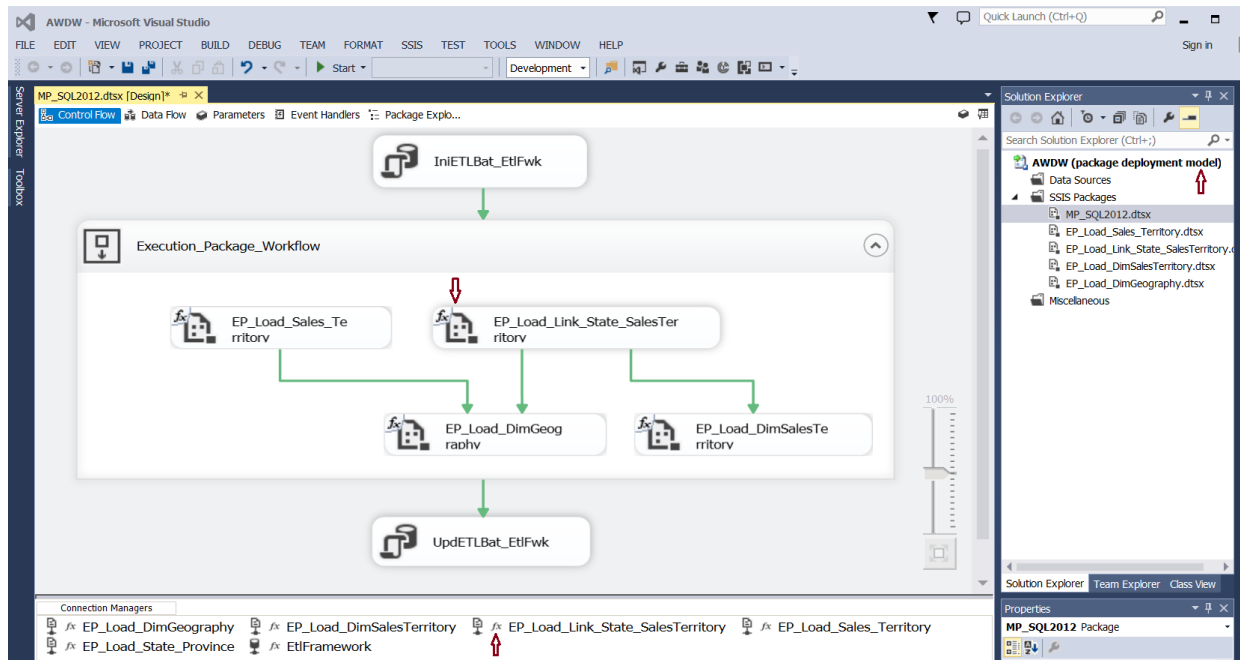


Figure 31: Upgraded master package

One useful feature for SSIS 2012 and SSIS 2014 is that an “Expression Adorner” exists for every task and connection that leverages expressions. All connections and Execute Package tasks within the sample master package have expressions. The arrows in Figure 31 point to the expression indicator for one task and connection. This did not exist in previous versions and is a useful addition for SSIS developers.

Notice that the package ran successfully without any post-upgrade changes. Also notice how the solution is being tagged as a package deployment model. As stated previously, it is worth looking into the project deployment model’s capabilities and running the Integration Services Project Conversion Wizard to convert from the package deployment model to the project deployment model. We will cover this topic shortly.

Finally, don’t forget about your package configurations. Make sure that:

- The XML configuration file is in the correct location.
- Environment variables are defined.
- Configurations have valid connection strings/values.

The last screenshot in this section demonstrates SSIS 2014's enhanced usability for developers. Figure 32 is a screenshot of the data flow for one of the converted execution packages. Note that subsequent to the upgrade, the data flow has been modified, disconnecting the OLE DB source (Src) from other transforms.

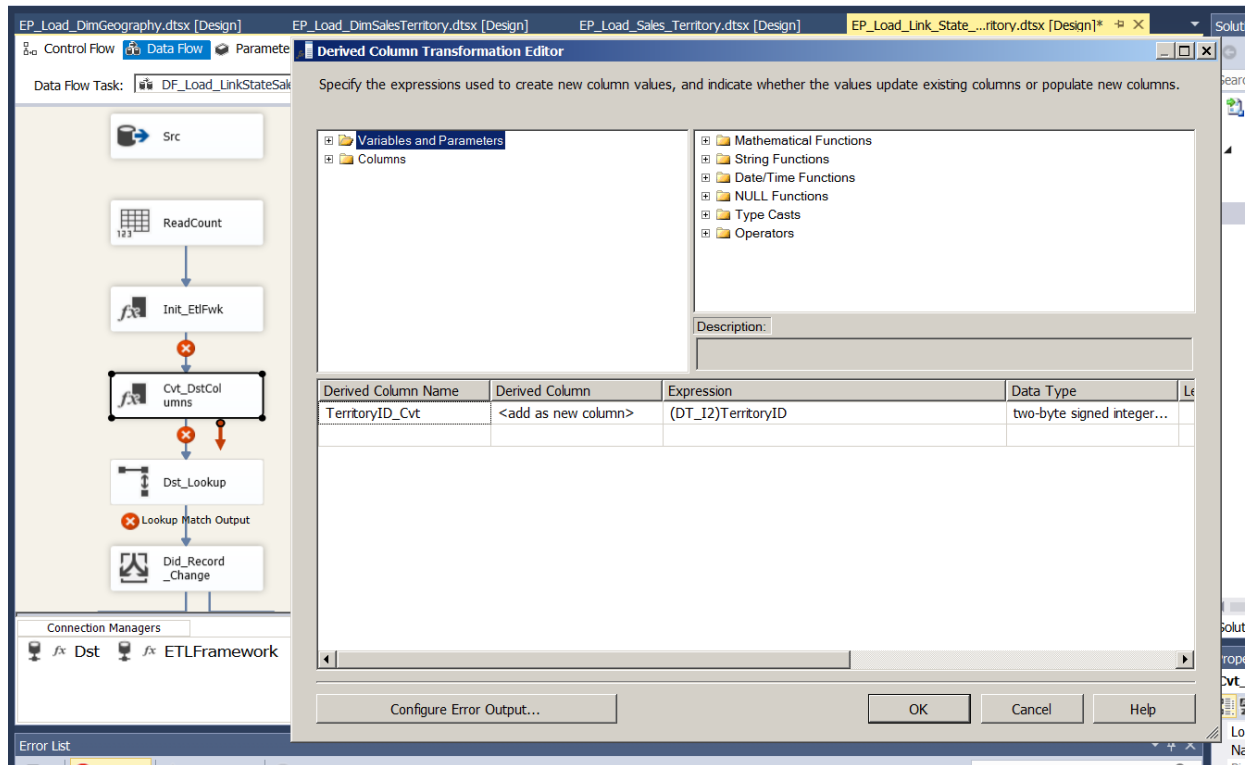


Figure 32: Modified data flow for one of the converted execution packages

Figure 32 shows how you can work with a transform (in this case, Derived Column) even though it is not connected to an upstream transform. This may seem like a minor point, but it is a great example of a usability feature in SSIS 2012 and SSIS 2014. Previous versions would not let you open a transform that did not have upstream transform input feed.

In summary, the SSIS Package Upgrade Wizard walks you through a series of input screens before upgrading your packages to SSIS 2014. This wizard is thorough and the majority of packages will run post upgrade without changes to the code. Note that package configuration values will need to be changed.

This is covered, along with other best practices, in the MSDN article ["5 Tips for a Smooth SSIS Upgrade to SQL Server 2012"](http://msdn.microsoft.com/en-us/library/hh667275.aspx) (<http://msdn.microsoft.com/en-us/library/hh667275.aspx>), which is equally applicable to SSIS 2014.

Now that the packages have been upgraded and successfully tested, you can look to convert to the project deployment model for SSIS 2012 and SSIS 2014.

Convert to the Project Deployment Model

The SSIS 2012 and SSIS 2014 project deployment model provides many features that can help you reduce your ETL total cost of ownership (TCO). You use the Integration Services Project Conversion Wizard to convert a project to the project deployment model. Start by opening the project that was just converted to SSIS 2014, as shown in Figure 33.

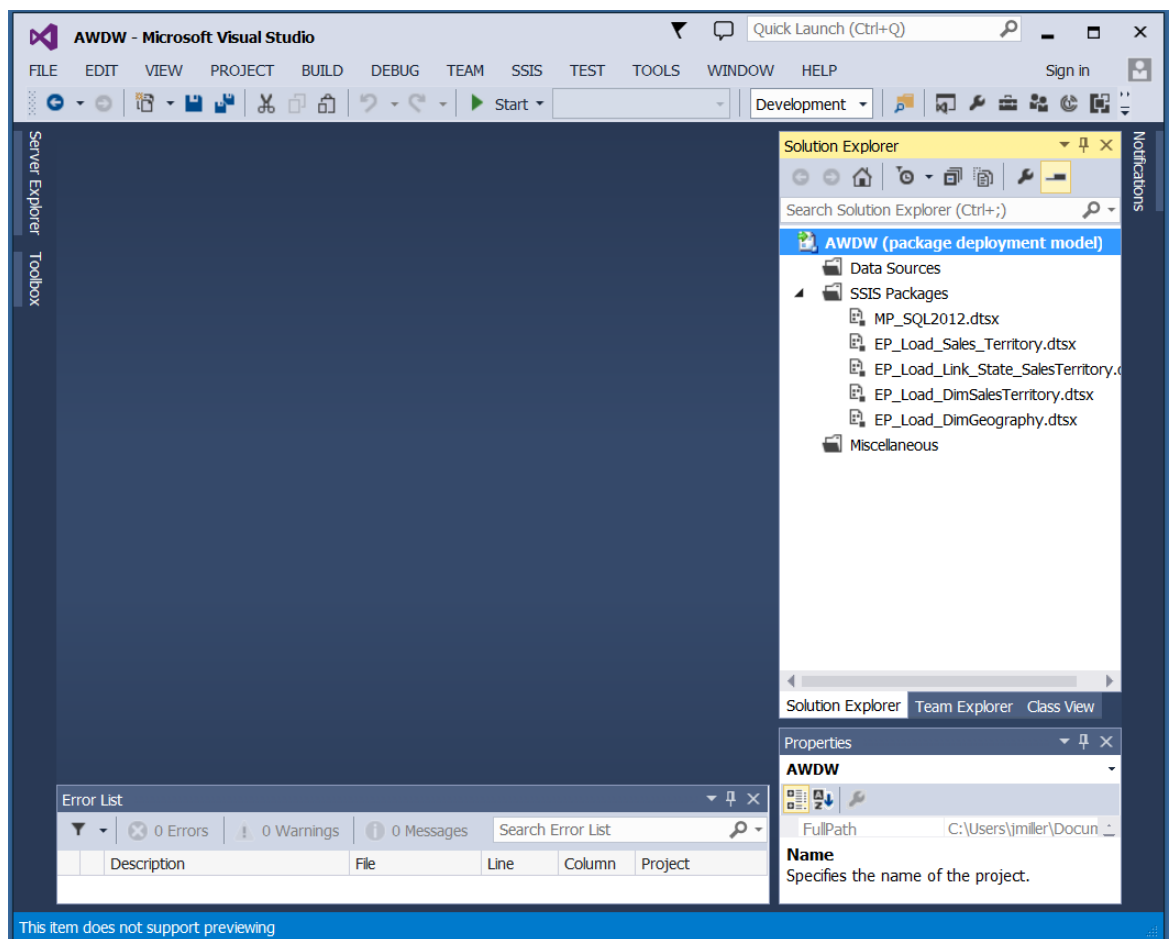


Figure 33: SSIS project using the package deployment model

A Project Deployment Model conversion wizard is accessible from the project's context menu, as shown in Figure 34. You get to this context menu by highlighting the project and then pressing the right mouse button.

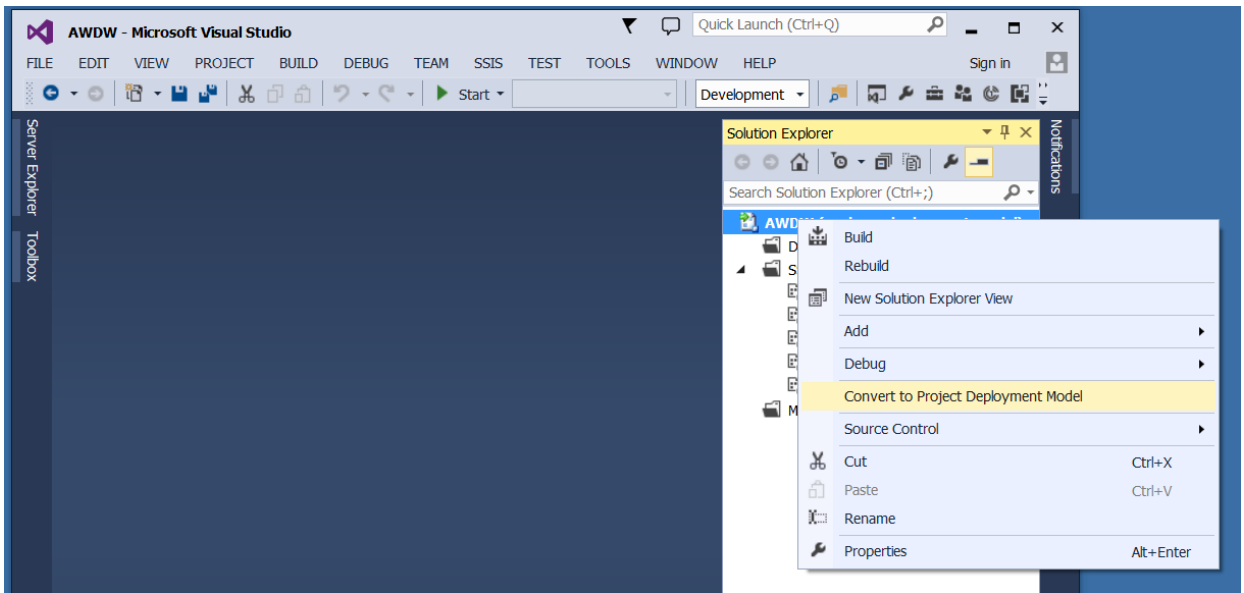


Figure 34: Starting the project conversion wizard

Selecting this option starts the wizard. Its Introduction screen is shown in Figure 35.

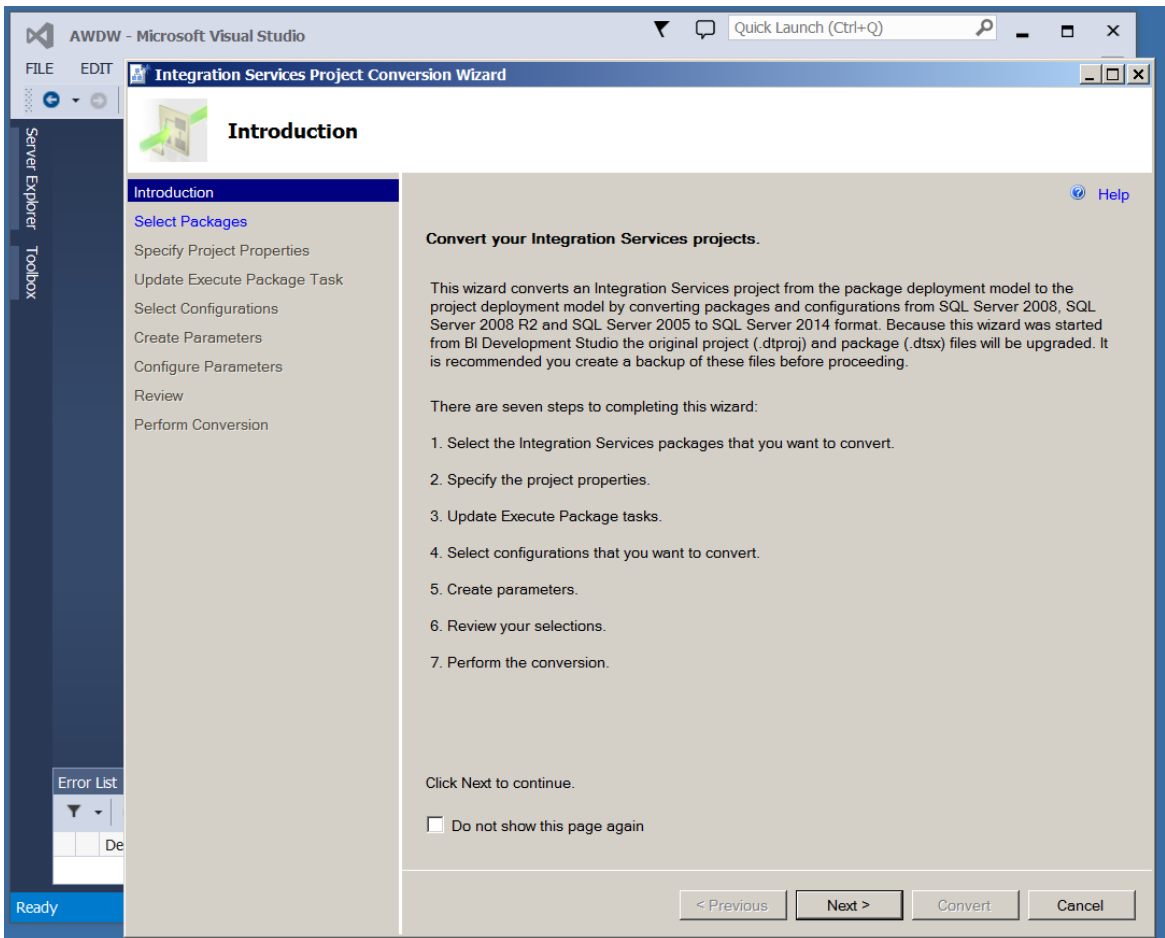


Figure 35: Reviewing the conversion steps in the Introduction page

Clicking Next moves the wizard to the Select Packages page, shown in Figure 36.

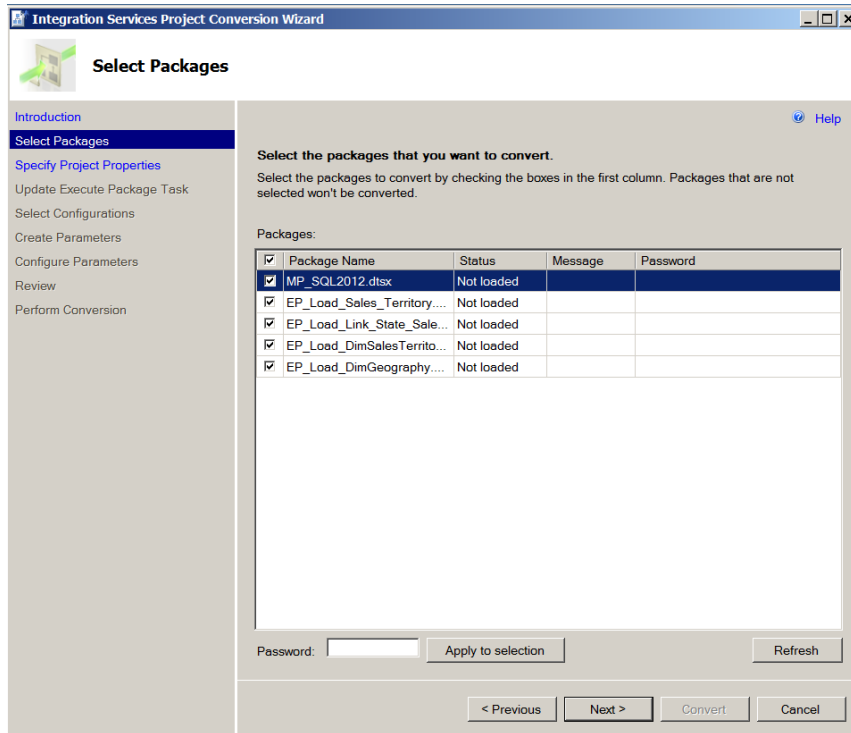


Figure 36: Selecting the packages to convert

Select the packages targeted for conversion, and click the Next button. Specify any necessary project properties and click Next again to navigate to the Update Execute Package Task page shown in Figure 37.

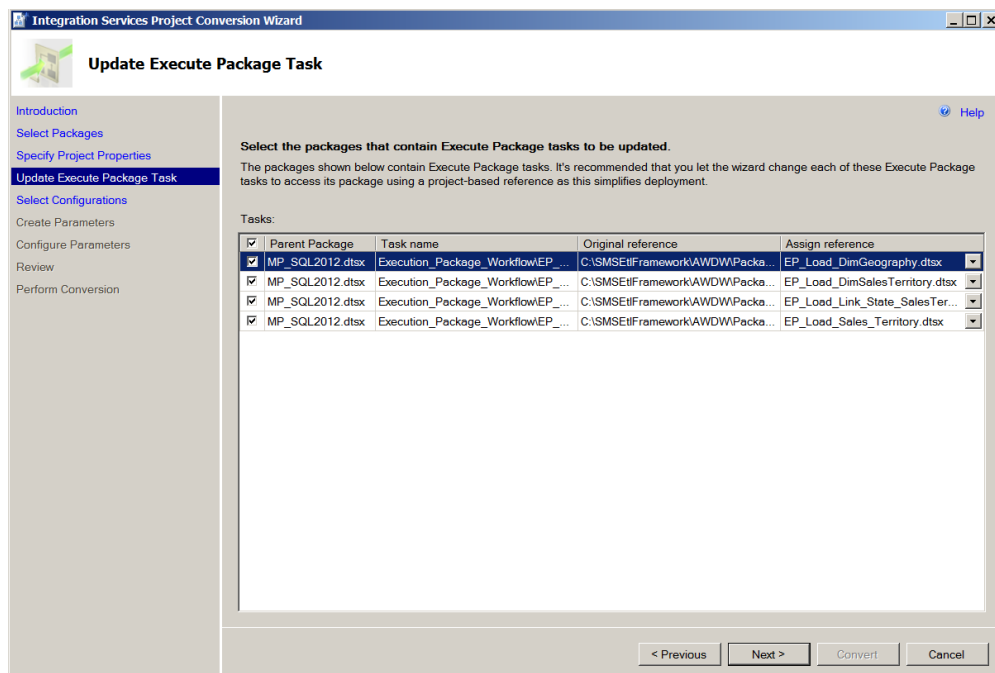


Figure 37: Selecting the Execute Package tasks to update

The wizard has identified the master package and all execution package references. Moving to the new project reference model ensures that the project is self-contained. Some benefits are that missing packages will not occur during deployment and the need for separate file connections for each Execute package task is eliminated. Selecting the Next button brings up the Select Configurations screen shown in Figure 38.

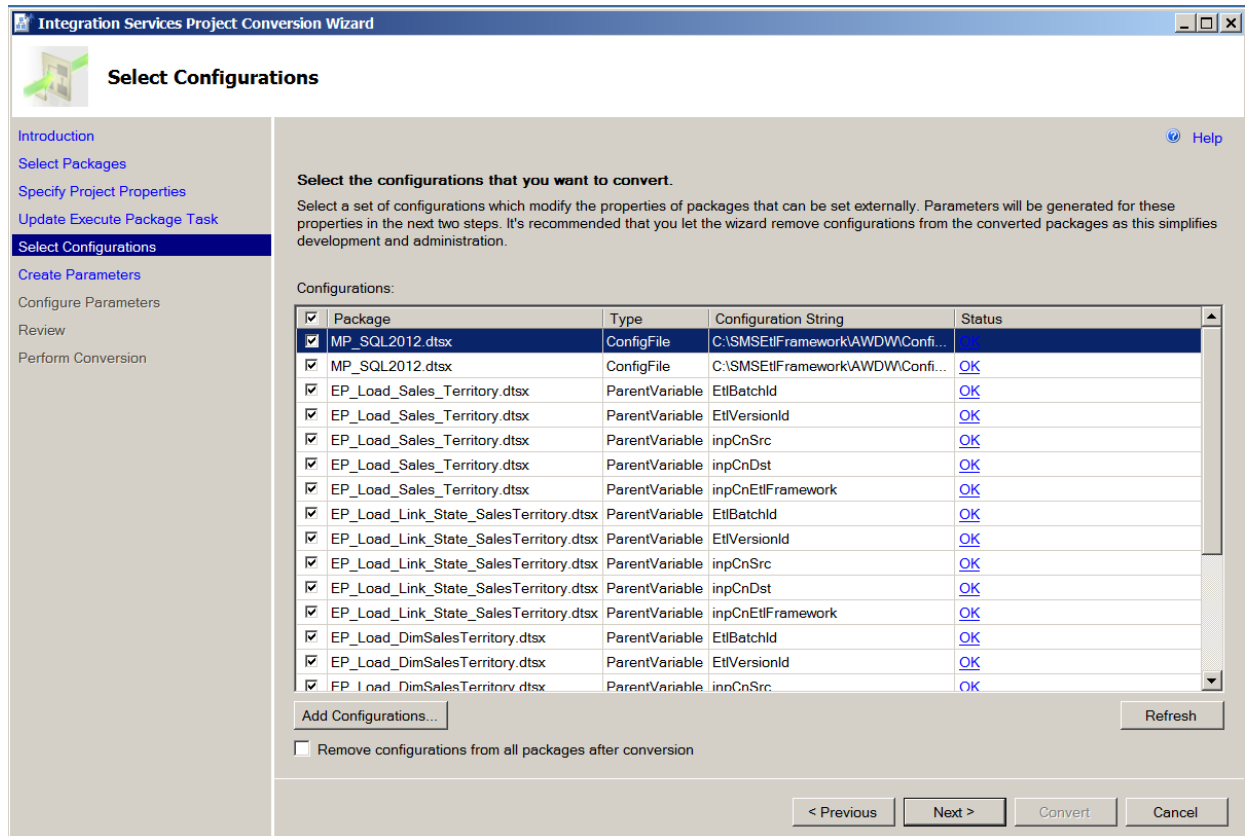


Figure 38: Selecting the configurations to convert

The wizard identifies all configuration parameters and their type, and ensures that the external references exist. In this example, these are the environment variable and XML configuration file. You have the option of adding and deleting configurations from the list. You also have the option of removing the configurations from the packages. Select that option since you are changing over to the SSIS 2014 project model, which removes the need for the environment variable and XML file configurations.

Keep all of the configurations that were identified and click the Next button to navigate to the Create Parameters page shown in Figure 39.

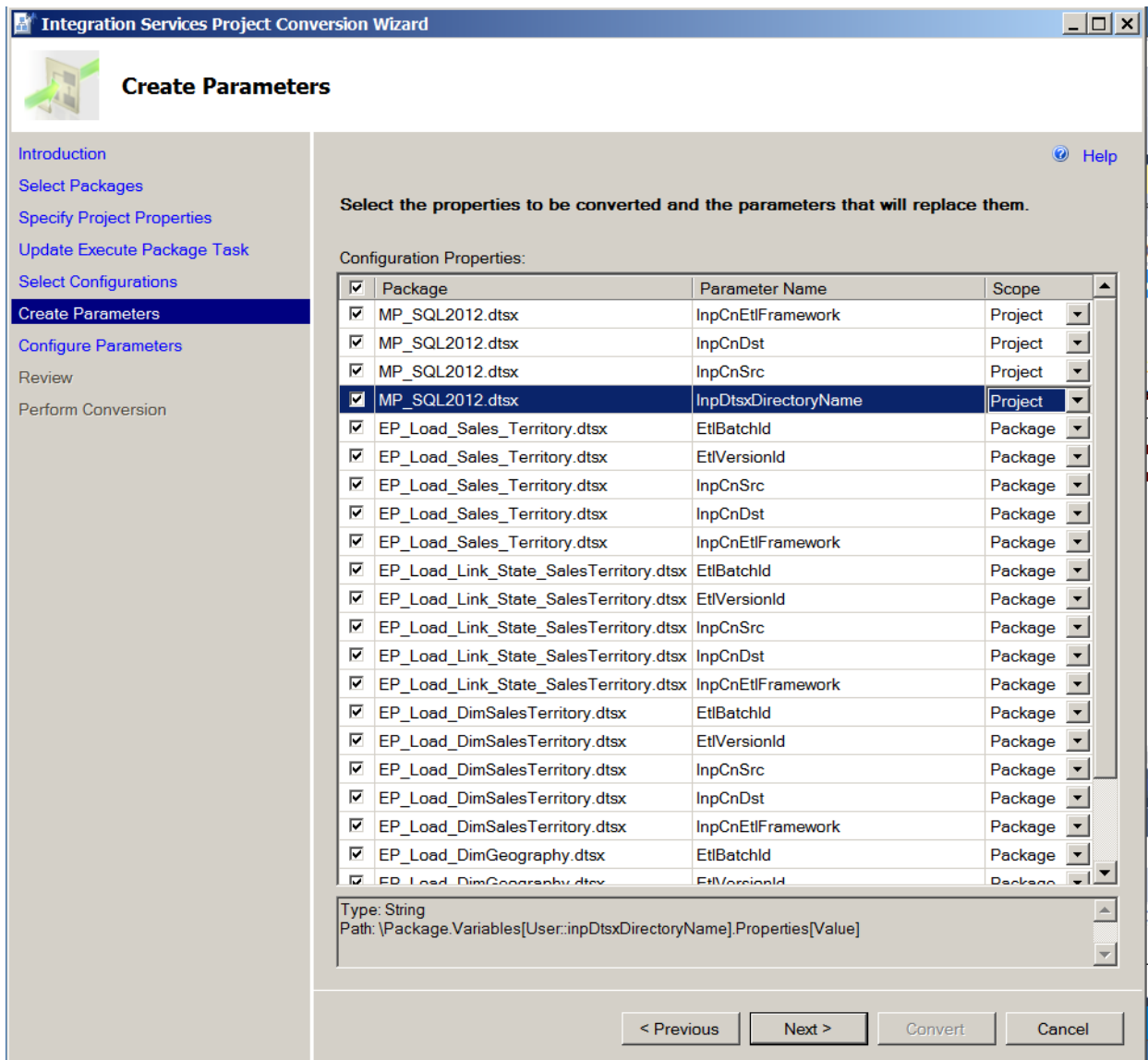


Figure 39: Creating the parameters

The next step is to change the scope of all master package configurations from Package to Project level. Note that all of the execution packages “Parent package” configurations are converted to package-level parameters. The administrator will be able to use SSMS or T-SQL to easily configure configuration values once the project and packages are deployed to the SSIS catalog. You will see how these are configured when you review the Execute Package Task Editor page later in this section.

Click the Next button to navigate to the Configure Parameters screen shown in Figure 40.

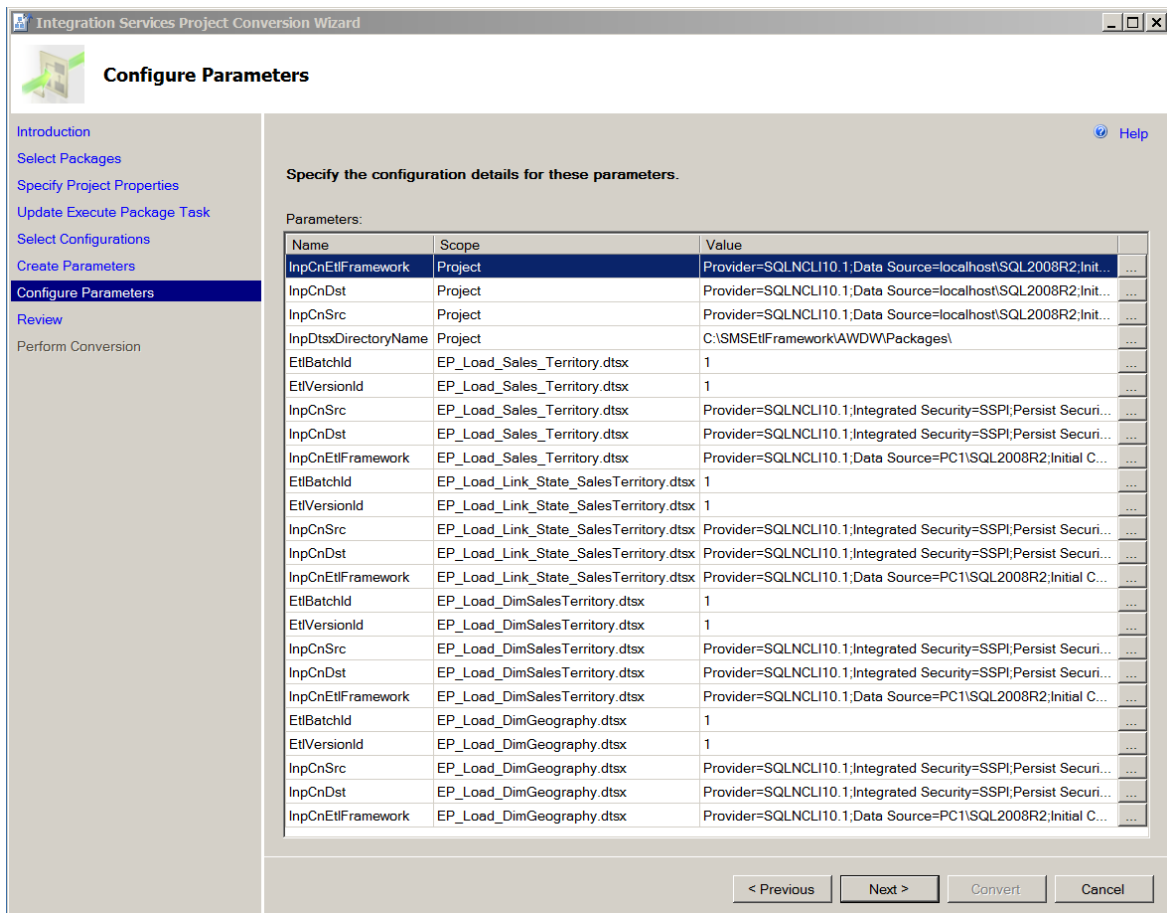


Figure 40: Configuring the parameters

On the Configure Parameters page, you can change the settings for each configuration value. Clicking the ellipsis (...) button will result in the Set Parameters pop-up window shown in Figure 41.

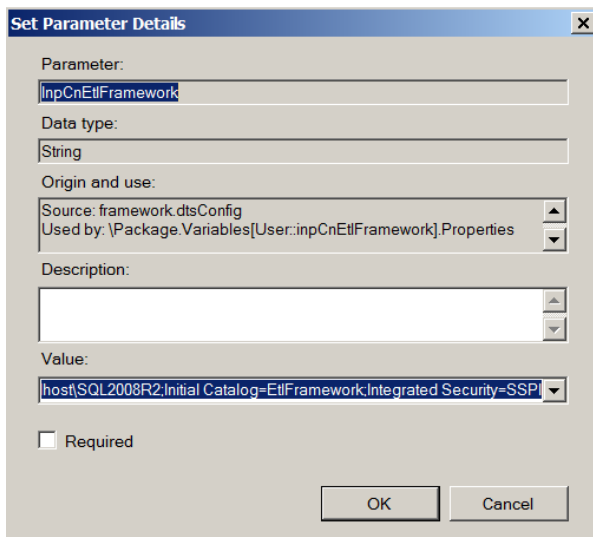


Figure 41: Setting the parameter details

The parameter value and description can be modified. You can also select the Required check box to make it a required attribute. Press the Cancel button to leave these values as is for now. Click the Next button to go to the Review screen shown in Figure 42.

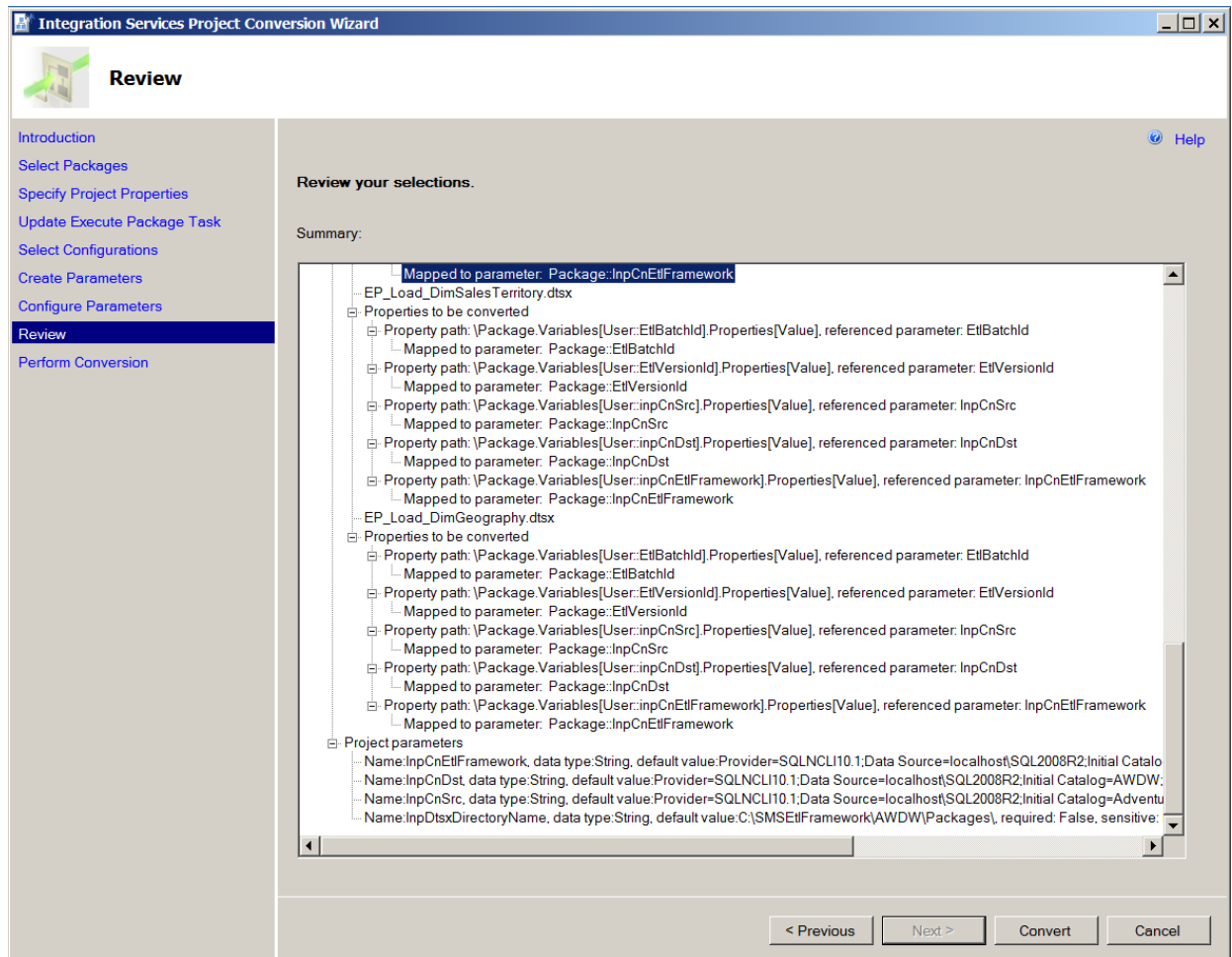


Figure 42: Reviewing the selections

Confirm the selections made in prior screens before clicking the Convert button, which starts the upgrade process. The wizard will display the Results screen shown in Figure 43, which provides the conversion results as each package is converted. When complete, it displays a pop-up window that reminds you that the project will need to be saved for the upgrade changes to be applied.

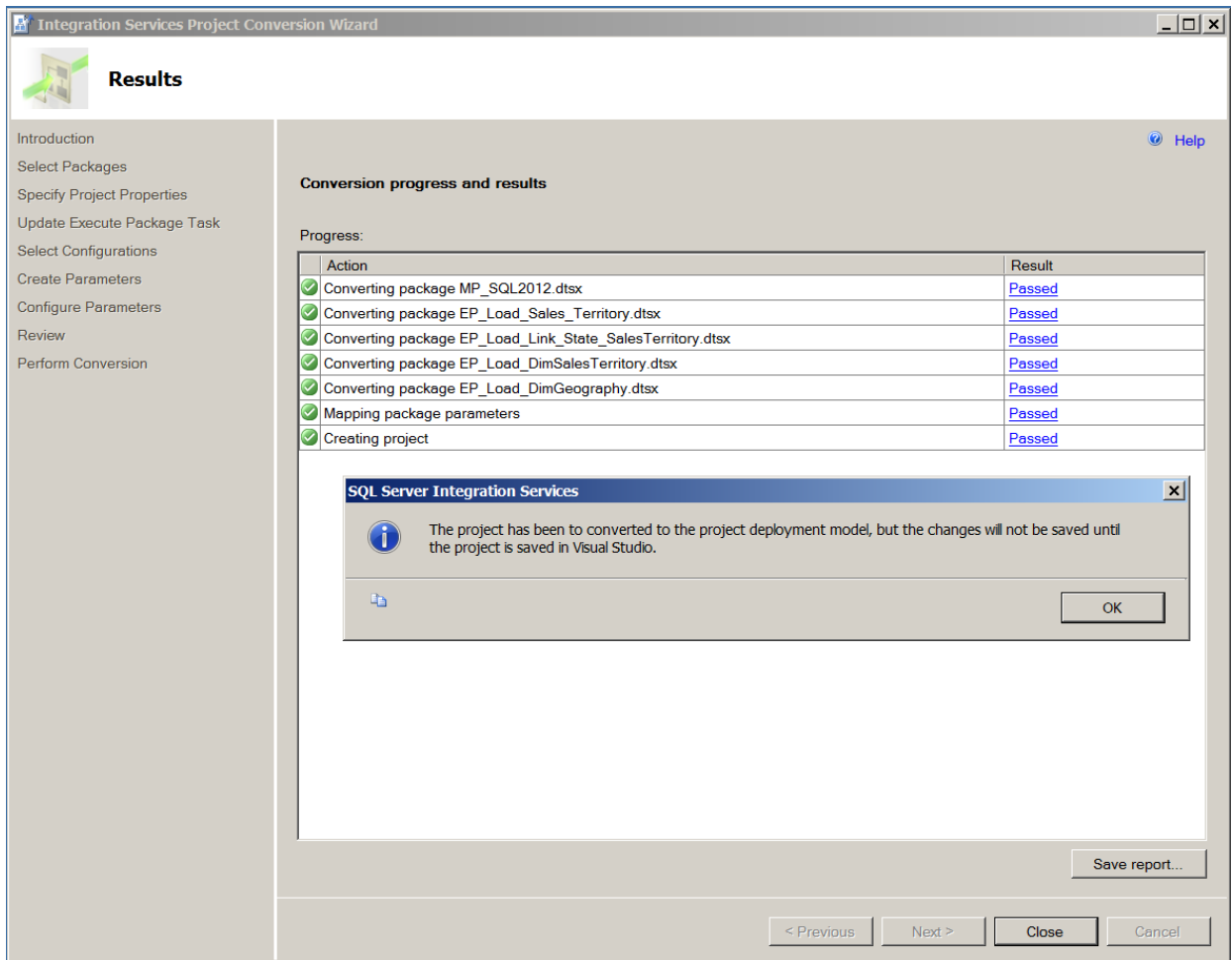


Figure 43: Reviewing the results

Click OK to remove the pop-up window and then click Close to complete the wizard. Now that the wizard has completed, let's view what is different for the project deployment model, starting with the project parameters. Highlight Project.params, and select Open to navigate to the project parameters window shown in Figure 44.

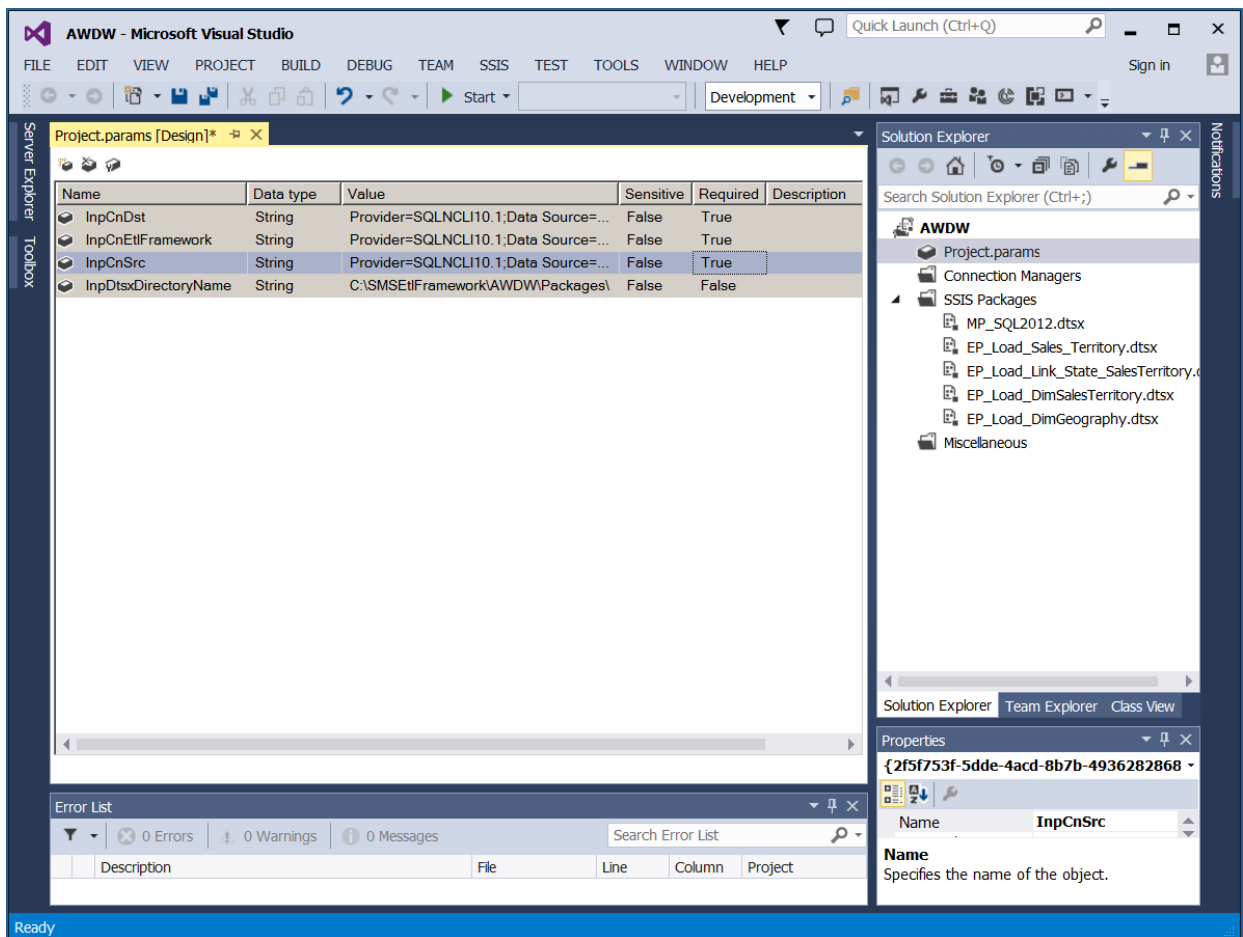


Figure 44: Project parameters window

Notice that you can set the value, add a description, and set the Sensitive and Required flags. Setting the Required flag will ensure that the parameter value is provided at runtime (manually changed for three project parameters as shown in Figure 44). This is a great addition since it eliminates a common SSIS issue with prior versions: running in production with values set in development or test.

Setting the Sensitive attribute to true allows sensitive information to be set at the parameter level. When this occurs, all logging and UI screens will not display sensitive information within the parameter. The scenario for setting the sensitive attribute is when connection strings contain sensitive information, such as usernames and passwords.

The final step is to delete the InpDtsxDirectoryName project parameter. This parameter was used for dynamically configuring the DTSX package filename referenced in the master package's Execute Package task. This is no longer required when you set your package type reference type to Project Reference.

To see how this converted, open EP_Load_Sales_Territory in the Execute Package Task Editor, as shown in Figure 45. Note that this is a partial screenshot to reduce the picture size.

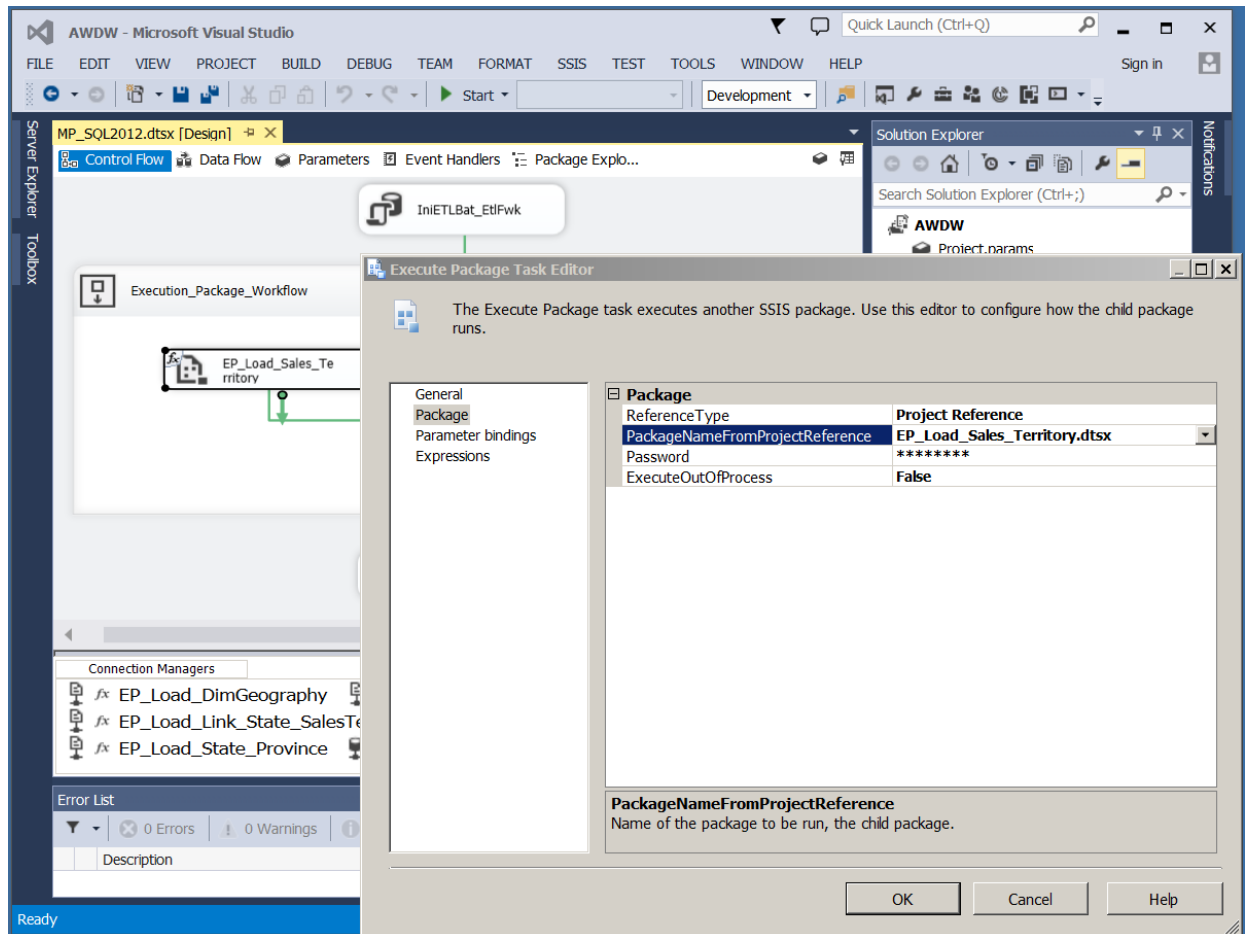


Figure 45: Execute Package Task Editor

Notice how the reference type is now Project Reference, as opposed to External Reference. This is possible because the project is self-contained, which eliminates the potential issue of setting and referencing an invalid directory and file.

The SSIS 2014 Execute Package Task Editor also contains parameter bindings. This allows the parameters to be defined and set from the parent package. Selecting *Parameter bindings* in the left pane navigates you to the parameter bindings' values shown in Figure 46.

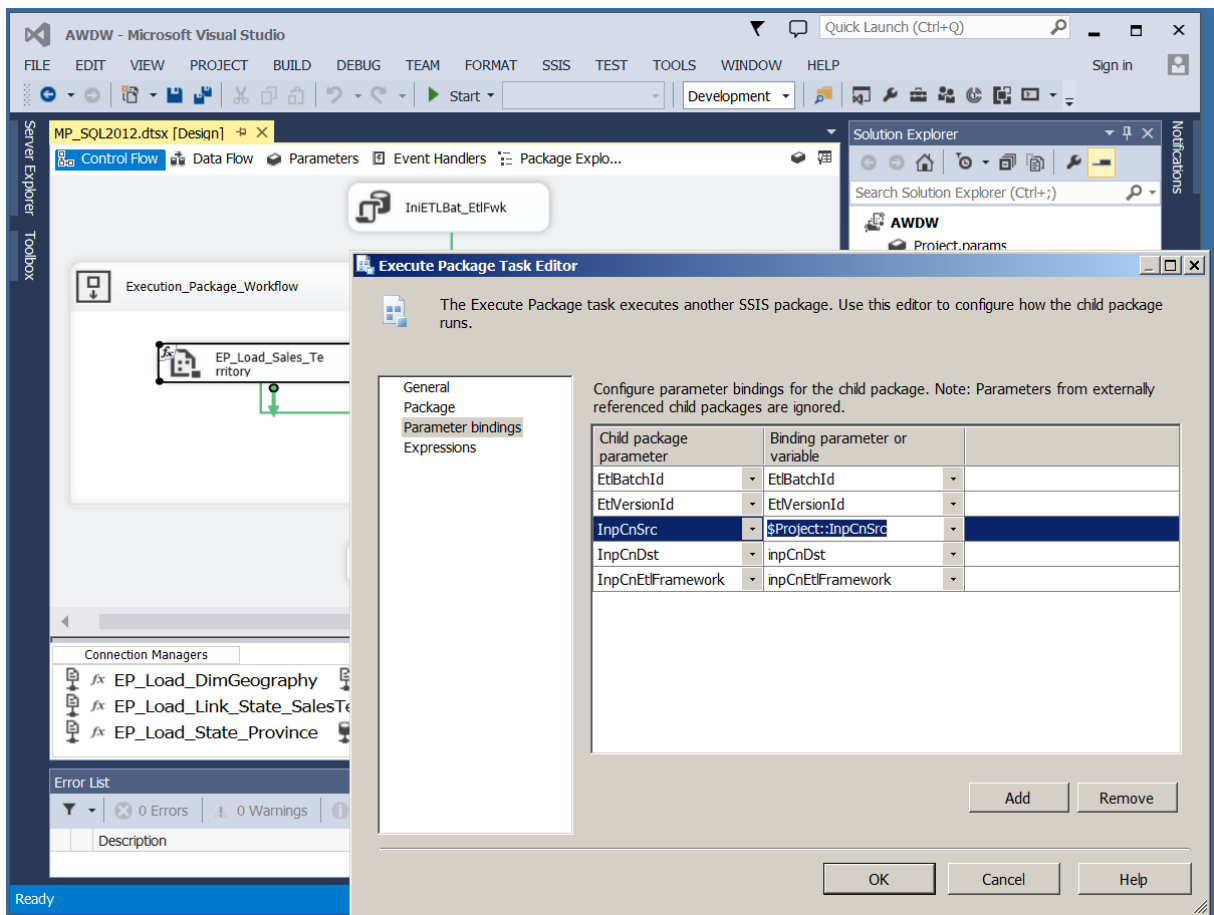


Figure 46: Parameter bindings screen

In Figure 46, notice that the child package parameter named `InpCnSrc` has been manually remapped to a project parameter named `$Project::InpCnSrc`. SSIS 2014 sets the child package values from the parent. Previous versions of SSIS worked in the other direction—that is, the execute package would use parent package configurations to initialize its values. You can also use project-level variables to initialize child package variables.

This isn't required, but it does eliminate a level of indirection. The initial model was for the package configuration to load a master package variable, which in turn was used to load the execution package variable. Put another way, rather than the `$Project::InpCnSrc` parameter loading the `inpCnSrc` variable, which is then bound as a parameter, you can directly bind `$Project::InpCnSrc`.

It should be noted that execution package connection string configuration can also be implemented using SSIS 2014 shared data connections. For more information on shared connection managers, see [What's new \(Integration Services\)](http://msdn.microsoft.com/en-us/library/bb522534(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/bb522534\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/bb522534(v=sql.120).aspx)).

The last screenshot, shown in Figure 47, is the converted version of the master package.

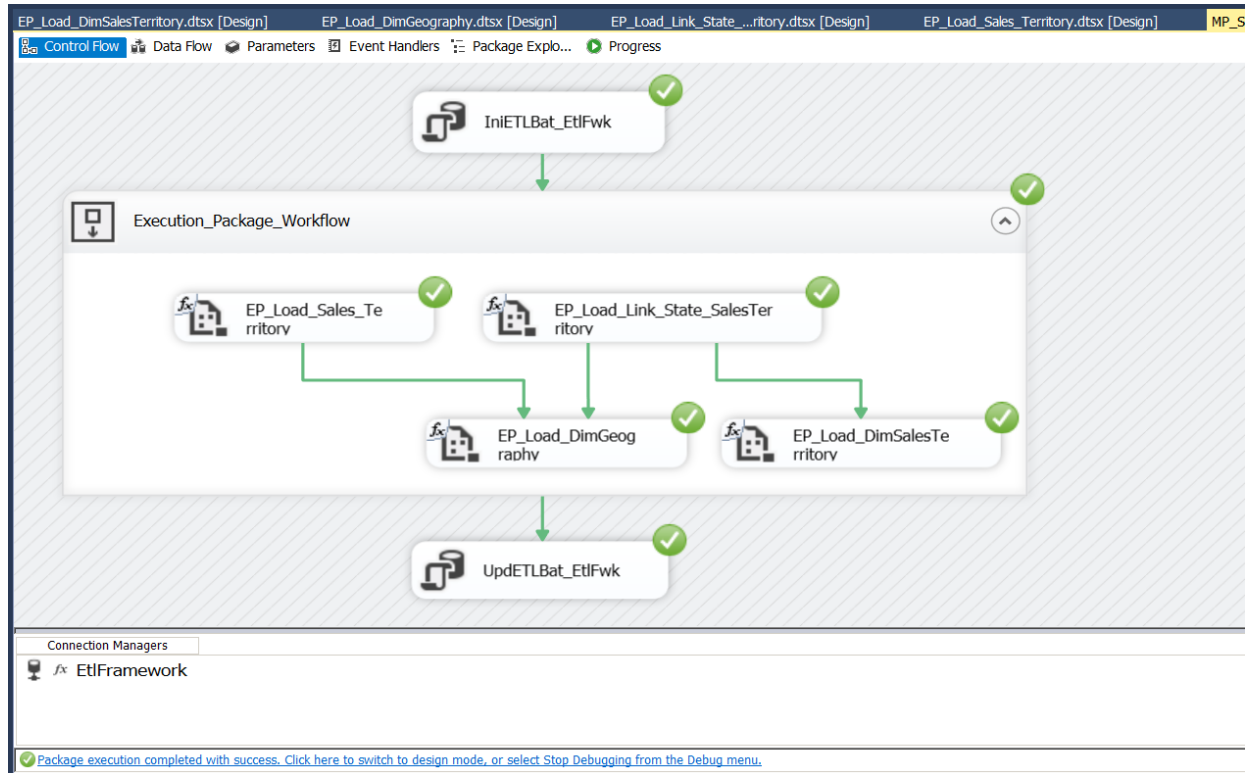


Figure 47: Master package after the conversion

Notice how the master package executed successfully after the conversion. Also notice that file connections are no longer required since the reference type has been changed to Project Reference, which was shown in Figure 45.

Now that you have finished the conversion, the next step is to deploy the project, which is out of the scope of this upgrade guide chapter. For more information on project deployments, see [Deployment Projects and Packages](http://msdn.microsoft.com/en-us/library/hh213290(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/hh213290\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/hh213290(v=sql.120).aspx)). For more information on the SSIS 2014 server, see [Integration Services \(SSIS\) Server](http://msdn.microsoft.com/en-us/library/gg471508(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/gg471508\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/gg471508(v=sql.120).aspx)).

In summary, it is highly recommended that you consider converting to SSIS 2014's project deployment model even if you are currently using an existing ETL Framework. In the provided sample, the legacy SSIS configurations were replaced with SSIS 2014 parameters. This reduces the moving parts within a solution, which in turn reduces TCO.

SSIS 2014 preserves the ETL Framework custom logging and batch/lineage logic while improving logging capabilities. These logging capabilities combined with the use of project and package parameters reduce the need for third-party ETL Frameworks. It is

recommended that any new development utilize the built-in SSIS 2014 features unless there is already a widely adopted and integrated ETL Framework in place.

Additional References

The following resources provide additional information on upgrading to SSIS 2014 as well as on the SSIS 2012 and SSIS 2014 feature set and new capabilities:

- Jamie Thomson's blog [SSIS Junkie](http://sqlblog.com/blogs/jamie_thomson/default.aspx)
(http://sqlblog.com/blogs/jamie_thomson/default.aspx)
- [SQL Server Integration Services](http://msdn.microsoft.com/en-us/sqlserver/cc511477.aspx)
(<http://msdn.microsoft.com/en-us/sqlserver/cc511477.aspx>)

Chapter 18: Reporting Services

Introduction

Reporting Services in SQL Server 2014 is largely unchanged from the prior two product versions. Notable changes to SQL Server Reporting Services (SSRS) for SQL Server 2014 are related to the Visual Studio report design environment, called SQL Server Data Tools for Business Intelligence (SSDT-BI). Refer to Chapter 15 for more information about SSDT-BI and the versions of Visual Studio that support SQL Server 2014 report development.

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 (DBAs), 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 R2, SSRS was updated with significantly new features, ease-of-use improvements and some incremental improvements in later versions. Customers currently using SSRS 2005/2008/2008 R2/2012 need to determine the best approach to upgrading their existing reports and/or their existing environment to SSRS in SQL Server 2014. The options available for upgrading depend on how your SSRS 2005/2008/2008 R2/2012 environment is currently deployed and what level of availability and upgrade testing you need.

For more information about deploying topologies in SSRS native mode, see [Planning a Deployment Topology](http://msdn.microsoft.com/en-us/library/ms157293(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms157293\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms157293(v=sql.120).aspx)).

For more information about deploying topologies in SSRS in SharePoint integrated mode, see [Deployment Topologies for Reporting Services in SharePoint Integrated Mode](http://msdn.microsoft.com/en-us/library/bb510781(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/bb510781\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/bb510781(v=sql.120).aspx)).

For more information about SSRS native mode scale-out configuration, see [Configure a Native Mode Report Server Scale-Out Deployment \(SSRS Configuration Manager\)](http://msdn.microsoft.com/en-us/library/ms159114.aspx) (<http://msdn.microsoft.com/en-us/library/ms159114.aspx>).

Reporting Services Editions

SSRS comes with all core editions of SQL Server 2014, with each edition meant to address specific reporting needs throughout an organization. You can review the full list of features available in each edition by reading [Features Supported by the Editions of SQL Server 2014](http://technet.microsoft.com/en-us/library/cc645993(v=sql.120).aspx) ([http://technet.microsoft.com/en-us/library/cc645993\(v=sql.120\).aspx](http://technet.microsoft.com/en-us/library/cc645993(v=sql.120).aspx)) in SQL Server 2014 Books Online.

In addition, you can find more information about SSRS versions and editions in [How to: Detect Version Information \(Reporting Services\)](http://msdn.microsoft.com/en-us/library/bb630446(SQL.120).aspx) ([http://msdn.microsoft.com/en-us/library/bb630446\(SQL.120\).aspx](http://msdn.microsoft.com/en-us/library/bb630446(SQL.120).aspx)) in SQL Server 2014 Books Online.

Note that in this chapter, we refer to SQL Server 2005 with Service Pack (SP4) simply as "SQL Server 2005" to simplify the text and readability.

It is generally recommended that you upgrade each edition of SSRS 2005/2008/2008 R2/2012 to the same edition of SSRS 2014. However, certain cross-edition upgrades are supported. Specifically, you can upgrade the Standard Edition of SSRS 2005/2008/2008 R2/2012 to the Enterprise Edition of SSRS 2014 (as well as to the Standard Edition of SSRS 2014).

If you upgrade the Standard Edition of SSRS 2005/2008/2008 R2/2012 to the Business Intelligence or Enterprise Edition of SSRS 2014, you will be able to use some new features, such as data-driven subscriptions, custom security extensions, scale-out capabilities (alerting and Power View in SharePoint integrated mode). For information about the new features, see the "Reporting Services" section in [Features Supported by the Editions of SQL Server 2014](http://msdn.microsoft.com/en-us/library/cc645993(v=SQL.120).aspx#reporting) ([http://msdn.microsoft.com/en-us/library/cc645993\(v=SQL.120\).aspx#reporting](http://msdn.microsoft.com/en-us/library/cc645993(v=SQL.120).aspx#reporting)).

For complete information about version and edition upgrade paths, see Chapter 1, "Upgrade Planning and Deployment," and [Supported Version and Edition Upgrades](http://msdn.microsoft.com/en-us/library/ms143393(v=SQL.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms143393\(v=SQL.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143393(v=SQL.120).aspx)) in SQL Server 2014 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 2014.

Upgrade Considerations

When upgrading from SSRS 2005/2008/2008 R2/2012 to SSRS 2014, you should consider the following possible issues.

SSRS is installed as a default instance. If the report server database resides within the default instance of SQL Server 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 2014 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 2014. 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 (2008 SP2 or later) 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, on startup of the upgraded Report Server service, the auto-upgrade feature modifies the table structures of the report server database to reflect the schema needed for SSRS 2014. The SQL Server 2014 Report Server service will continue to connect to the SQL Server (SQL Server 2008 SP2 or later) relational engine, with the new database schema in place.

SSRS includes client and server components. If you upgrade an SSRS 2005 installation to SSRS 2014 (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 2014 server, report developers might see a disparity between report preview in Report Designer and how the report is rendered at runtime. Note, however, that once you upgrade Report Designer on a given client, you can no longer use it to publish reports to an SSRS 2005 server. Report namespace differences prevent publishing to the prior version of the report server, but you can still use Report Designer from SSRS 2014 to deploy to SQL Server 2008 or later by setting the TargetServerVersion property.

If you have the client components of SSRS 2005/2008/2008 R2/2012 installed on a report server, upgrading the server to SSRS 2014 will remove them. If you need the previous SSRS 2005/2008/2008 R2/2012 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 2005/2008/2008 R2/2012 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 an SSRS 2005 report server, simply stop Microsoft

Internet Information Services (IIS) and the Reporting Services Windows service. To stop an SSRS 2008/2008 R2/2012 report server, you should use Reporting Services Configuration Manager to stop the report server instance. When the first report server is upgraded, the shared report server database will be upgraded. After finishing the upgrades, simply restart 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 2014 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 2014.
- Before upgrading SQL Server, enable Windows Authentication for SQL Server Agent and verify the default configuration. (For example, verify that the SQL Server Agent service account is a member of the SQL Server sysadmin group.)
- Before upgrading from one edition of SQL Server 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 component in [Features Supported by the Editions of SQL Server 2014](http://technet.microsoft.com/en-us/library/cc645993(v=sql.120).aspx) ([http://technet.microsoft.com/en-us/library/cc645993\(v=sql.120\).aspx](http://technet.microsoft.com/en-us/library/cc645993(v=sql.120).aspx)) in SQL Server 2014 Books Online.
- 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 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 re-create any logins and other user objects in the master, msdb, and model system databases. For details about version and edition upgrade paths, see [Supported Version and Edition Upgrades](http://msdn.microsoft.com/en-us/library/ms143393(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms143393\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143393(v=sql.120).aspx)).
- To upgrade to SQL Server 2014, you must be running a supported operating system. You can review the hardware and software requirements for SQL Server 2014 by reading [Hardware and Software Requirements for Installing SQL Server 2014](http://technet.microsoft.com/en-us/library/ms143506(v=sql.120).aspx) ([http://technet.microsoft.com/en-us/library/ms143506\(v=sql.120\).aspx](http://technet.microsoft.com/en-us/library/ms143506(v=sql.120).aspx)).
- 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 standalone 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 [Upgrade a SQL Server Failover Cluster Instance \(Setup\)](http://technet.microsoft.com/en-us/library/ms191295(v=sql.120).aspx) ([http://technet.microsoft.com/en-us/library/ms191295\(v=sql.120\).aspx](http://technet.microsoft.com/en-us/library/ms191295(v=sql.120).aspx)). For more information about database migration, see [Use the Copy Database Wizard](http://technet.microsoft.com/en-us/library/ms188664(v=sql.120).aspx) ([http://technet.microsoft.com/en-us/library/ms188664\(v=sql.120\).aspx](http://technet.microsoft.com/en-us/library/ms188664(v=sql.120).aspx)) in SQL Server 2014 Books Online.
- To upgrade SQL Server 2005 to SQL Server 2014 on a computer that is running Windows Server 2008, you must be running SQL Server 2005 SP4.

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

You can upgrade SSRS 2005/2008/2008 R2/2012 installations to SSRS 2014 in one of two ways: through an in-place upgrade (supported by Setup) or a side-by-side migration (installing a clean SQL Server 2014 instance and then moving data and metadata from SQL Server 2005/2008/2008 R2/2012 to SQL Server 2014).

In-Place Upgrade

With an in-place upgrade, SSRS 2005/2008/2008 R2/2012 is removed and replaced by SSRS 2014. During the upgrade process, the SSRS databases are upgraded, and users will not be able to access SSRS 2005/2008/2008 R2/2012 reports. After the in-place upgrade is complete, only SSRS 2014 will remain. With an in-place upgrade, you test SSRS 2014 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 2005/2008/2008 R2/2012 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 2014 is completed, you need to uninstall SSRS 2014, reboot, reinstall SSRS 2005/2008/2008 R2/2012, and then restore the SSRS 2005/2008/2008 R2/2012 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 2014 alongside SSRS 2005/2008/2008 R2/2012, which remains until uninstalled. During the upgrade process, users can continue to access the SSRS 2005/2008/2008 R2/2012 reports (unaffected by the upgrade process), but performance might be slower. After SSRS 2014 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 re-create server settings on a new server. For more information about this process, see [Migrate a Reporting Services Native Mode Installation](http://technet.microsoft.com/en-us/library/ms143724(v=sql.120).aspx) ([http://technet.microsoft.com/en-us/library/ms143724\(v=sql.120\).aspx](http://technet.microsoft.com/en-us/library/ms143724(v=sql.120).aspx)).

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 1 shows which upgrade options can be applied to the 2005/2008/2008 R2/2012 configurations described earlier in this chapter. Note that you can use these options regardless of which edition of SSRS 2005/2008/2008 R2/2012 is in place.

Table 1: Upgrade Options for SSRS 2005/2008/2008 R2/2012

Reporting Services 2005/2008/2008 R2/2012 Configuration	In-Place Upgrade?	Side-by-Side Upgrade?
Single-server installation (2005/2008/2008 R2/2012)	Yes	Yes
Remote catalog installation on SQL Server 2005	Yes	Yes
Scale-out installation (2005/2008/2008 R2/2012)	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 2005/2008/2008 R2/2012 installation. Thus, planning for a failed upgrade process is critical.

For detailed, step-by-step instructions on how to prepare for an upgrade, see Section

14.2 “Preparing to Upgrade” in Chapter 14 of the [SQL Server 2008 R2 Upgrade Technical Reference Guide](http://download.microsoft.com/download/3/0/D/30DB8D46-8ACF-442A-99A2-0F4CE74AE14D/SQL_Server_2008_R2_Upgrade_Technical_Reference_Guide.docx) (http://download.microsoft.com/download/3/0/D/30DB8D46-8ACF-442A-99A2-0F4CE74AE14D/SQL_Server_2008_R2_Upgrade_Technical_Reference_Guide.docx).

Important Reporting Services 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, through configuration tools, through 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 [Modify a Reporting Services Configuration File \(RSreportserver.config\)](http://technet.microsoft.com/en-us/library/bb630448(v=sql.120).aspx) (http://technet.microsoft.com/en-us/library/bb630448(v=sql.120).aspx) in SQL Server 2014 Books Online.

Note: To review a list of which settings were deleted or moved, see [Breaking Changes in SQL Server Reporting Services in SQL Server 2014](http://technet.microsoft.com/en-us/library/ms143380(v=sql.120).aspx) (http://technet.microsoft.com/en-us/library/ms143380(v=sql.120).aspx) in SQL Server 2014 Books Online.

Storing Configuration Settings

Most configuration settings are stored in configuration files included with SSRS. For detailed, step-by-step instructions on how to prepare for an upgrade, see Section 14.2.2 “Storing Configuration Settings” in Chapter 14 of the [SQL Server 2008 R2 Upgrade Technical Reference Guide](http://download.microsoft.com/download/3/0/D/30DB8D46-8ACF-442A-99A2-0F4CE74AE14D/SQL_Server_2008_R2_Upgrade_Technical_Reference_Guide.docx) (http://download.microsoft.com/download/3/0/D/30DB8D46-8ACF-442A-99A2-0F4CE74AE14D/SQL_Server_2008_R2_Upgrade_Technical_Reference_Guide.docx).

Deprecated Features

This section describes SSRS features that have been deprecated and will not be supported in future releases.

Semantic Modeling Language Report Models

Semantic modeling language (SMDL) report models have been deprecated. Although you can continue to use existing SMDL report models as data sources in SQL Server 2014 Reporting Services reports, you should update your reports to remove their dependency on them.

For complete information about deprecated functionality in SSRS 2014, see [Deprecated Features in SQL Server Reporting Services in SQL Server 2014](http://technet.microsoft.com/en-us/library/ms143509(v=sql.120).aspx) ([http://technet.microsoft.com/en-us/library/ms143509\(v=sql.120\).aspx](http://technet.microsoft.com/en-us/library/ms143509(v=sql.120).aspx)) in SQL Server 2014 Books Online.

Discontinued Functionality

There are no discontinued features in SSRS 2012 or SSRS 2014. Before you upgrade, see [Discontinued Functionality to SQL Server Reporting Services in SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms144231(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms144231\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms144231(v=sql.120).aspx)) in SQL Server 2014 Books Online for any last-minute changes.

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 2014 Upgrade Advisor identifies many breaking changes. Chapter 1, "Upgrade Planning and Deployment," and [Use Upgrade Advisor to Prepare for Upgrades](http://msdn.microsoft.com/en-us/library/ms144256(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms144256\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms144256(v=sql.120).aspx)) describe how to use this tool to find and fix problems before the upgrade.

Report Model Designer Not Available in SSDT

Report Model projects (introduced in SSRS 2005) are no longer supported by SSDT, and the Report Model designer is not available in SSRS 2014. You cannot create new Report Model projects or open existing projects in SSDT, and you cannot create or update report models. To update report models, you can use SSRS 2008 R2 or earlier tools. You can continue to use report models as data sources in reports authored in SSRS 2014 tools such as Report Builder and Report Designer. The query designer that you use to create queries to extract report data from report models continues to be available in SSRS 2014.

For complete information about breaking changes in SSRS 2014, see [Breaking Changes in SQL Server Reporting Services in SQL Server 2014](http://technet.microsoft.com/en-us/library/ms143380(v=sql.120).aspx) (http://technet.microsoft.com/en-us/library/ms143380(v=sql.120).aspx) in SQL Server 2014 Books Online or Section 14.2.5 “Breaking Changes” in Chapter 14 of the [SQL Server 2008 R2 Upgrade Technical Reference Guide](http://download.microsoft.com/download/3/0/D/30DB8D46-8ACF-442A-99A2-0F4CE74AE14D/SQL_Server_2008_R2_Upgrade_Technical_Reference_Guide.docx) (http://download.microsoft.com/download/3/0/D/30DB8D46-8ACF-442A-99A2-0F4CE74AE14D/SQL_Server_2008_R2_Upgrade_Technical_Reference_Guide.docx).

Behavior Changes

There are a number of behavior changes in this release that might require corrective action after the upgrade is complete. In this section, we look at fundamental changes to this release functionality that might affect how you work.

View Items Permission Will Not Download Shared Datasets (SharePoint Mode)

In SSRS 2014, users with the SharePoint permission of View Items can no longer download the contents of Reporting Services shared data sets like they did in previous SSRS versions. This behavior change is now consistent with the View Items permissions for reports, data sources, and models.

For complete information about the behavior changes in SSRS 2014, see [Behavior Changes in SQL Server Reporting Services in SQL Server 2014](http://technet.microsoft.com/en-us/library/ms143200(v=sql.120).aspx) (http://technet.microsoft.com/en-us/library/ms143200(v=sql.120).aspx) in SQL Server 2014 Books Online.

Updating Report Projects and Definitions for Use in SSDT-BI

You must update existing report projects and definitions for use within the Visual Studio/SSDT-BI report designer so that you can make updates and changes to existing reports. When an existing report project is opened within SSDT-BI, the project will need to be upgraded to the new format. After opening the project, the SSDT-BI environment will launch the Visual Studio Conversion Wizard, which you can use to perform the upgrade.

For complete information about updating report projects and definitions, see Section 14.2.7 “Updating Report Projects and Definitions for Use in BI Development Studio” in Chapter 14 of the [SQL Server 2008 R2 Upgrade Technical Reference Guide](http://download.microsoft.com/download/3/0/D/30DB8D46-8ACF-442A-99A2-0F4CE74AE14D/SQL_Server_2008_R2_Upgrade_Technical_Reference_Guide.docx) (http://download.microsoft.com/download/3/0/D/30DB8D46-8ACF-442A-99A2-0F4CE74AE14D/SQL_Server_2008_R2_Upgrade_Technical_Reference_Guide.docx).

Important: Once a report has been converted to the SSRS 2008 or SSRS 2008 R2 schema, it can no longer be published to an SSRS 2005 instance. However, SSRS 2014 can read previous version RDLs without conversion.

In the SQL Server 2014 version of SSDT-BI, you can work with SQL Server 2008 or SQL Server 2008 R2 versions of report definitions and Report Server projects. You can edit, preview, and deploy any version of the reports.

Upgrade Tools

SQL Server 2014 Upgrade Advisor helps you prepare for upgrades to SQL Server 2014. 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 information about how to install and run Upgrade Advisor, see Chapter 1, "Upgrade Planning and Deployment."

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. You can also back up or detach databases from a 32-bit instance of SQL Server and then restore or attach them to a 64-bit instance of SQL Server if the databases are not published in replication. In this case, you must also re-create any logins and other user objects in the 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 2005/2008/2008 R2/2012 to SSRS 2014, there is a range of potential issues you might face during an upgrade, as you saw earlier in this 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 Upgrade Advisor's recommendations and guidance for possible mitigation options and strategies. Chapter 1 also lists those issues that cannot be detected by Upgrade Advisor or whose detection would result in too many false-positive results.

Review 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 2014, see [Reporting Services Backward Compatibility](http://technet.microsoft.com/en-us/library/ms143251(v=sql.120).aspx) ([http://technet.microsoft.com/en-us/library/ms143251\(v=sql.120\).aspx](http://technet.microsoft.com/en-us/library/ms143251(v=sql.120).aspx)) in SQL Server 2014 Books Online.

For a complete list of the SSRS upgrade issues that Upgrade Advisor detects, see “Reporting Services Upgrade Issues” in the SQL Server 2014 Upgrade Advisor Help file.

For more information about known issues and workarounds, see Section 14.2.10 “Known Issues and Workarounds” in Chapter 14 of the [SQL Server 2008 R2 Upgrade Technical Reference Guide](http://download.microsoft.com/download/3/0/D/30DB8D46-8ACF-442A-99A2-0F4CE74AE14D/SQL_Server_2008_R2_Upgrade_Technical_Reference_Guide.docx) (http://download.microsoft.com/download/3/0/D/30DB8D46-8ACF-442A-99A2-0F4CE74AE14D/SQL_Server_2008_R2_Upgrade_Technical_Reference_Guide.docx).

Backup and Rollback Plan

Before upgrading to SSRS 2014, 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 2014.
2. Use the System Configuration Checker (SCC) to scan the report server for any conditions that might prevent a successful installation of SQL Server 2014. For more information, see [Check Parameters for the System Configuration Checker](http://msdn.microsoft.com/en-us/library/ms143753(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms143753\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143753(v=sql.120).aspx)).
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(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms144228\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms144228(v=sql.120).aspx)) and Chapter 5, “Database Security,” in this guide.
4. Run Upgrade Advisor on the report server to determine any issues that might prevent you from successfully upgrading.
5. Back up your symmetric key. For details, see [Back Up and Restore Reporting Services Encryption Keys \(SSRS Native Mode\)](http://msdn.microsoft.com/en-us/library/ms157275(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms157275\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms157275(v=sql.120).aspx)) in SQL Server 2014 Books Online.
6. Back up your report server databases. For details, see [Moving the Report Server Databases to Another Computer](http://msdn.microsoft.com/en-us/library/ms156421(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms156421\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms156421(v=sql.120).aspx)) in SQL Server 2014 Books Online.
7. Back up the following report server configuration files:

- a. Rsreportserver.config
- b. Rswebapplication.config
- c. Rssvrpolicy.config
- d. Rsmgrpolicy.config
- e. Reportingserviceservice.exe.config
- f. Web.config (for both the report server and Report Manager ASP.NET applications)
- g. 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 [Upgrade and Migrate Reporting Services](http://msdn.microsoft.com/en-us/library/ms143747(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms143747\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143747(v=sql.120).aspx)).

Upgrading from SQL Server 2005

You can upgrade SSRS 2005 to SSRS 2014 in one of two ways: through an in-place upgrade or a side-by-side migration.

In-Place Upgrade

If you are performing an in-place upgrade of an SSRS 2005 installation to SSRS 2014, you need to know the following: If SSRS 2005 is installed as a default instance and the report server database resides within the default instance of SQL Server 2005 on the same server, the relational engine and the report server must be upgraded together. You cannot upgrade the Reporting Services component without upgrading the relational engine. If the report server database resides within a named instance of SQL Server on the same server or resides on a remote server (SQL Server 2005), you must upgrade the Database Engine instance hosting the remote catalog first and then upgrade the Reporting Services component.

In an in-place upgrade of an SSRS 2005 installation to SSRS 2014, 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 2005 to SSRS 2014 (you follow the same steps for an in-place upgrade from SSRS 2008/2008 R2/2012 to SSRS 2014):

1. Insert the SQL Server installation media. 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 server operating system that hosts the SSRS instance you want to upgrade does not meet the minimum requirements for SQL Server 2014, Setup will block the setup and display a message specifying the minimum requirements. Click OK, install the minimum requirements, and then restart Setup.
3. When the prerequisites are installed, the Installation Wizard lets you go forward and launch the SQL Server Installation Center. To upgrade an existing instance of SQL Server, click the *Upgrade from SQL Server 2005, SQL Server 2008, SQL Server 2008 R2, or SSRS 2012* option on the installation page. Figure 1 shows the upgrade selection screen.

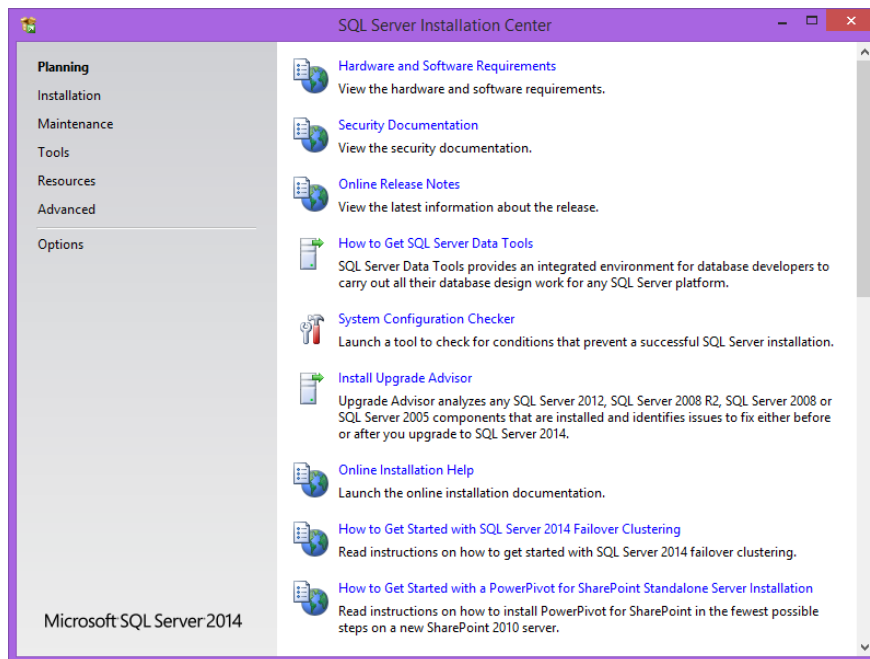


Figure 1: Upgrade selection page

4. The Setup Support Rules process will check all the rules concerning the setup files. If any rule reports a failure, you must correct the problem, which you can

do without closing Setup. After fixing the problem, you need to rerun the Setup Support Rules process by clicking the *Re-run* button. Click Next to continue.

5. 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.
6. 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.
7. The Product Updates window lets Setup install the latest updates. Click Next to look for the latest updates and include them in the installation process.
8. The Setup Support Rules process again checks for issues that might prevent the proper installation of the Setup support files. If any rule reports a failure, you must correct the problem, which you can do without closing Setup. After fixing the problem, you need to rerun the Setup Support Rules process by clicking the *Re-run* button. Click Next to continue.
9. On the Select Instance page, specify the instance of SQL Server to upgrade. Figure 2 shows the Select Instance screen. 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 2014. To continue, click Next.

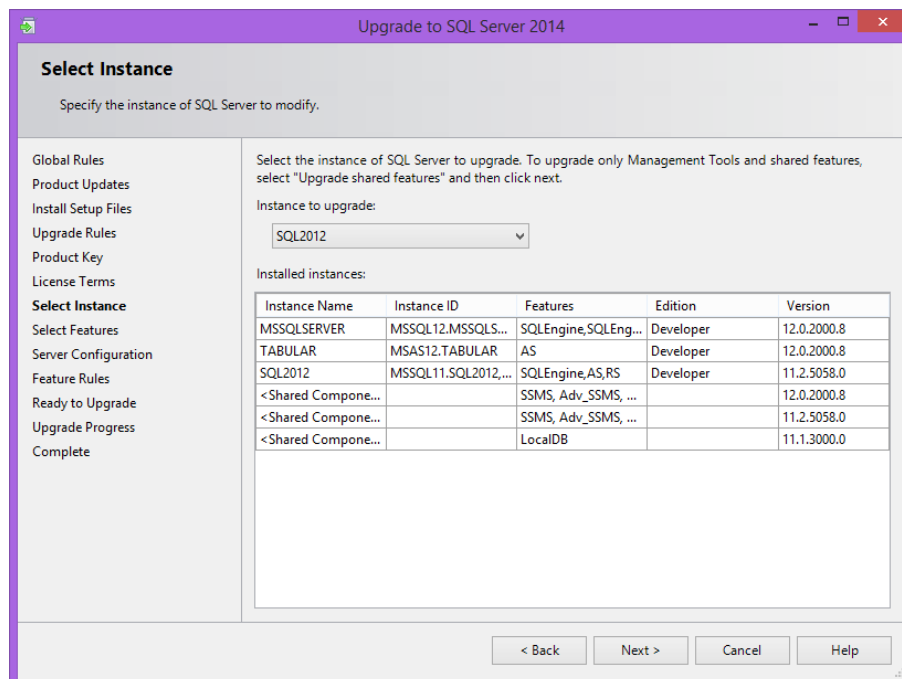


Figure 2: Select Instance screen

10. 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 2014 after the upgrade operation is complete, see [Add Features to an Instance of SQL Server 2014 \(Setup\)](http://msdn.microsoft.com/en-us/library/cc281940(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/cc281940\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/cc281940(v=sql.120).aspx)) in SQL Server 2014 Books Online. Figure 3 shows the Select Features screen. Click Next to continue.

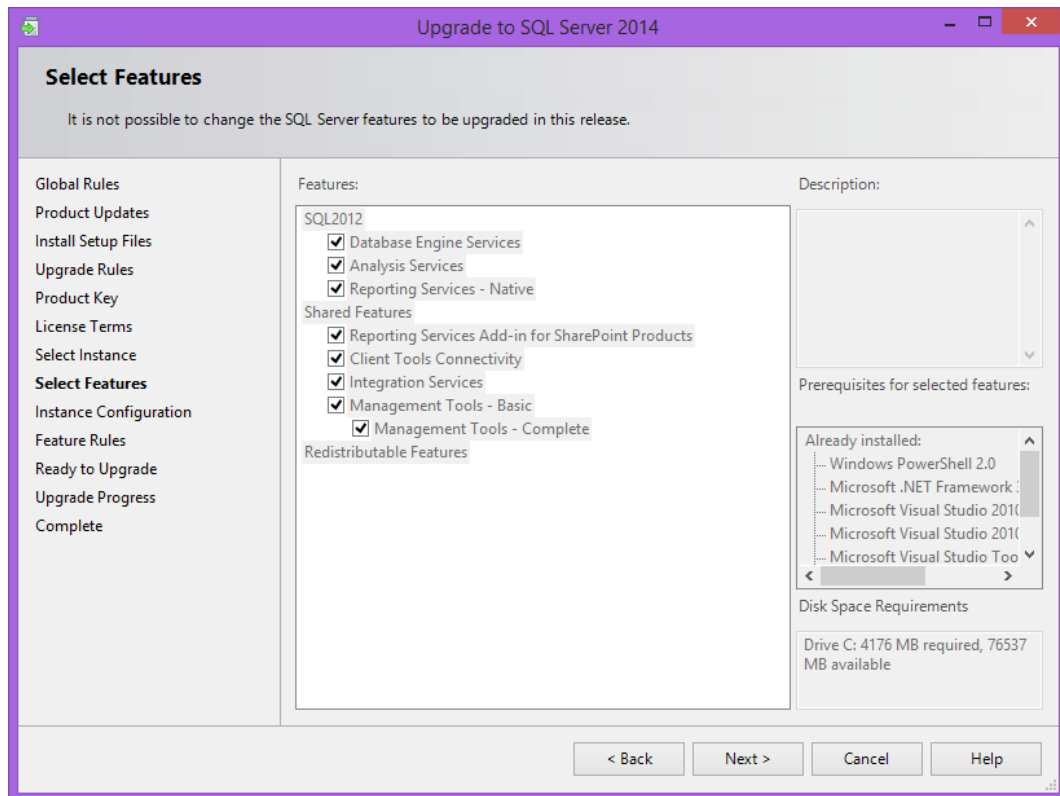


Figure 3: Select Features screen

11. In the Instance Configuration page, you are prompted to specify the name and instance ID for the instance of SQL Server. The instance ID becomes part of the installation path. This is used to identify 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 would be MSSQLSERVER. To use a non-default instance ID, specify a different value in the Instance ID text box. Click Next to continue.

12. The Disk Space Requirements page calculates the required disk space for the features you specify and then compares the required disk space to the available disk space on the computer where Setup is running. Click Next to continue.
13. In the Service Accounts tab on the Server Configuration page, you can specify the login accounts for the SQL Server services. The actual services that are configured on this page depend on the features installed. Click Next to continue.
Note: The workflow 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.
14. On the Full-Text Search Upgrade Options page, specify the upgrade options for the databases being upgraded. For more information, see Chapter 6, "Full-Text Search," in this guide.
15. 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.
16. The Setup Support Rules process will check one more set of rules to validate your computer configuration with the SQL Server features you have specified before the upgrade operation begins. If some rule failed, you should install the needed component or apply the corrective action. For example, if Microsoft .NET Framework 3.5 SP1 is not installed, the Setup Support Rules process will prompt you to install the feature, depending on the operating system used.
17. The Ready to Upgrade page displays a tree view of upgrade options that were specified during Setup. To continue, click Install. During the upgrade, the Upgrade Progress page provides a status bar so that you can monitor the progress as Setup proceeds.
18. After installation, the Complete page (see Figure 4) provides a link to the summary log file for the installation and other important notes. For information about Setup log files, see [View and Read SQL Server Setup Log Files](http://msdn.microsoft.com/en-us/library/ms143702(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms143702\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143702(v=sql.120).aspx)). To complete the installation process, click Close.

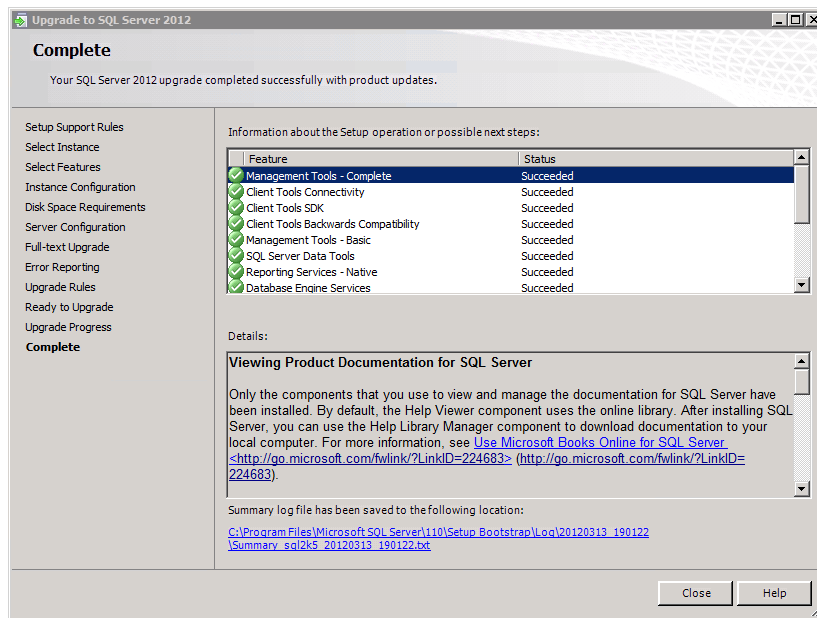


Figure 4: Complete screen

19. 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 more information about upgrading to SQL Server 2014, see [Upgrade to SQL Server 2014 Using the Installation Wizard \(Setup\)](http://msdn.microsoft.com/en-us/library/ms144267(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms144267\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms144267(v=sql.120).aspx)).

Side-by-Side Upgrade

You can upgrade SSRS 2005 installations to SSRS 2014 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 2014 alongside the existing SSRS 2005 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 2014 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, the workstations and computers hosting the Report Designer or SSRS 2005 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 2014. 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 2014 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 2014 report server, you can install SSRS 2014 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.

For information about installing SQL Server 2014, see [Install SQL Server 2014 from the Installation Wizard \(Setup\)](http://msdn.microsoft.com/en-us/library/ms143219(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms143219\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143219(v=sql.120).aspx)).

For complete information about installing the new instance, see Section 14.3.2.1 “Installing the New Instance” in Chapter 14 of the [SQL Server 2008 R2 Upgrade Technical Reference Guide](http://download.microsoft.com/download/3/0/D/30DB8D46-8ACF-442A-99A2-0F4CE74AE14D/SQL_Server_2008_R2_Upgrade_Technical_Reference_Guide.docx) (http://download.microsoft.com/download/3/0/D/30DB8D46-8ACF-442A-99A2-0F4CE74AE14D/SQL_Server_2008_R2_Upgrade_Technical_Reference_Guide.docx).

Configuring the New Instance

After you have installed the new instance, you should use the Reporting Services Configuration tool to configure the instance. Launch the tool from the Configuration Tools group within the Microsoft SQL Server 2014 Start menu group.

For complete information about configuring the new instance, see Section 14.3.2.2 “Configuring the New Instance” in Chapter 14 of the [SQL Server 2008 R2 Upgrade Technical Reference Guide](http://download.microsoft.com/download/3/0/D/30DB8D46-8ACF-442A-99A2-0F4CE74AE14D/SQL_Server_2008_R2_Upgrade_Technical_Reference_Guide.docx) (http://download.microsoft.com/download/3/0/D/30DB8D46-8ACF-442A-99A2-0F4CE74AE14D/SQL_Server_2008_R2_Upgrade_Technical_Reference_Guide.docx).

How Schema, Metadata, and Report Server Content Is Updated

The report server database is upgraded in several stages:

1. The schema is upgraded automatically after setup and service startup or when you select a SQL Server 2005/2008/2008 R2/2012 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.

Note: Published reports and compiled report snapshots are updated on first use. For more information, see [Upgrade Reports](http://msdn.microsoft.com/en-us/library/ms143674(v=sql.120).aspx) (http://msdn.microsoft.com/en-us/library/ms143674(v=sql.120).aspx).

Review [Upgrade a Report Server Database](http://msdn.microsoft.com/en-us/library/ms403392(v=sql.120).aspx) (http://msdn.microsoft.com/en-us/library/ms403392(v=sql.120).aspx) for more information.

2. 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 2005 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 2005/2008/2008 R2/2012 is installed). Click Apply before changing to another section to save the changes.
3. 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 2005 instance. Figure 5 shows this option being used to restore a symmetric encryption key.

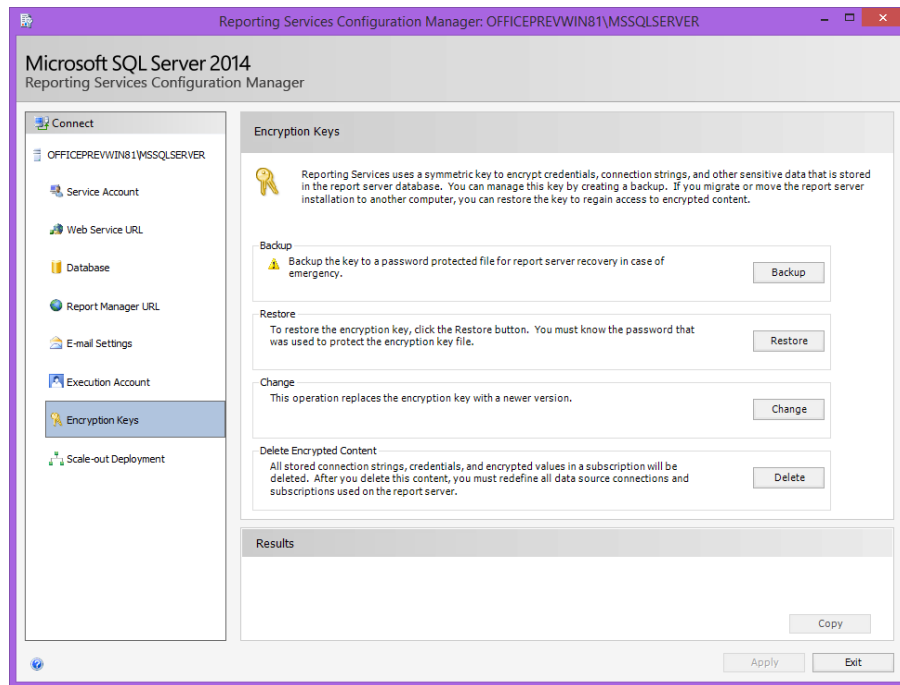


Figure 5: Restoring the symmetric encryption key

Note: If the symmetric encryption key has not yet been extracted from SSRS 2005, simply do so by using the `rskeymgmt` utility (found in the `<Drive:>\Program Files(x86)\Microsoft SQL Server\120\Tools\Binn\` if running x64). 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 not lose any report or sever content but will need to manually recreate any encrypted content (such as existing data source credentials) after the upgrade.

4. Use the configuration tool's Email Settings and Execution Account options to establish other extended configuration settings for SSRS 2014. Use these options to set these extended options for the new instance.
5. 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_SQL2014`, you can use the URL `http://MachineName/Reports_SQL2014` 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 2005 should be present and available within SSRS 2014.

Upgrading from SQL Server 2008

You can upgrade SSRS 2008 to SSRS 2014 in one of two ways: through an in-place upgrade or a side-by-side migration.

In-Place Upgrade

When upgrading an installation of SSRS 2008 in place to SSRS 2014, 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.

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 2008 SP2 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, on startup of the upgraded Report Server service, the auto-upgrade feature modifies the table structures of the report server database to reflect the schema needed for SSRS 2014. The SQL Server 2014 Report Server service will continue to connect to the SQL Server 2008 SP2 relational engine, with the new database schema in place. If a SQL Server 2005 Database Engine instance is hosting the report server database, you cannot upgrade the Reporting Services component without upgrading the relational engine. If the report server database resides within a named instance of SQL Server on the same server or resides on a remote server (SQL Server 2005), you must upgrade the Database Engine instance hosting the report server database first and then upgrade the Reporting Services component.

Note: The minimum product level requirements to upgrade SQL Server 2008 to SQL Server 2014 is SQL Server 2008 SP2 either remote catalog or default instance.

Upgrading via the Setup Application

Upgrading from SSRS 2008 to SSRS 2014 via the Setup application is identical to upgrading from SSRS 2005 via the Setup application. For detailed steps, see "Upgrading via the Setup Application" in the "Upgrading from SQL Server 2005" section earlier in this chapter.

Side-by-Side Upgrade

Upgrading from SSRS 2008 to SSRS 2014 via the side-by-side method is identical to

doing a side-by-side upgrade from SSRS 2005. For detailed steps, see “Side-by-Side Upgrade” in the “Upgrading from SQL Server 2005” section earlier in this chapter.

Installing the New Instance

The steps and options for installing the new instance are the same as in an SSRS 2005 upgrade. You can find those steps in the “Installing the New Instance” section under “Upgrading from SQL Server 2005.”

Configuring the New Instance

After you have installed the new SSRS 2014 instance, you should use the Reporting Services Configuration tool to configure it. Start the tool from the Configuration Tools group within the Microsoft SQL Server 2014 Start menu group.

The steps for configuring the new instance are the same as in an SSRS 2005 upgrade. You can find those steps in “Configuring the New Instance” in the “Upgrading from SQL Server 2005” section earlier in this chapter.

Upgrading from SQL Server 2008 R2 or SQL Server 2012

You can upgrade SSRS 2008 R2 or SSRS 2012 to SSRS 2014 in one of two ways: through an in-place upgrade or a side-by-side migration.

In-Place Upgrade

When upgrading an installation of SSRS 2008 R2 or SSRS 2012 in-place to SSRS 2014, 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.

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 2008 SP2 or SQL Server 2012 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, on startup of the upgraded Report Server service, the auto-upgrade feature modifies the table structures of the report server database to reflect the schema needed for SSRS 2014. The SQL Server 2014 Report Server service will continue to connect to the original relational engine instance, with the new database schema in place. If a SQL Server 2005 Database Engine instance is hosting the report server

database, you cannot upgrade the Reporting Services component without upgrading the relational engine. If the report server database resides within a named instance of SQL Server on the same server or resides on a remote server (SQL Server 2005), you must upgrade the Database Engine instance hosting the report server database first and then upgrade the Reporting Services component.

Upgrading via the Setup Application

Upgrading from SSRS 2008 R2 or SSRS 2012 to SSRS 2014 via the Setup application is identical to upgrading from SSRS 2005 via the Setup application. For detailed steps, see "Upgrading via the Setup Application" in the "Upgrading from SQL Server 2005" section earlier in this chapter.

Side-by-Side Upgrade

Upgrading from SSRS 2008 R2 or SSRS 2012 to SSRS 2014 via the side-by-side method is identical to doing a side-by-side upgrade from SSRS 2005. For detailed steps, see "Side-by-Side Upgrade" in the "Upgrading from SQL Server 2005" section earlier in this chapter.

Installing the New Instance

The steps and options for installing the new instance are the same as in an SSRS 2005 upgrade. You can find those steps in the "Installing the New Instance" section under "Upgrading from SQL Server 2005."

Configuring the New Instance

After you have installed the new SSRS 2014 instance, you should use the Reporting Services Configuration tool to configure it. Start the tool from the Configuration Tools group within the Microsoft SQL Server 2014 Start menu group.

The steps for configuring the new instance are the same as in an SSRS 2005 upgrade. You can find those steps in "Configuring the New Instance" in the "Upgrading from SQL Server 2005" section earlier in this chapter.

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\120\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 *YYMMDD_hhmmss*.

You can find detailed Setup logs at the following location: <drive>:\Program Files\Microsoft SQL Server\120\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 filename format is *Summary_[machine name]_timestamp_[execution phase]*. The final summary log is copied to %Program Files%\Microsoft SQL Server\120\Setup Bootstrap\Log folder and named *Summary.txt* for quick reference.

Note: Setup won't archive the log files in a .cab file.

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 [View and Read SQL Server Setup Log Files](http://msdn.microsoft.com/en-us/library/ms143702(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ms143702\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms143702(v=sql.120).aspx)).

Post-Upgrade Tasks

After upgrading to SSRS 2014 from SSRS 2005/2008/2008 R2/2012, it is important to ensure that the upgrade ran successfully and to configure SSRS 2014:

1. To begin, particularly if you performed an in-place upgrade, use the Reporting Services 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://localhost/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, at minimum, the Visual Studio 2008 CLR.
4. SSRS 2005, SSRS 2008, and SSRS 2008 R2 came with an ad hoc reporting tool called Report Builder (now referred to as Report Builder 1.0, not to be confused with Report Builder 3.0). Settings are available for backward compatibility. Although Report Builder 1.0 is a deprecated feature and should not be used for new development, the required security settings are still available in SSRS 2014. You can use SSMS to connect to the Reporting Services instance and change the existing security role definitions to provide end-user access to Report Builder 1.0. The existing role definitions are shown in Table 2. Again, we should emphasize that continued use of Report Builder models and Report Builder 1.0 reports is discouraged and won't be supported in the future.

Table 2: 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 2005 and SSRS 2014

When you upgrade an SSRS 2005 instance to SSRS 2014 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 2014 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. In addition, if a set of complex reports requires additional testing efforts, migrating each report individually could be beneficial.

For more information about manually move existing files, see Section 14.7.1 “Moving Reports Between SSRS 2000 and SSRS 2005, or SSRS 2005 and SSRS 2008 R2” in Chapter 14 of the [SQL Server 2008 R2 Upgrade Technical Reference Guide](http://download.microsoft.com/download/3/0/D/30DB8D46-8ACF-442A-99A2-0F4CE74AE14D/SQL_Server_2008_R2_Upgrade_Technical_Reference_Guide.docx) (http://download.microsoft.com/download/3/0/D/30DB8D46-8ACF-442A-99A2-0F4CE74AE14D/SQL_Server_2008_R2_Upgrade_Technical_Reference_Guide.docx).

Moving Reports Between SSRS 2008, SSRS 2008 R2, SSRS 2012, and SSRS 2014

A report definition file includes a reference to the RDL namespace that specifies the version of the report definition schema that is used to validate the .rdl file. In the versions of SSDT-BI for SQL Server 2014, you can work with both SQL Server 2008 and SQL Server 2008 R2 versions of report definitions and Report Server projects. You can edit, preview, and deploy either version of the reports.

If you open, update, and then save a SQL Server 2008 report definition, it is saved as a SQL Server 2008 report definition unless you added features that were introduced in SSRS 2008 R2. In such a case, the report definition is saved as an SSRS 2008 R2 report definition to ensure that the definition is valid and the report will run. For more information, see [Deployment and Version Support in SQL Server Data Tools \(SSRS\)](http://msdn.microsoft.com/en-us/library/ee635898(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/ee635898\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ee635898(v=sql.120).aspx)) in SQL Server 2014 Books Online.

When you open an .rdl file in Report Designer in SSDT-BI that was created for the SQL Server 2005 namespace, Report Designer automatically creates a backup file and upgrades the report to the current namespace. If you save the upgraded report definition, you have saved the converted .rdl file. As soon as you save it, you cannot

open it in earlier versions of Report Designer. This is the only way you can upgrade these versions of report definition files.

You can deploy an .rdl file created in an earlier version of SSRS to a SQL Server 2014 report server, and it is automatically upgraded on first use. The report server stores the report definition file in the original format. The report is automatically upgraded the first time it is viewed, but the stored report definition file remains unchanged.

For more information, see [Upgrade Reports](http://msdn.microsoft.com/en-us/library/ms143674(v=sql.120).aspx) (http://msdn.microsoft.com/en-us/library/ms143674(v=sql.120).aspx).

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 2005, you need to redeploy the extensions or assemblies for use with SSRS 2014, as explained in Section 14.7.3 “Deploying Custom Extensions and Assemblies” in Chapter 14 of the [SQL Server 2008 R2 Upgrade Technical Reference Guide](http://download.microsoft.com/download/3/0/D/30DB8D46-8ACF-442A-99A2-0F4CE74AE14D/SQL_Server_2008_R2_Upgrade_Technical_Reference_Guide.docx) (http://download.microsoft.com/download/3/0/D/30DB8D46-8ACF-442A-99A2-0F4CE74AE14D/SQL_Server_2008_R2_Upgrade_Technical_Reference_Guide.docx).

Verifying Configuration Files

In some cases, configuration changes made within SSRS 2005 (particularly those made manually within the configuration files) are not created automatically by the Reporting Services 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 2005. You should compare each SSRS 2005 configuration file that you saved as part of the pre-upgrade planning process to its new counterpart.

For more information, see Section 14.7.4 “Verifying Configuration Files” in Chapter 14 of the [SQL Server 2008 R2 Upgrade Technical Reference Guide](http://download.microsoft.com/download/3/0/D/30DB8D46-8ACF-442A-99A2-0F4CE74AE14D/SQL_Server_2008_R2_Upgrade_Technical_Reference_Guide.docx) (http://download.microsoft.com/download/3/0/D/30DB8D46-8ACF-442A-99A2-0F4CE74AE14D/SQL_Server_2008_R2_Upgrade_Technical_Reference_Guide.docx).

Uninstalling SSRS 2005/2008/2008 R2/2012

If you performed a side-by-side upgrade on a single server, you can use the new SSRS 2014 instance as configured alongside your SSRS 2005/2008/2008 R2/2012 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 2014 to use the names originally configured for SSRS 2005/2008/2008 R2/2012. 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 2014 for upgrade purposes.

For more information, see Section 14.7.5 “Uninstalling SSRS 2000, SSRS 2005, or SSRS 2008” in Chapter 14 of the [SQL Server 2008 R2 Upgrade Technical Reference Guide](http://download.microsoft.com/download/3/0/D/30DB8D46-8ACF-442A-99A2-0F4CE74AE14D/SQL_Server_2008_R2_Upgrade_Technical_Reference_Guide.docx) (http://download.microsoft.com/download/3/0/D/30DB8D46-8ACF-442A-99A2-0F4CE74AE14D/SQL_Server_2008_R2_Upgrade_Technical_Reference_Guide.docx).

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 backups of at least the following elements:

- Databases, applications, and configuration files
- The 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 2014.

Additional References

For an up-to-date collection of additional references for upgrading SQL Server 2014, see the following links:

- [SQL Server 2014 Web Site](http://www.microsoft.com/sqlserver/en/us/default.aspx)
(<http://www.microsoft.com/sqlserver/en/us/default.aspx>)
- [Books Online for SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms130214(v=sql.120).aspx)
([http://msdn.microsoft.com/en-us/library/ms130214\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/ms130214(v=sql.120).aspx))
- [SQL Server Developer Center](http://msdn.microsoft.com/en-us/sqlserver)
(<http://msdn.microsoft.com/en-us/sqlserver>)
- [SQL Server TechNet Resources](http://technet.microsoft.com/en-us/sqlserver)
(<http://technet.microsoft.com/en-us/sqlserver>)

Chapter 19: Data Mining

Introduction

Data mining is one of the most powerful analytical tools in the SQL Server Business Intelligence (BI) suite. Data mining was first introduced as part of SQL Server 2000 Analysis Services (SSAS 2000), the database platform's OLAP and BI component. Although it was SQL Server's first foray 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, SQL Server's data mining functionality reached from developers, database professionals, and advanced business analysts to end users.

The data mining success story continues in SSAS 2008 and SSAS 2008 R2. Using the foundation that SSAS 2005 laid, Microsoft enhanced data mining in SSAS 2008, adding new features and improving existing functionality. In SSAS 2008 R2, SSAS 2012, and SSAS 2014, Microsoft has focused on other parts of the BI suite, especially on the Business Intelligence Semantics Model (BISM), and therefore data mining does not differ from SSAS 2008. There are behavior and even breaking changes that you need to consider before upgrading SQL Server 2005/2008/2008 R2/2012 to SQL Server 2014. There is a new developer's tool, SQL Server Data Tools (SSDT), which replaces Business Intelligence Development Studio (BIDS). You can install SSAS 2014 in Tabular or Multidimensional mode; data mining is included in Multidimensional mode only. In addition to covering those changes, this chapter discusses the key steps 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 essential data mining upgrade resources, including the following:

- For details about data mining functionality in SSAS 2014, see [Data Mining \(SSAS\)](http://msdn.microsoft.com/en-us/library/bb510516(v=sql.120).aspx) ([http://msdn.microsoft.com/en-us/library/bb510516\(v=sql.120\).aspx](http://msdn.microsoft.com/en-us/library/bb510516(v=sql.120).aspx)) in SQL Server 2014 Books Online.
- For additional SQL Server data mining information, see the [Data Mining forum](http://social.msdn.microsoft.com/Forums/en-US/sqldatamining/threads) (<http://social.msdn.microsoft.com/Forums/en-US/sqldatamining/threads>).

Data Mining Features in the Various SQL Server Versions and Editions

Before you start upgrading, 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 versions:

- DC = Datacenter Edition, available in SQL Server 2008 R2
- EE = Enterprise Edition, available in SQL Server 2005/2008/2008 R2/, and 2012/2014
- BI = Business Intelligence Edition, available in SQL Server 2012/2014
- SE = Standard Edition, available in SQL Server 2005/2008/2008 R2/2012/2014
- WG = Workgroup Edition, available in SQL Server 2005/2008/2008 R2
- WE = Web Edition, available in SQL Server 2008/2008 R2/2012/2014
- SSE = SQL Server Express Edition and Express with Tools, available in SQL Server 2005/2008/2008 R2/2012/2014
- SSEA = SQL Server Express Edition with Advanced Services, available in SQL Server 2005/2008/2008 R2/2012/2014

In addition to the editions just mentioned, Microsoft offers the Developer Edition and the Enterprise Evaluation Edition. They have the same functionality as the Enterprise Edition, but the licensing is different.

Data mining features are supported on quite a granular level in SQL Server 2005/2008/2008 R2, as Table 1 shows. Cells with a light gray background show editions and features available in SQL Server 2008 and SQL Server 2008 R2 only.

Table 1: Data Mining Features in SQL Server 2005, SQL Server 2008, and SQL Server 2008 R2 Editions

Feature	DC	EE	SE	WG	WE	SSE	SSEA
Standard data mining algorithms	Yes	Yes	Yes	No	No	No	No
Data mining tools: wizards, editors, and query builders	Yes	Yes	Yes	No	No	No	No
Algorithm viewers	Yes	Yes	Yes	No	No	No	No
Enhanced integrated OLAP and data mining functionality (MDX prediction function and DM dimensions)	Yes	Yes	Yes	No	No	No	No

Feature	DC	EE	SE	WG	WE	SSE	SSEA
Reporting integration with DM prediction queries	Yes	Yes	Yes	No	No	No	No
Parallelism for model processing	Yes	Yes	No	No	No	No	No
Parallelism for model prediction	Yes	Yes	No	No	No	No	No
Text-Mining Term Extraction transformation (SSIS)	Yes	Yes	No	No	No	No	No
Text-Mining Term Lookup transformation (SSIS)	Yes	Yes	No	No	No	No	No
Data Mining Query transformation (SSIS)	Yes	Yes	No	No	No	No	No
Data Mining processing destination (SSIS)	Yes	Yes	No	No	No	No	No
Algorithm plug-in API	Yes	Yes	No	No	No	No	No
Advanced configuration and tuning options for data mining algorithms	Yes	Yes	No	No	No	No	No
Unlimited concurrent data mining queries	Yes	Yes	No	No	No	No	No
Unlimited attributes for association rules	Yes	Yes	No	No	No	No	No
Multiple prediction targets for Naïve Bayes, Neural Network, and Logistic Regression	Yes	Yes	No	No	No	No	No
Cross-validation	Yes	Yes	No	No	No	No	No
Models on filtered subsets of mining structure data	Yes	Yes	No	No	No	No	No
Time series: custom blending between ARTXP and ARIMA models	Yes	Yes	No	No	No	No	No
Time series: prediction with new data	Yes	Yes	No	No	No	No	No
Time series: cross-series prediction	Yes	Yes	No	No	No	No	No
Sequence prediction	Yes	Yes	No	No	No	No	No

As you can see, in SQL Server 2008 R2, many features are supported in the Enterprise and Datacenter Editions only. Beside those two editions, Standard is the only edition that supports data mining.

Microsoft simplified licensing for SQL Server 2012 and SQL Server 2014. There are fewer editions, with the three major editions being the Standard, Business Intelligence, and Enterprise Editions. Basic data mining starts with the Standard Edition, and all data mining features are available in the Business Intelligence and Enterprise Editions.

Table 2 shows details of data mining features supported by the different editions of SQL Server 2012 and SQL Server 2014.

Table 2: Data Mining Features in SQL Server 2012 and SQL Server 2014

Feature	EE	BI	SE	WE	SSE	SSEA
Standard data mining algorithms	Yes	Yes	Yes	No	No	No
Data mining tools: wizards, editors, and query builders	Yes	Yes	Yes	No	No	No
Parallelism for model processing	Yes	Yes	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	Yes	No	No	No	No
Advanced configuration and tuning options for data mining algorithms	Yes	Yes	No	No	No	No
Unlimited concurrent data mining queries	Yes	Yes	No	No	No	No
Unlimited attributes for association rules	Yes	Yes	No	No	No	No
Multiple prediction targets for Naïve Bayes, Neural Network, and Logistic Regression	Yes	Yes	No	No	No	No
Cross-validation	Yes	Yes	No	No	No	No
Models on filtered subsets of mining structure data	Yes	Yes	No	No	No	No
Time series: custom blending between ARTXP and ARIMA models	Yes	Yes	No	No	No	No
Time series: prediction with new data	Yes	Yes	No	No	No	No
Time series: cross-series prediction	Yes	Yes	No	No	No	No
Sequence prediction	Yes	Yes	No	No	No	No

Please note that if you are planning to downgrade an edition when migrating from SQL Server 2008 R2 to SQL Server 2014—for example, downgrading from the Enterprise Edition to the Business Intelligence Edition—you are going to lose some data mining functionality. Edition downgrading is not a supported in-place upgrade path, so you would have to migrate your mining models using other means, as we describe later in this chapter. Because SQL Server 2014 brings quite a few data mining enhancements compared to SQL Server 2005, rebuilding your 2005 forecasting data mining models is probably the best strategy. Additional validation of the 2005 predictive model is recommendable as well. Upgrading SQL Server 2008/2008 R2/2012 data mining models should be a straightforward process, as there are nearly no changes in version 2014. However, please check the discontinued and deprecated features, and the

breaking and behavior changes in SQL Server 2012 and SQL Server 2014 in order to prevent unpleasant surprises in your data mining applications.

Preparing to Upgrade

After you select the SQL Server 2014 edition that suits your needs, you need to investigate which features are deprecated in SQL Server 2012. These features will not affect your upgrade, but you will need to update your models to stop using them before your next upgrade. You also need to know what functionality cannot be upgraded because it is discontinued or because it has changed in SQL Server 2014. And you also need to be aware of some behavior changes between data mining in SQL Server 2005/2008/2008 R2 and SQL Server 2014; 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 a complete reference of SSAS changes in SQL Server 2014, see [Analysis Services Backward Compatibility](http://msdn.microsoft.com/en-us/library/ms143479.aspx) (<http://msdn.microsoft.com/en-us/library/ms143479.aspx>) in SQL Server 2014 Books Online.

Deprecated Features

There are no specific deprecated data mining features in SSAS 2014.

Discontinued Functionality

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 have been added. In contrast, SSAS 2014 supports standard PMML 2.1 and discontinues SSAS 2000 PMML extensions, meaning that you cannot use them. Note that 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 2014 from PMML and store it in an SSAS 2014 database. If you export it from SSAS 2014, standard PMML will be generated. Some of the most important SQL Server 2000 extensions to PMML 1.0 include:

- Support for nested tables.
- The Discretized, Ordered, and Cyclical model variables besides 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 with a continuous domain.

For a complete list of discontinued features in SSAS 2014, see [Discontinued Analysis Services Functionality in SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms143229.aspx) (<http://msdn.microsoft.com/en-us/library/ms143229.aspx>) in SQL Server 2014 Books Online.

There's only a short list of discontinued data mining functionality from SSAS 2005 to SSAS 2008/2008 R2/2012/2014:

- Mining Execution Location connection string property
- Mining Location connection string property

In SSAS 2014, the 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 2014 ignores the setting. If you need to upgrade your mining models from SQL Server 2000, please note that a direct upgrade path to 2014 is not supported anymore. You should upgrade to a later version of SQL Server 2005, 2008, 2008 R2, or 2012 first. Because a direct upgrade path is not supported, note that the following is discontinued as well:

- Migration Wizard, which is used to migrate SSAS 2000 databases to newer versions
- Decision Support Objects (DSO) library, which provides compatibility with SSAS databases

Breaking Changes

If you upgrade your data mining modes from SSAS 2005 to SSAS 2014, 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 SSAS 2014. If you are using ODBC data sources, you need to change them to OLE DB providers.
- DSOs are not installed by default when you install SQL Server 2014.
- 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 2014. In SSAS 2005, VBA functions return 0 or an empty string when NULL or empty values are used as arguments. In SQL Server 2014, VBA functions return NULL.

For information about some of these breaking changes, see [Breaking Changes to Analysis Services Features in SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms143742.aspx) (<http://msdn.microsoft.com/en-us/library/ms143742.aspx>) in SQL Server 2014 Books Online.

Behavior Changes

There are no specific behavior changes in the mining models when you upgrade them from SSAS 2005/2008/2008 R2/2012 to SSAS 2014. To confirm this information, see [Behavior Changes to Analysis Services Features in SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms143682.aspx) (<http://msdn.microsoft.com/en-us/library/ms143682.aspx>) in SQL Server 2014 Books Online.

Running Upgrade Advisor

It does not make much sense to run the Upgrade Advisor for checking the data mining models. The in-place upgrade from SQL Server 2005 with Service Pack 4 (SP4) is supported; the SQL Server 2014 Upgrade Advisor inspects SQL Server 2005 and later. However, there are nearly no changes in data mining from SQL Server 2008/2008 R2/2012 to SQL Server 2014, and only minor changes in data mining from SQL Server 2005. Therefore, the Upgrade Advisor typically does not find any issues with data mining models in SSAS 2005 and later.

However, this does not mean that you should just upgrade the 2005 models. You should also perform post-upgrade tasks. As we will show you later in this chapter, it makes sense to revise the 2005 predictive and forecasting models after upgrading to SSAS 2014.

Upgrading from SQL Server 2005

In Chapter 1, you learned how to start the SQL Server 2014 Setup program and perform an in-place and side-by-side upgrade. This section assumes that you have already installed SQL Server 2014, so we will not describe that process here. Instead, we will focus only on data mining issues you might face in your upgrade from SSAS 2005 to SSAS 2012. As you will see, the in-place upgrade is somewhat simpler, but the side-by-side upgrade strategy gives you more options for migrating mining models.

To demonstrate upgrading data mining models from SQL Server 2005 to SQL Server 2014, let's consider a sample SSAS 2005 database that has a data source from the SQL Server 2005 AdventureWorksDW demo database, a data source view with all 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 using demographic data and four 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 for bike models in different regions. The Association Rules algorithm model uses the vAssocSeqOrders and vAssocSeqLineItems database views to try to find out 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 1 shows the columns usage for Association Rules, and Figure 2 shows the columns usage for Sequence Clustering.

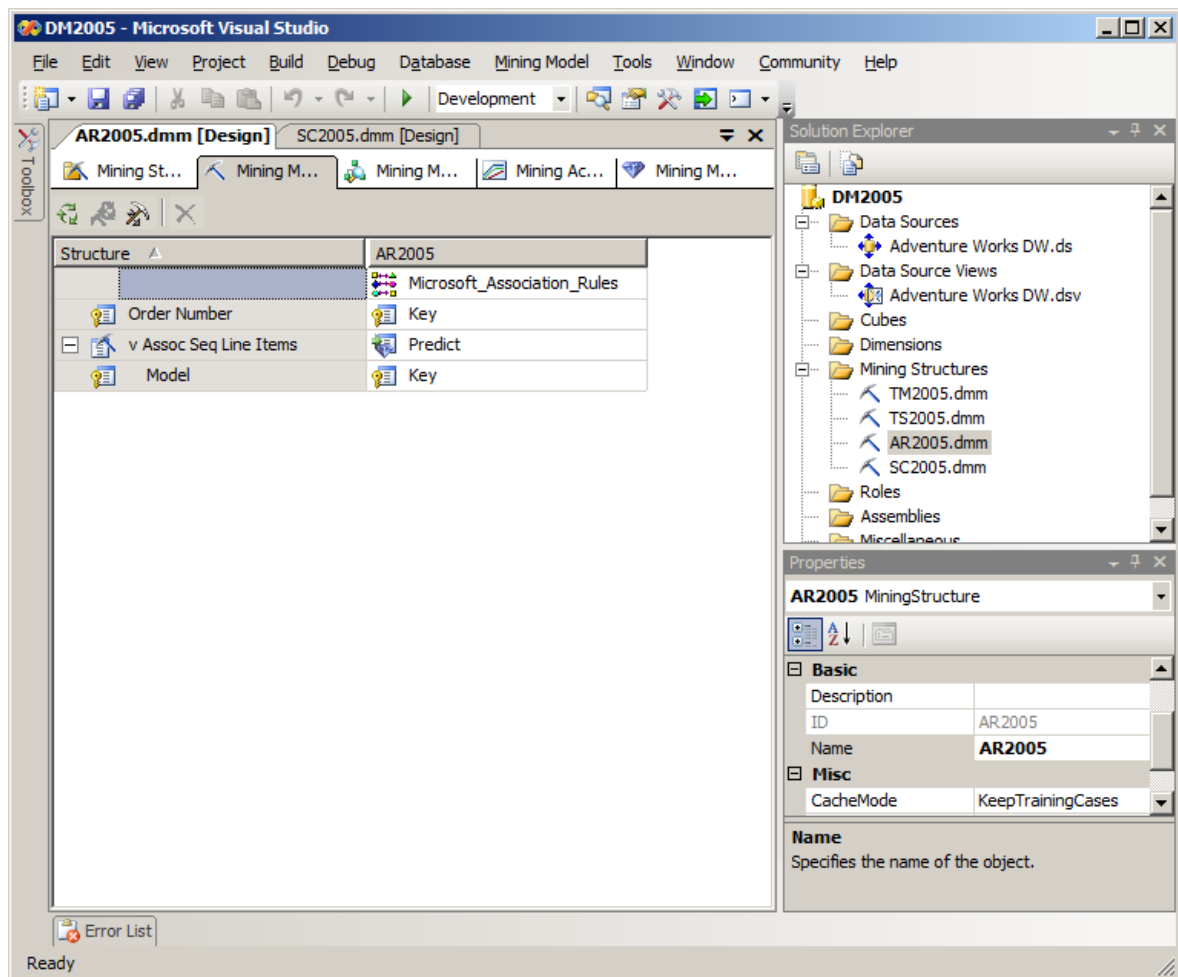


Figure 1: Columns usage for Association Rules

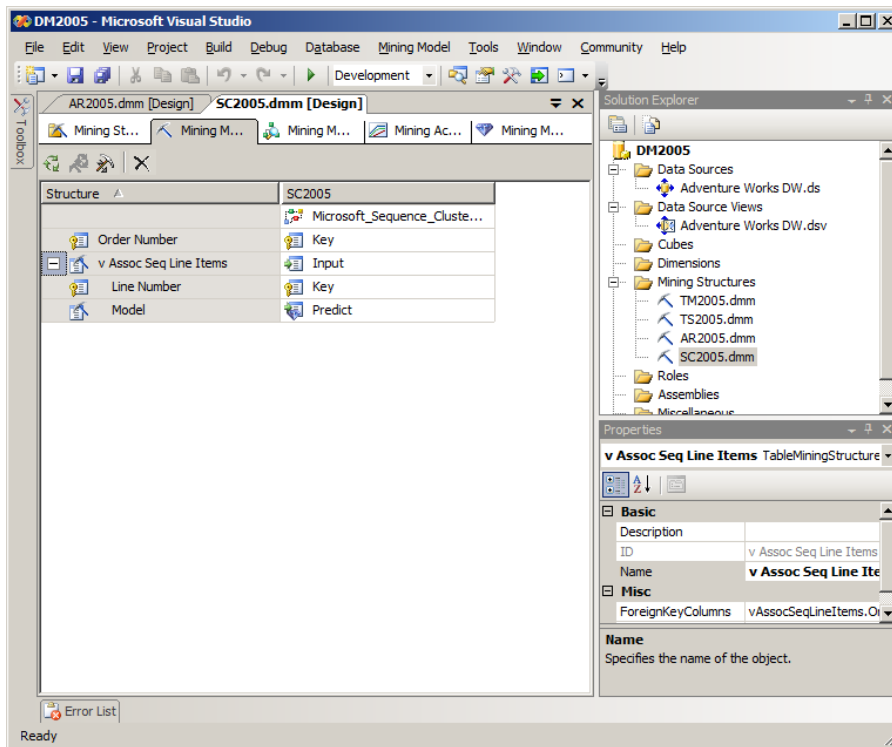


Figure 2: Columns usage for Sequence Clustering

All objects in the SSAS 2005 database that need to be upgraded for data mining models in our case include a data source, a data source view, and four mining structures with seven mining models, as shown in Figure 3.

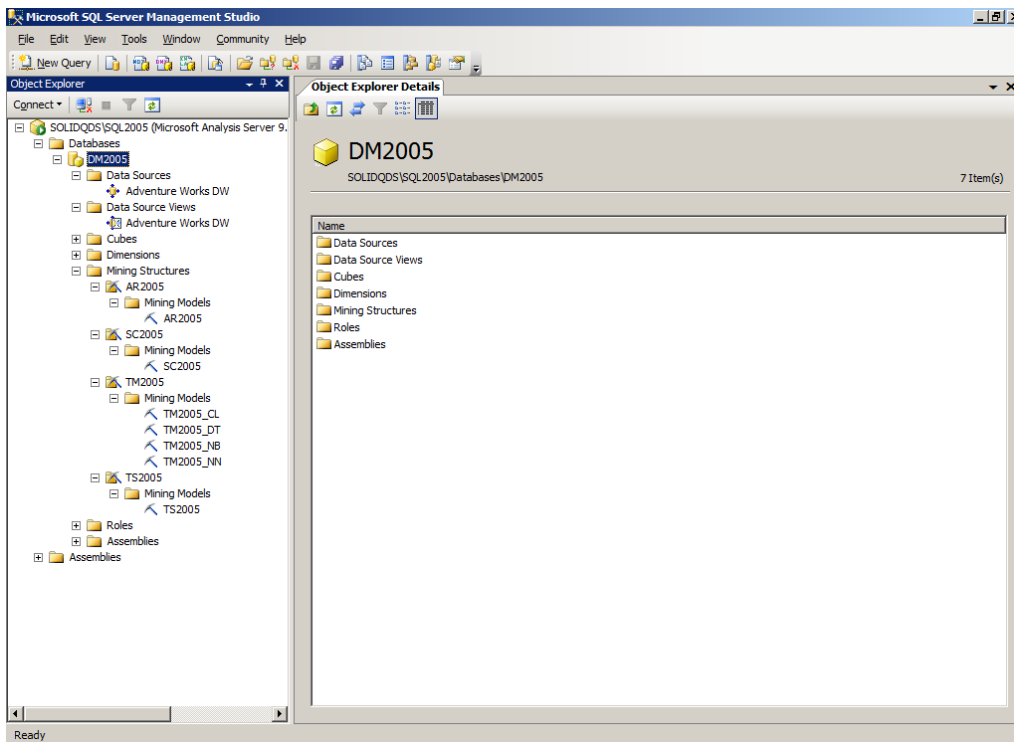


Figure 3: SSAS 2005 database with all objects that need to be upgraded

In-Place Upgrade

You start an in-place upgrade from SSAS 2005 to SSAS 2014 by running SQL Server Setup. The upgrade process is painless for the data mining models. Your SSAS databases are automatically upgraded, and you can continue using your mining models the same way you used them in SSAS 2005. In addition, you can use your SSAS 2005 mining projects in SSDT 2014 to continue with development. When you open the SSAS 2005 data mining project in SSDT 2014 for the first time, the conversion is launched automatically, and your project is converted to version 2014. Consider using a version control system to maintain previous versions, or back up the project manually before you convert it to SSDT 2014. Figure 4 shows how the conversion launches with a review of the project and the solution changes that are about to be made when you open a BIDS 2005 project.

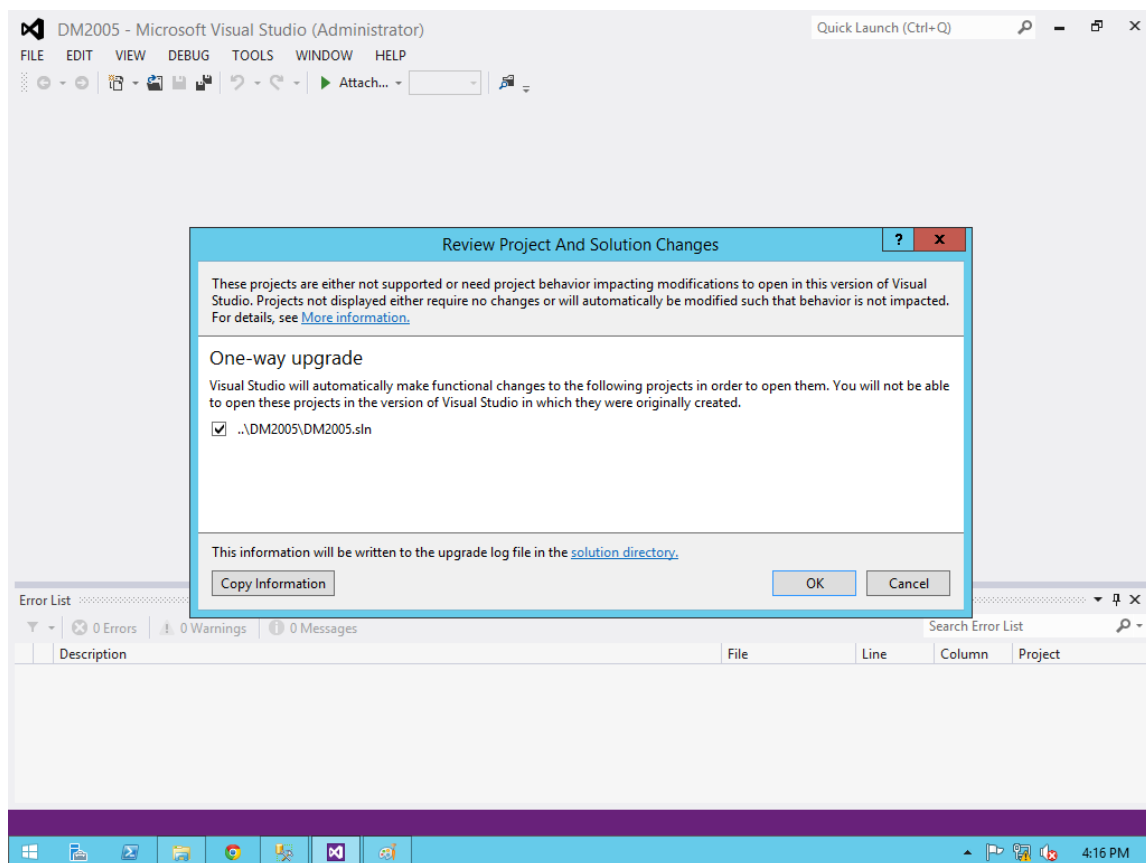


Figure 4: SSDT 2014's conversion launches when you open a BIDS 2005 project

If you want to perform an in-place upgrade, run SQL Server Setup and select the *Upgrade from SQL Server 2005, SQL Server 2008, SQL Server 2008 R2, or SQL Server 2012* link from the Installation tab of the SQL Server Installation Center, as shown in Figure 5.

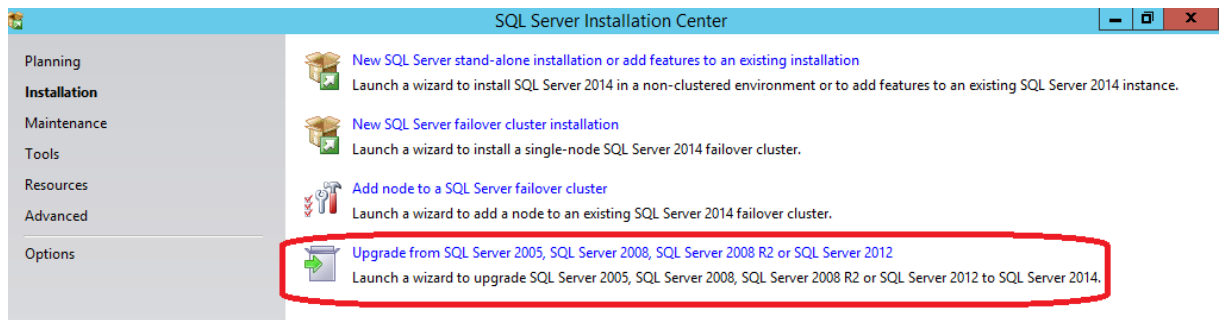


Figure 5: Select the Upgrade link for an in-place upgrade

Because the upgrading process was already described in previous chapters, we are not showing all the details here. Just be sure to select the correct SSAS 2005 instance. The upgraded instance will be in Multidimensional and Data Mining mode; there is no in-place upgrade to SSAS 2014 Tabular mode available from SQL Server Setup.

You will also need to perform some important data-mining tasks after your in-place upgrade from SSAS 2005 to SSAS 2014. There are many valuable features in SSAS 2014 data mining that can help you deploy a different predictive model in production, consolidate mining structures, or refine forecasting models. We will discuss these important considerations in the “Post-Upgrade Tasks” section later in this chapter.

Side-by-Side Upgrade

For a side-by-side upgrade, you have plenty of options for migrating your mining models from SSAS 2005 to SSAS 2014:

- You can back up the SSAS 2005 database and restore it on SSAS 2014.
- With SSMS, you can create an XMLA script for creating the complete database or any object in the database and then execute the script on SSAS 2014.
- You can open the SSAS 2005 project in SSDT 2014 and deploy it on SSAS 2014.
- You can reverse-engineer an SSAS 2005 database in SSDT 2014 to create a 2014 project and then deploy the project on SSAS 2014.

You can also quickly import SSAS 2005 data mining models to an SSAS 2014 database by using the EXPORT and IMPORT DMX commands. However, note that SSAS 2014 supports data mining only if it is installed in Multidimensional and Data Mining mode. SSAS 2014 in Tabular mode does not support data mining at all, and you cannot migrate your mining models to an SSAS 2014 Tabular instance.

With the EXPORT DMX command, you can export a complete mining structure, one or more mining models, or a model or structure with dependencies. Exporting with

dependencies means that all objects needed to process the structure, such as the data source and the data source view, 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 [TM2005]
    TO 'C:\Upgrade2014WP\TM2005_Structure.abf';
-- Exporting a single model
EXPORT MINING MODEL [TM2005_DT]
    TO 'C:\Upgrade2014WP\TM2005_Model.abf';
-- Exporting a model with dependencies
EXPORT MINING MODEL [AR2005]
    TO 'C:\Upgrade2014WP\AR2005_Model_Dependencies.abf'
WITH DEPENDENCIES;
```

In SSAS 2014, you can use SSMS to create an empty database. Note that you cannot create objects you need for processing the mining structures, namely data sources and data source views, from SSMS. You can use SSDT 2014 to create a SSAS project that includes only data sources and data source views, deploy it, and then import mining models and structures.

If you already have the destination SSAS 2014 database and you need to import only a mining structure, import it from the backup file with the complete structure, as follows:

```
-- Importing complete structure
IMPORT
    FROM 'C:\Upgrade2014WP\TM2005_Structure.abf';
```

Note that if you import from a file with only the mining model, the associated structure is created as well. Therefore, you cannot have a structure with 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:\Upgrade2014WP\TM2005_Model.abf';
```

This command imports from the file to which you exported the TM2005_DT model. And as we just noted, the TM2005 structure cannot exist in the destination database because it is re-created there during the import. If you executed the second import, the third one fails. If there is no TM2005 structure in the destination database, then the third import succeeds. After a successful third import, the TM2005 structure contains only one model, TM2005_DT.

After the import, you should try to process the complete database to check whether all dependent objects were imported correctly.

Post-Upgrade Tasks

After your upgrade from SSAS 2005 to SSAS 2014, you should check the data source connection string. Some providers, like SQL Server Native Client version 1.0 (SQLNCLI.1), are not supported, as Figure 6 shows.

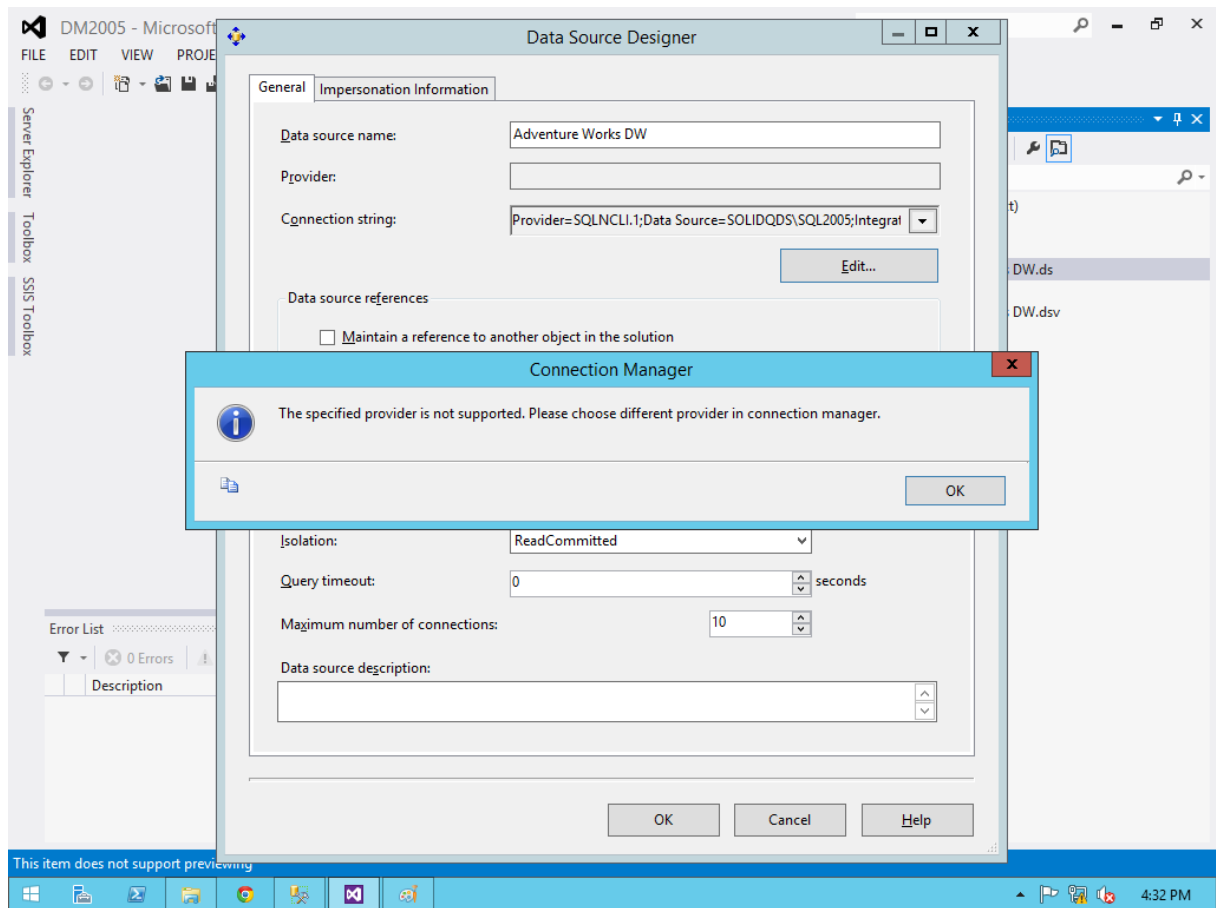


Figure 6: Some providers are not supported

You should change the provider to an appropriate supported one. Figure 7 shows how to change the provider to SQL Server Native Client 11.0.

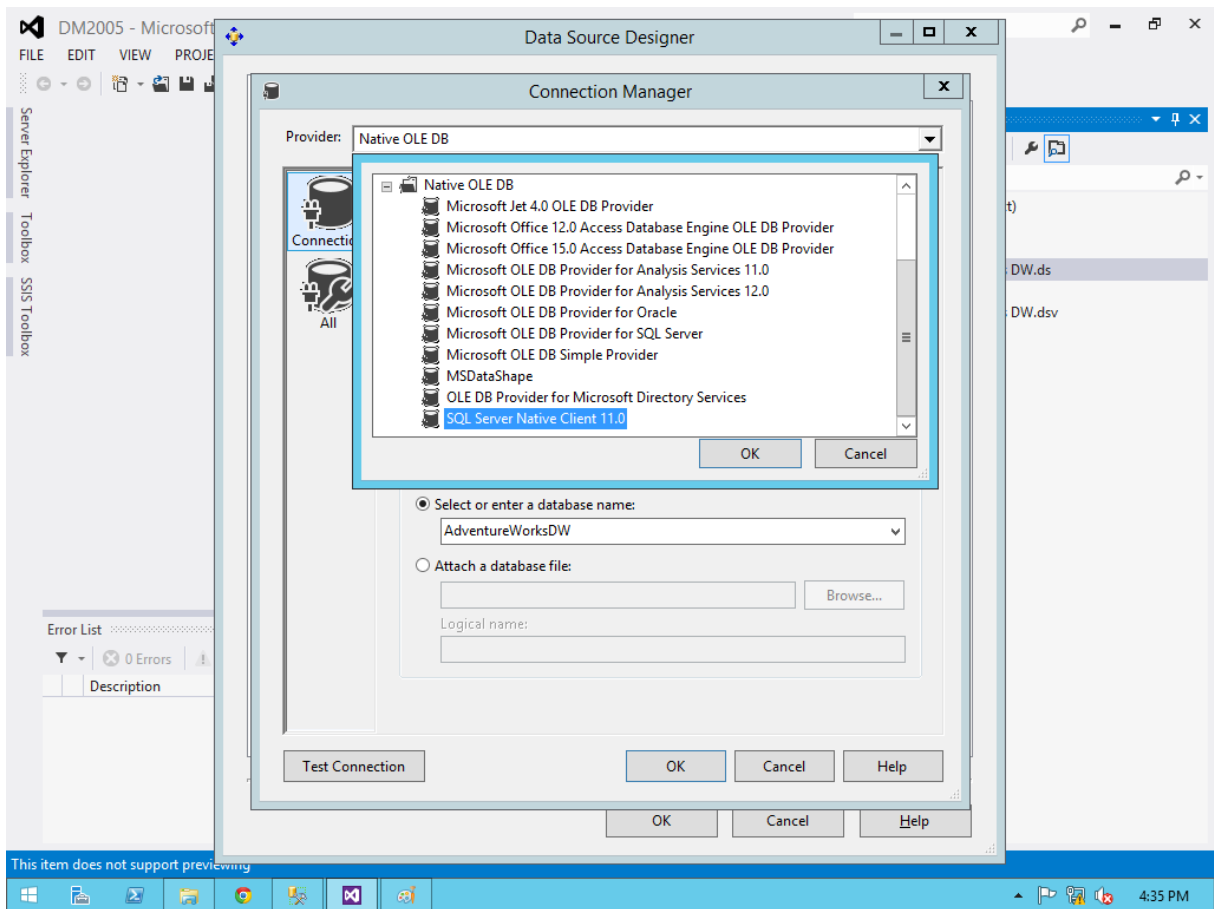


Figure 7: Changing the provider to a supported one

You should also check the accuracy and the robustness of your predictive models before deciding 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 need to 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 8 shows a Lift Chart for the predictive models we created in the sample SSAS 2005 database for the value 1 (buyers) of the predicted variable (Bike Buyer). From this chart, you can easily see the performance of the different models.

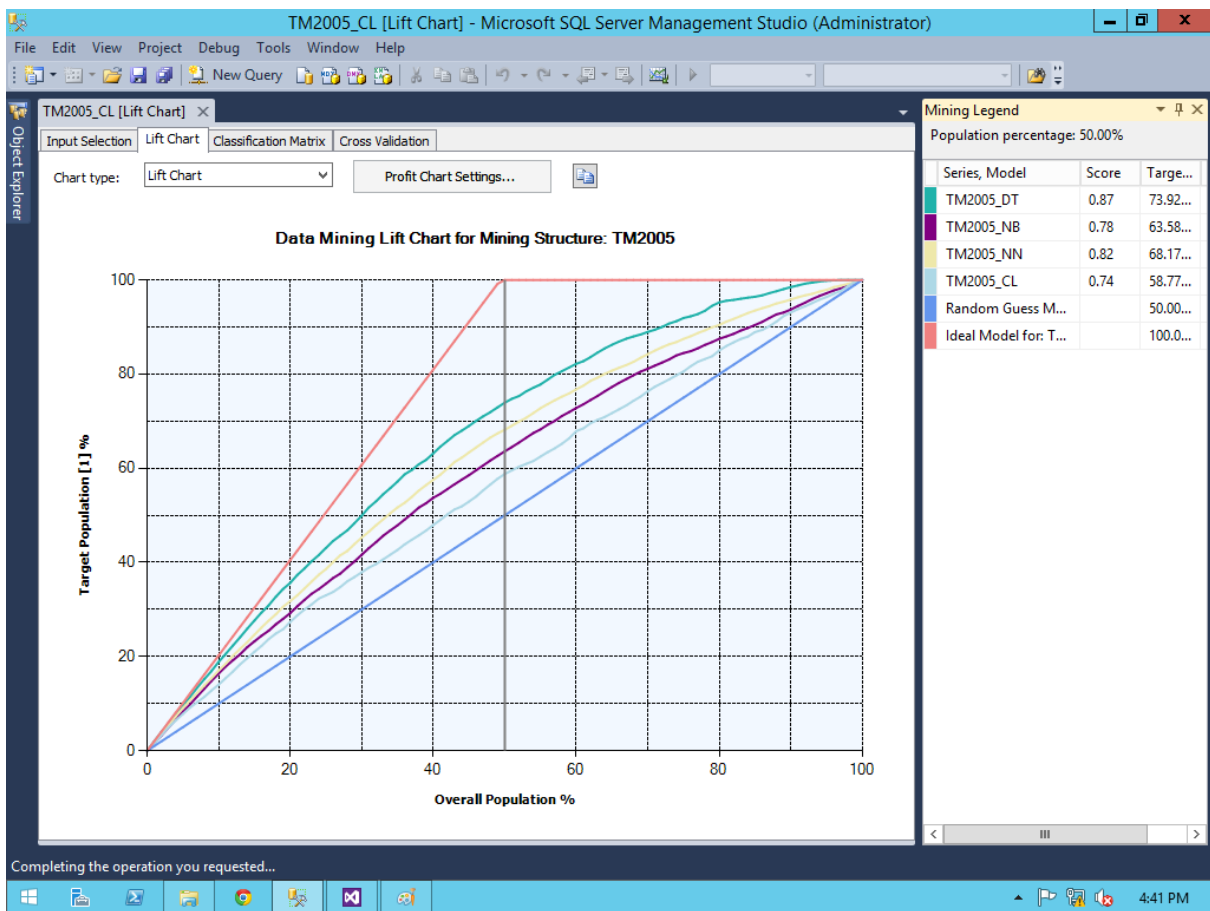


Figure 8: Lift Chart for predicting a single value

In the chart, notice that there is a total of six curves and lines. The four curves represent 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 Adventure Works customers buy bikes. If you could predict with 100 percent probability which customer is going to buy a bike and which is not, you would need to target 50 percent of the population only to get all bike buyers. The lower line is the Random Guess line. If you would pick out cases of the population randomly, you would need 100 percent of the cases for 100 percent of bike buyers. Likewise, 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, which is where the name Lift Chart comes from. 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 zero 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. In terms of accuracy of predictions from the demo data 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

A Lift Chart is useful, but it does not tell you 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 2014, you can test the reliability of predictive models by using cross-validation. With cross-validation, you partition your training data set 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 different accuracy measures across partitions. If the measures across various partitions differ a lot, the model is not robust on different training/test set combinations.

Figure 9 shows the cross-validation settings you can specify as well as the cross-validation results of predictive models.

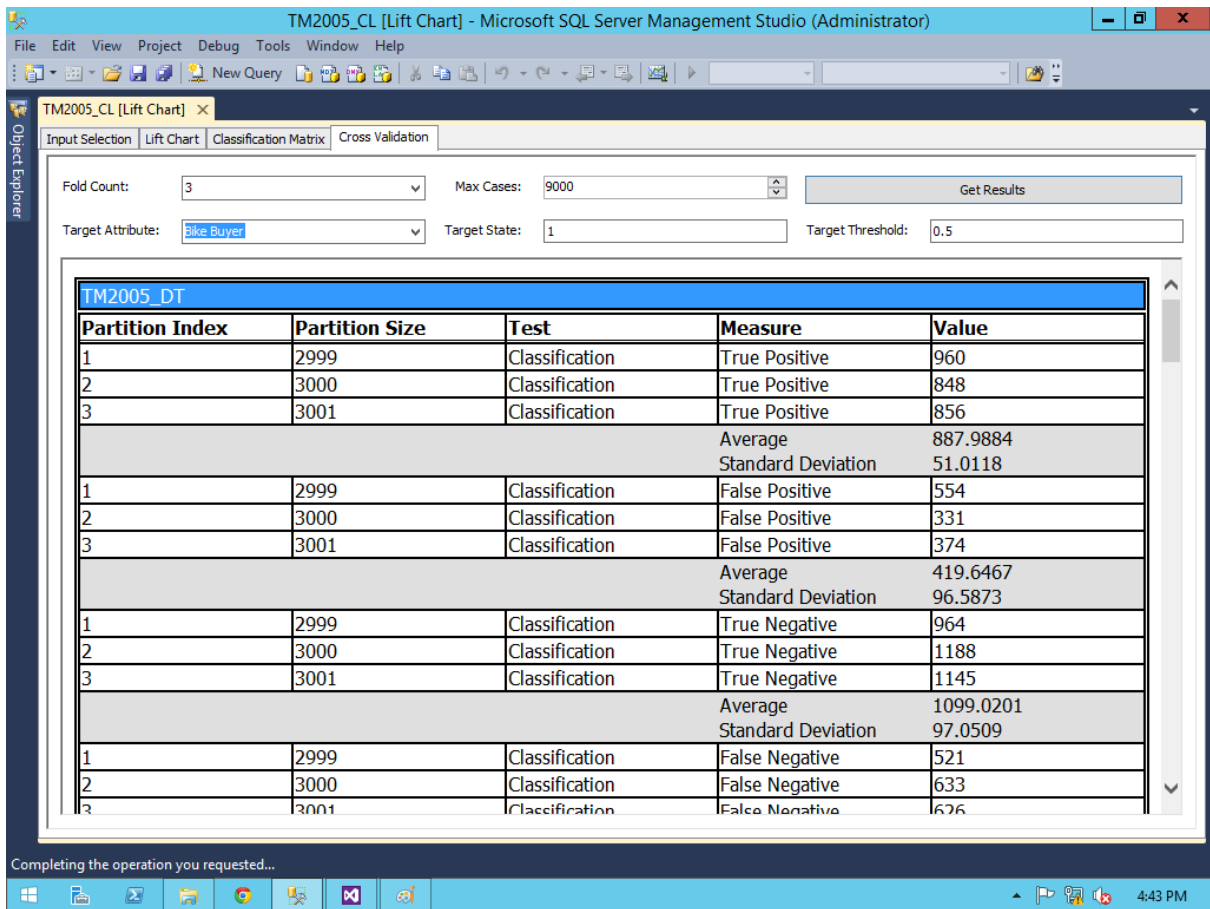


Figure 9: Cross-validation of predictive models—Decision Trees

You can define the following cross-validation settings:

- **Fold Count.** With this setting, you define how many partitions you want to create in your training data. In Figure 10, 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. 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 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 different 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 partitions where the predicted probability is greater than your accuracy threshold and the predicted state matches the target state.

You can see in Figure 9 that the True Positive classification of Decision Trees does not give you very consistent results across partitions. The first partition has approximately 13 percent more True Positive scores than the first two 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 quite high. However, when checking the Neural Network model, which Figure 10 shows, you can see that it is more consistent for the True Positive classification, which means that this model is more robust on different data sets than the Decision Trees model.

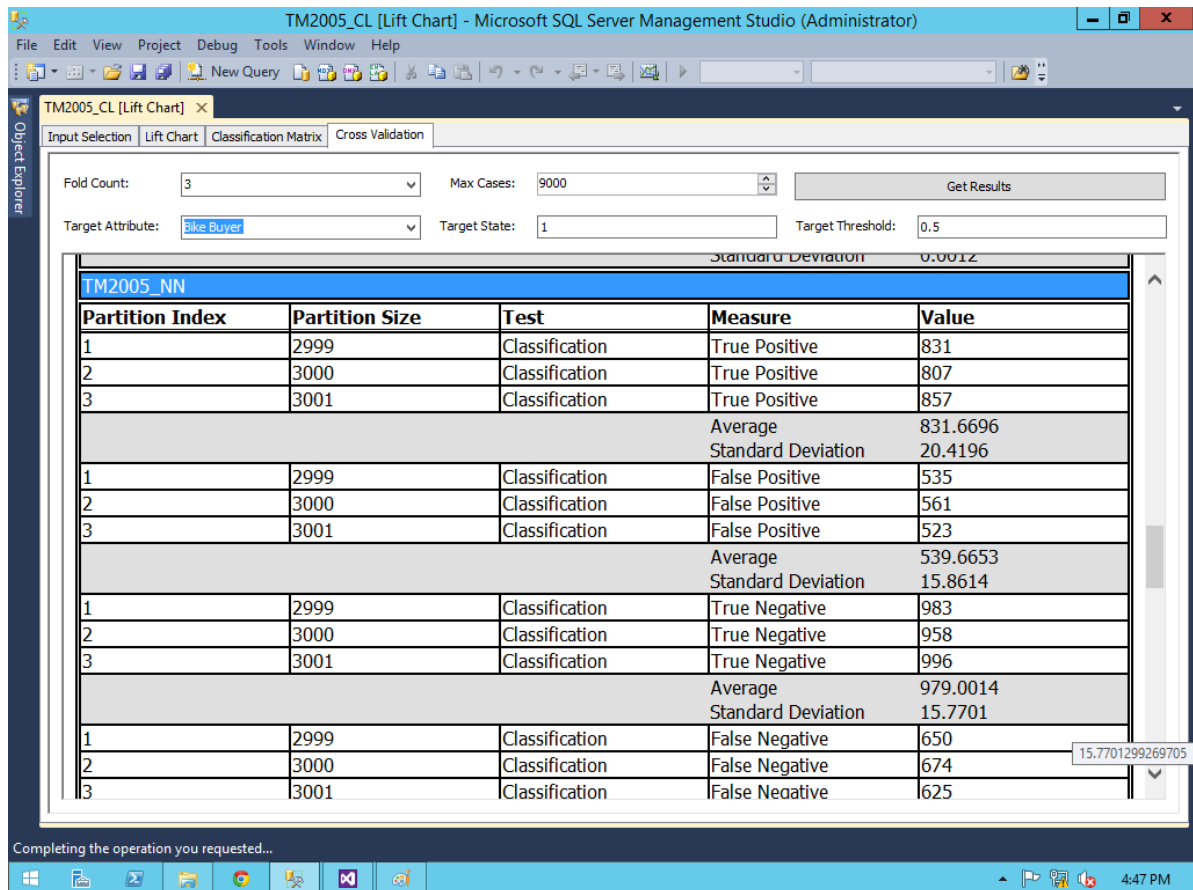


Figure 10: Cross-validation of predictive models—Neural Network

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 deciding 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 2012. To learn more about cross-validation, see [Cross-Validation \(Analysis Services – Data Mining\)](http://msdn.microsoft.com/en-us/library/bb895174.aspx) (<http://msdn.microsoft.com/en-us/library/bb895174.aspx>) in SQL Server 2014 Books Online.

Model Filtering

In SSAS 2005, you have to create a different mining structure if you want to use just a subset of data for an additional mining model. In SSAS 2008/2008 R2/2012/2014, you can filter a specific model to use only a subset of data for training. For example, using the same structure, you can create a model trained on the complete training set, another one trained only on the female population subset, and the third one trained only on the male population subset. You can then compare the performance of the models trained on the complete population with those trained on the various subsets. If you used different structures for subsets of training data in SSAS 2005, you should consider consolidating those structures into one structure in SSAS 2014 so that you can compare the performance of the models in a single Lift Chart or with a single cross-validation. To learn more about model filtering, see [Filters for Mining Models \(Analysis Services - Data Mining\)](http://msdn.microsoft.com/en-us/library/bb895167.aspx) (<http://msdn.microsoft.com/en-us/library/bb895167.aspx>) in SQL Server 2014 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, probability is good that it will perform well for forecasting future values.

You control the creation of historical models by using two algorithm parameters: HISTORICAL_MODEL_COUNT and HISTORICAL_MODEL_GAP. The first one controls the number of historical models that will be built, and the second one controls the number of time slices between historical models.

Figure 11 uses SSAS 2005 to show historical forecasts (the dotted lines before the current point in time) for the R-250 model for sales amount in Europe. What you can see is that the forecasts are very unstable and, thus, not very reliable. You can also see that the forecasts (the dotted lines after the current time point) become even negative after a future time point (about 20 points in the future in this example).

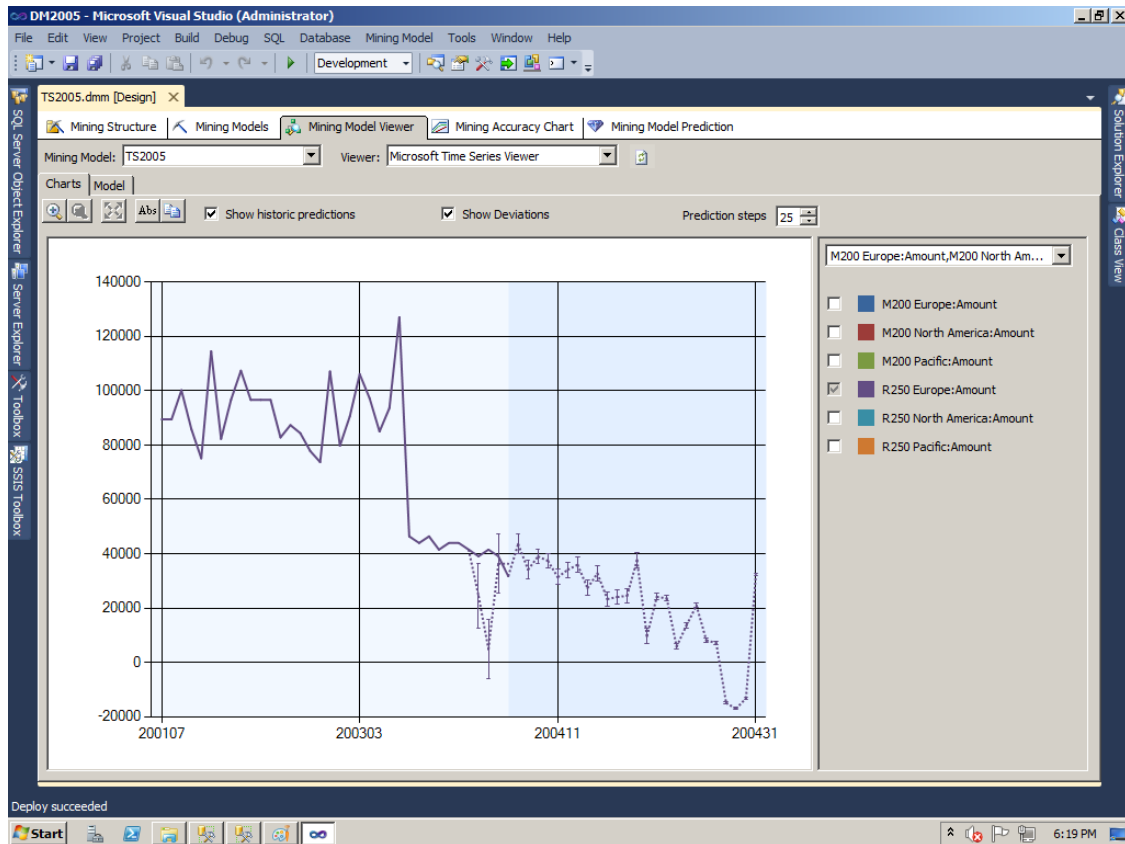


Figure 11: Historical and future forecasts in SSAS 2005

The reason for this instability is that SSAS 2005 Time Series use a single algorithm, Auto-Regression Trees with Cross-Prediction (ARTXP); this algorithm provides good short-term forecasts only. SSAS notes this instability in long-term forecasts and simply stops forecasting.

In SSAS 2008/2008 R2/2012/2014, you can use a blend of two different Time Series algorithms for forecasting. Besides ARTXP, SSAS 2014 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 2014, you should refine the blend of ARTXP 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. The second one (available only in Enterprise Edition) lets you define the blend manually.

As you can see in Figure 12, the upgraded version of the Time Series algorithm uses a MIXED forecast method (default). Therefore, ARTXP is used for short-term forecasts and ARIMA for long-term forecasts.

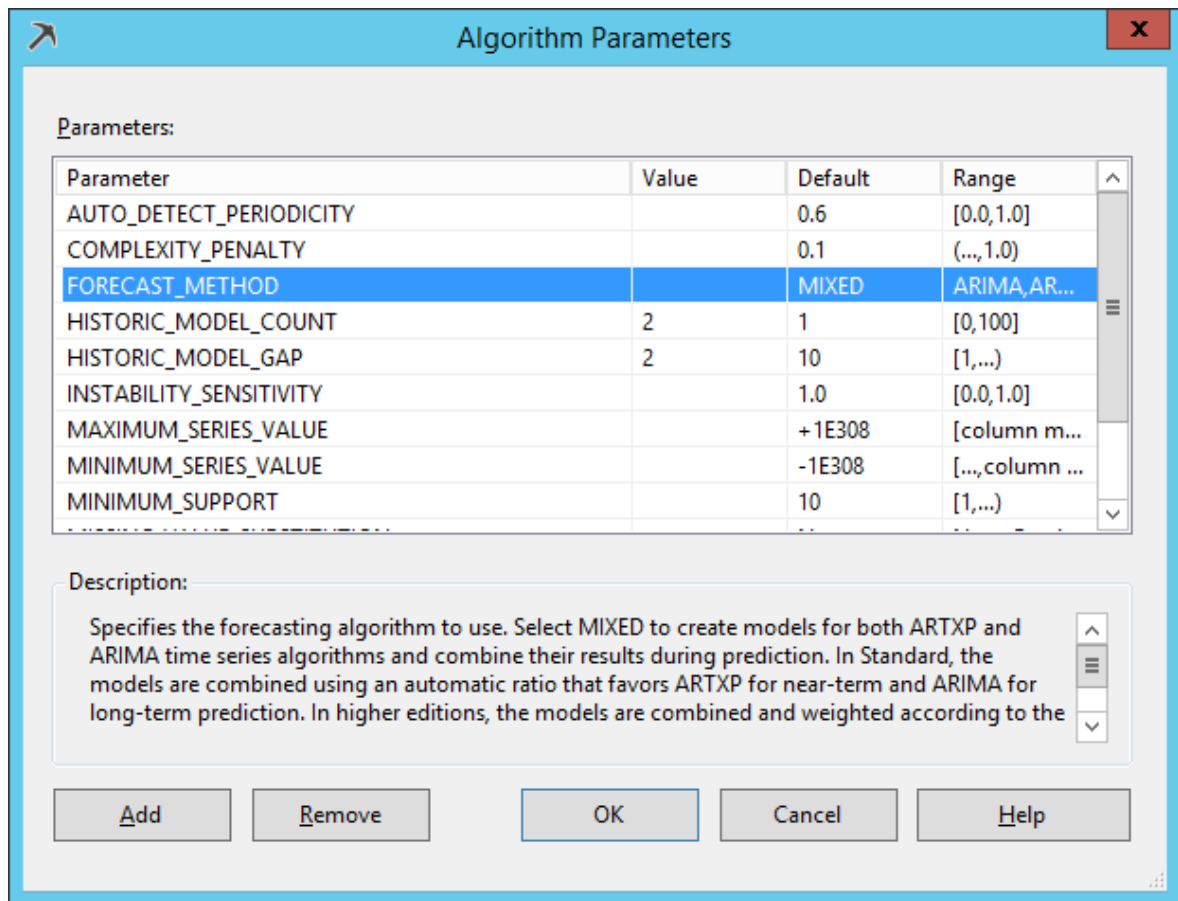


Figure 12: Time Series algorithm parameters in SSAS 2014

Figure 13 shows the forecast for the R-250 model for sales amount in Europe. As you can see, forecasts quickly stabilize and even long-term forecasts never achieve impossible values, such as values lower than zero. However, it appears that the historical forecasts are unstable. This is because we used only forecasts for two points in the past (the HISTORICAL_MODEL_GAP parameter), and thus only ARTXP method was used.

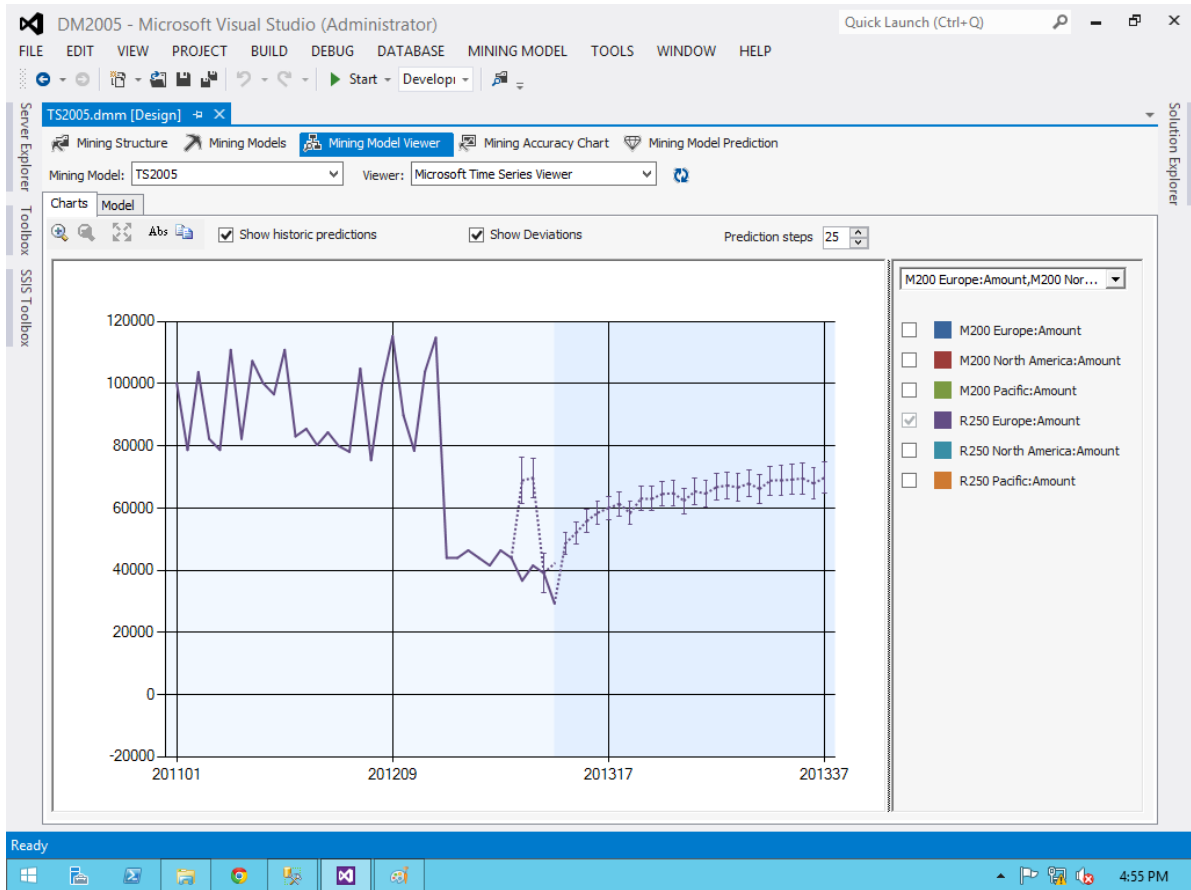


Figure 13: Historical and future forecasts in SSAS 2014

To learn more about Time Series algorithm parameters, see [Microsoft Time Series Algorithm](http://msdn.microsoft.com/en-us/library/ms174923.aspx) (<http://msdn.microsoft.com/en-us/library/ms174923.aspx>) in SQL Server 2014 Books Online.

Upgrading from SQL Server 2008, SQL Server 2008 R2, and SQL Server 2012

There is not much to say for upgrading data mining models from SQL Server 2008/2008 R2/2012 to SQL Server 2014 for two reasons. First, data mining is a mature feature in SQL Server. Second, there aren't any new features in data mining in SSAS 2014.

Migration from SSAS 2008/2008 R2/2012 to SSAS 2014 should succeed without problems and without the need for any additional post-upgrade tasks. This is valid for any kind of upgrade you use: in-place or side-by-side. In addition, you can use any of the following methods for a side-by-side upgrade:

- You can back up the SSAS 2008/2008 R2/2012 database and restore it on SSAS 2014.
- With SSMS, you can create an XMLA script for creating the complete database or any object in the database and then execute the script on SSAS 2014.
- You can open the SSAS 2008/2008 R2/2012 project in SSDT 2014 and deploy it on SSAS 2014.
- You can reverse-engineer an SSAS 2008/2008 R2/2012 database in SSDT 2014 to create a 2012 project and then deploy the project on SSAS 2014.

For example, Figure 14 shows the Backup Database window started from SSMS 2008 R2 in order to back up the SSAS 2008 R2 version of the database.

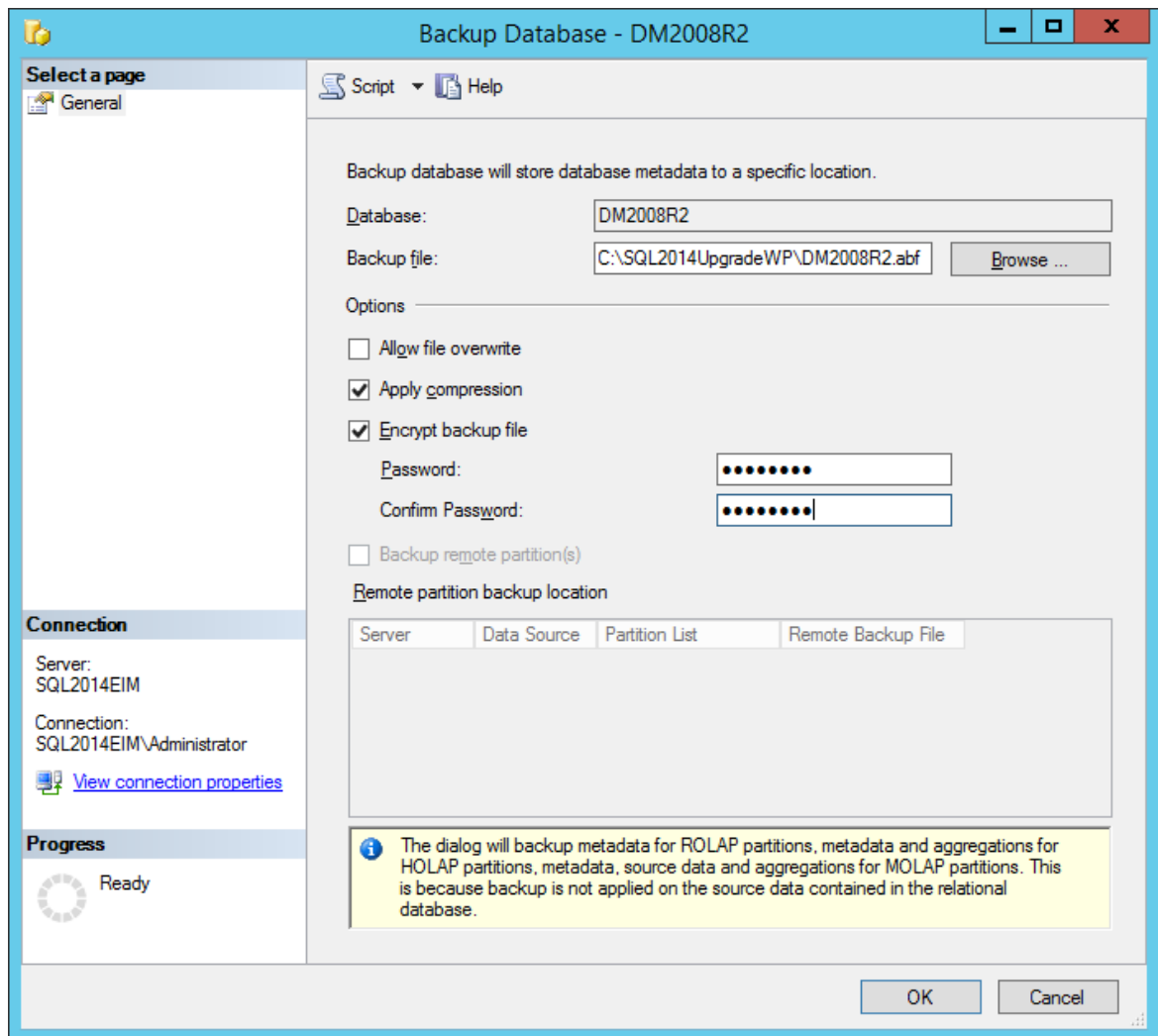


Figure 14: Backing up an SSAS 2008 R2 database

In Figure 15, you can see the restore window started in SSMS 2014.

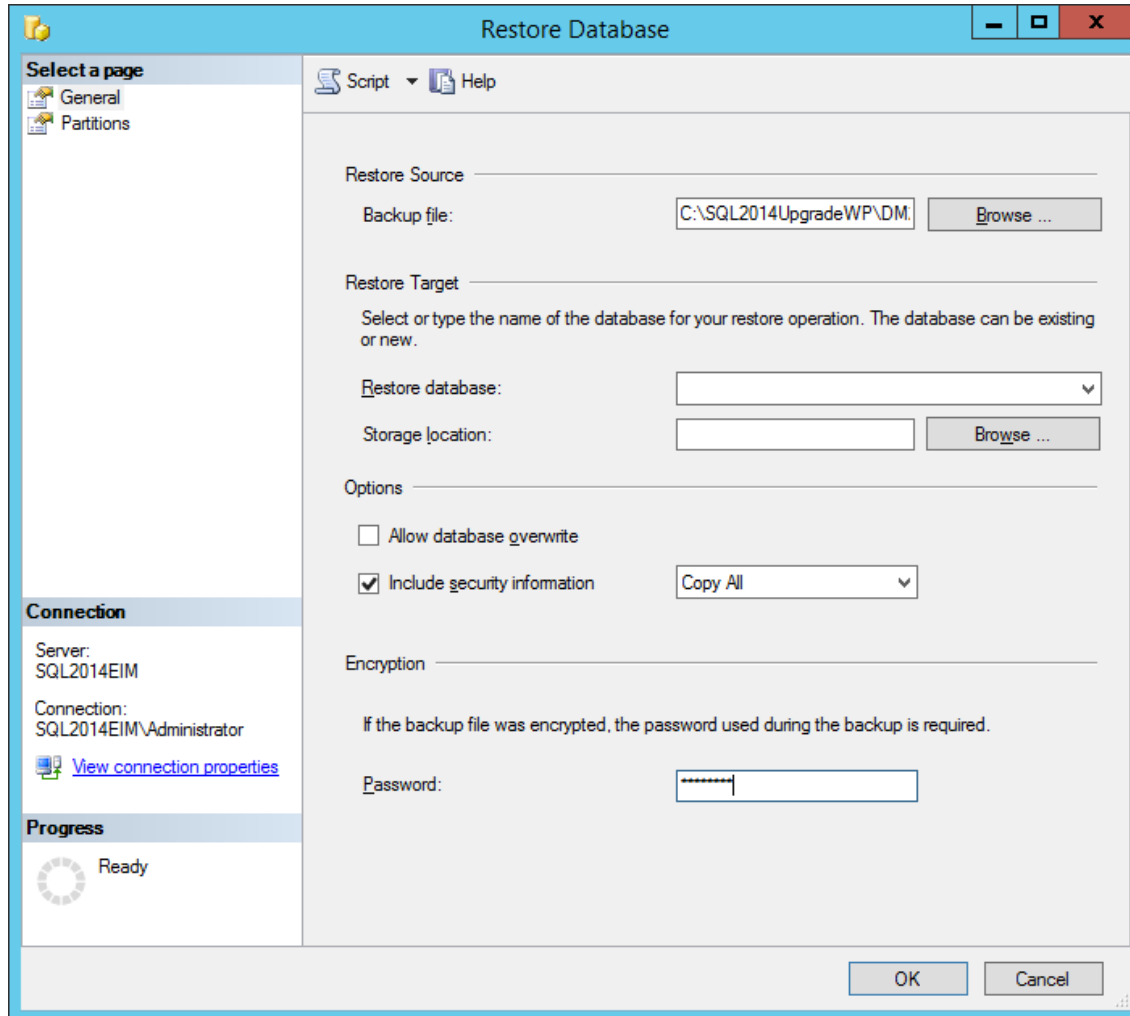


Figure 15: Restoring an SSAS 2008 R2 database on SSAS 2014

Conclusion

There are many good reasons to upgrade your data mining models to SQL Server 2014. 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 based on reliability. In addition, you can get much better long-term forecasting with the Time Series algorithm in SSAS 2014. Finally, you can consolidate multiple mining structures into one if you need to compare mining models trained on only a subset of the structure data. If you are upgrading from SSAS 2008/2008 R2/2012, you do not gain any new data mining features. However, you will probably want to consolidate all SSAS databases on a single version, so upgrading your data mining models makes sense.

For upgrading your data mining models, a side-by-side migration is preferred to an in-place upgrade. The most important reason is that with a side-by-side installation, you

leave your original models intact. However, if you do not have enough hardware power, you can perform an in-place upgrade. With thorough testing and planning, your upgrade can go smoothly whether your mining models are in SSAS 2005, SSAS 2008, SSAS 2008 R2, or SSAS 2012.

Additional References

- [SQL Server 2014 Web Site](http://www.microsoft.com/en-us/server-cloud/products/sql-server/)
(<http://www.microsoft.com/en-us/server-cloud/products/sql-server/>)
- [SQL Server TechNet Resources](http://technet.microsoft.com/en-us/sqlserver)
(<http://technet.microsoft.com/en-us/sqlserver>)
- [SQL Server Developer Center](http://msdn.microsoft.com/en-us/sqlserver)
(<http://msdn.microsoft.com/en-us/sqlserver>)

Chapter 20: Other Microsoft Applications and Platforms

Introduction

SQL Server serves as the data server at the back end of many of Microsoft's products. When you upgrade these products, you may find that the SQL Server version supporting the data layer within the product has also changed.

This chapter covers the following Microsoft products as they relate to upgrading to SQL Server 2014:

- Microsoft Lync Server 2013
- Microsoft Office SharePoint Server 2012
- Microsoft System Center
- Microsoft Dynamics

Microsoft Lync Server 2013

Microsoft Lync Server 2013 supersedes Microsoft Lync Server 2010. The current release of Lync Server 2013 supports SQL Server 2008 R2 and SQL Server 2012. See the following resources to learn more about Lync Server 2013 in general and as it relates to SQL Server:

- For information about Lync Server 2013, see the TechNet Library document collection at [Microsoft Lync Server 2013](http://technet.microsoft.com/en-us/library/gg398616.aspx) (http://technet.microsoft.com/en-us/library/gg398616.aspx).
- For information about Lync Server 2013 and SQL Server 2012, see [Configure SQL Server for Lync Server 2013](http://technet.microsoft.com/en-us/library/gg425848.aspx) (http://technet.microsoft.com/en-us/library/gg425848.aspx).
- For information about upgrading Lync Server 2010 to Lync Server 2013, check out [Migration from Lync Server 2010 to Lync Server 2013](http://technet.microsoft.com/en-us/library/jj205369.aspx) (http://technet.microsoft.com/en-us/library/jj205369.aspx).
- For details on SQL Server support in Lync Server 2013, see [Database software Support in Lync Server 2013](http://technet.microsoft.com/en-us/library/gg398990.aspx) (http://technet.microsoft.com/en-us/library/gg398990.aspx).

Microsoft Office SharePoint Server 2013

Microsoft Office SharePoint Server 2013 is one of the most widely used SQL Server applications on the Microsoft platform. Using SharePoint 2013 with SQL Server 2014 requires the use of the [May 2014 Cumulative Update](http://technet.microsoft.com/en-us/library/cc262485.aspx#section4) (http://technet.microsoft.com/en-us/library/cc262485.aspx#section4).

Note: Upgrading SQL Server beneath SharePoint is not supported in most cases. Instead, you should use a database migration approach.

For more information about how to optimize SQL Server 2014 with SharePoint Server 2013, see TechNet article [Hardware and Software requirements for SharePoint 2013](http://technet.microsoft.com/en-us/library/cc262485.aspx#section4) (http://technet.microsoft.com/en-us/library/cc262485.aspx#section4).

For an overview of SQL Server in the SharePoint environment, check out the TechNet article [Overview of SQL Server in a SharePoint environment SharePoint 2013](http://technet.microsoft.com/en-us/library/ff945791.aspx) (http://technet.microsoft.com/en-us/library/ff945791.aspx).

Microsoft System Center

The Microsoft System Center family of management products helps IT professionals manage their Windows Server infrastructure. The tools are especially useful in midsized to large data environments. For comprehensive information about System Center, see the [System Center](http://technet.microsoft.com/en-us/library/hh546785.aspx) web page (http://technet.microsoft.com/en-us/library/hh546785.aspx).

Two System Center products are especially relevant for SQL Server 2014: Systems Center 2012 Operations Manager (SCOM) and Data Protection Manager (DPM).

- The current SCOM Management Pack for SQL Server 2012 is compatible with SQL Server 2014.
- Data Protection Manager in System Center 2012 R2 supports SQL Server 2014.

Microsoft Dynamics

The Microsoft Dynamics products consist of a set of integrated financial, supply chain, and customer relationship management (CRM) solutions. The products include Dynamics AX, Dynamics CRM, Dynamics GP, Dynamics NAV, Dynamics SL, and Dynamics Retail Management System.

Each current Dynamics product supports SQL Server 2014 but has very specific requirements for Windows versions, SQL Server version, product service/feature packs, and so on.

You should upgrade a Dynamics application's database server to SQL Server 2014 only after the specific Dynamics Application has been officially supported on SQL Server 2014 or if you are following specific guidance from your Dynamics Technical Account Manager. If you are a registered Dynamics user, you can find this information at [Microsoft Dynamics Customers and Partners](https://mbs.microsoft.com/partnersource) (https://mbs.microsoft.com/partnersource).

Conclusion

As with any upgrade, planning is important for moving to any of these latest product versions.

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 [Upgrade to SQL Server 2014](http://msdn.microsoft.com/en-us/library/bb677622(v=sql.120).aspx) (http://msdn.microsoft.com/en-us/library/bb677622(v=sql.120).aspx) and [Windows Server 2012 R2 and Windows Server 2012](http://technet.microsoft.com/en-us/library/hh801901.aspx) (http://technet.microsoft.com/en-us/library/hh801901.aspx).

Also see the following resources:

- [SQL Server 2014 Web Site](http://www.microsoft.com/en-us/server-cloud/products/sql-server/)
(http://www.microsoft.com/en-us/server-cloud/products/sql-server/)
- [Books Online for SQL Server 2014](http://msdn.microsoft.com/en-us/library/ms130214(v=sql.120).aspx)
(http://msdn.microsoft.com/en-us/library/ms130214(v=sql.120).aspx)
- [SQL Server Developer Center](http://msdn.microsoft.com/en-us/sqlserver)
(http://msdn.microsoft.com/en-us/sqlserver)
- [SQL Server TechNet Resources](http://technet.microsoft.com/en-us/sqlserver)
(http://technet.microsoft.com/en-us/sqlserver)

Appendix 1: Version and Edition Upgrade Paths

Table 1 is taken from [Supported Version and Edition Upgrades](http://msdn.microsoft.com/en-us/library/ms143393.aspx) (<http://msdn.microsoft.com/en-us/library/ms143393.aspx>) in the Microsoft SQL Server 2014 product information on MSDN. You can upgrade from SQL Server 2005, SQL Server 2008, SQL Server 2008 R2, and SQL Server 2012.

Table 1: Supported Upgrade Scenarios from Earlier Versions of SQL Server to SQL Server 2014

Upgrade From	Supported Upgrade Path
SQL Server 2005 SP4 Enterprise	SQL Server 2014 Enterprise SQL Server 2014 Business Intelligence
SQL Server 2005 SP4 Developer	SQL Server 2014 Developer
SQL Server 2005 SP4 Standard	SQL Server 2014 Enterprise SQL Server 2014 Business Intelligence SQL Server 2014 Standard
SQL Server 2005 SP4 Workgroup	SQL Server 2014 Enterprise SQL Server 2014 Business Intelligence SQL Server 2014 Standard SQL Server 2014 Web
SQL Server 2005 SP4 Express, SQL Server 2005 SP4 Express with Tools, and SQL Server 2005 SP4 Express with Advanced Services	SQL Server 2014 Enterprise SQL Server 2014 Business Intelligence SQL Server 2014 Standard SQL Server 2014 Web SQL Server 2014 Express
SQL Server 2008 SP3 Enterprise	SQL Server 2014 Enterprise SQL Server 2014 Business Intelligence
SQL Server 2008 SP3 Developer	SQL Server 2014 Developer
SQL Server 2008 SP3 Standard	SQL Server 2014 Enterprise SQL Server 2014 Business Intelligence SQL Server 2014 Standard
SQL Server 2008 SP3 Small Business	SQL Server 2014 Standard
SQL Server 2008 SP3 Web	SQL Server 2014 Enterprise SQL Server 2014 Business Intelligence SQL Server 2014 Standard SQL Server 2014 Web
SQL Server 2008 SP3 Workgroup	SQL Server 2014 Enterprise SQL Server 2014 Business Intelligence SQL Server 2014 Standard SQL Server 2014 Web
SQL Server 2008 SP3 Express, SQL Server 2008 SP3 Express with Tools, and SQL Server 2008 SP3 Express with Advanced Services	SQL Server 2014 Enterprise SQL Server 2014 Business Intelligence SQL Server 2014 Standard SQL Server 2014 Web SQL Server 2014 Express

SQL Server 2008 R2 SP2 Datacenter	SQL Server 2014 Enterprise SQL Server 2014 Business Intelligence
SQL Server 2008 R2 SP2 Enterprise	SQL Server 2014 Enterprise SQL Server 2014 Business Intelligence
SQL Server 2008 R2 SP2 Developer	SQL Server 2014 Developer
SQL Server 2008 R2 SP2 Small Business	SQL Server 2014 Standard
SQL Server 2008 R2 SP2 Standard	SQL Server 2014 Enterprise SQL Server 2014 Business Intelligence SQL Server 2014 Standard
SQL Server 2008 R2 SP2 Web	SQL Server 2014 Enterprise SQL Server 2014 Business Intelligence SQL Server 2014 Standard SQL Server 2014 Web
SQL Server 2008 R2 SP2 Workgroup	SQL Server 2014 Enterprise SQL Server 2014 Business Intelligence SQL Server 2014 Standard SQL Server 2014 Web
SQL Server 2008 R2 SP2 Express, SQL Server 2008 R2 SP2 Express with Tools, and SQL Server 2008 R2 SP2 Express with Advanced Services	SQL Server 2014 Enterprise SQL Server 2014 Business Intelligence SQL Server 2014 Standard SQL Server 2014 Web SQL Server 2014 Express
SQL Server 2012 SP1 Enterprise	SQL Server 2014 Enterprise SQL Server 2014 Business Intelligence
SQL Server 2012 SP1 Developer	SQL Server 2014 Developer
SQL Server 2012 SP1 Standard	SQL Server 2014 Enterprise SQL Server 2014 Business Intelligence SQL Server 2014 Standard
SQL Server 2012 SP1 Web	SQL Server 2014 Enterprise SQL Server 2014 Business Intelligence SQL Server 2014 Standard SQL Server 2014 Web
SQL Server 2012 SP1 Express, SQL Server 2012 SP1 Express with Tools, SQL Server 2012 SP1 Express Management Studio, and SQL Server 2012 SP1 Express with Advanced Services	SQL Server 2014 Enterprise SQL Server 2014 Business Intelligence SQL Server 2014 Standard SQL Server 2014 Web SQL Server 2014 Express
SQL Server 2012 SP1 Business Intelligence	SQL Server 2014 Enterprise SQL Server 2014 Business Intelligence

Appendix 2: SQL Server 2014: Upgrade Planning Checklist

Decision	Factors	Notes
Preparing to Upgrade		
Decide to upgrade to SQL Server 2014	Identify the business reasons for upgrading to SQL Server 2014	
Choose SQL Server 2014 enhancements to implement	Select required and desired SQL Server 2014 features and enhancements for current and future development in the following categories: Relational Database Features <ul style="list-style-type: none"> • Database Engine • AlwaysOn • Security and auditing Business Intelligence Features <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • High level (mission critical) • Medium level • Low level Also classify according to whether instances are: <ul style="list-style-type: none"> • Default instance • Named instance • Virtual machine (VM) 	
Backward compatibility and upgrade tools	Gain familiarity with and select the appropriate upgrade tools for use in upgrade planning: <ul style="list-style-type: none"> • SQL Server 2014 Upgrade Advisor • Microsoft MAP Toolkit • Upgrade Assistant for SQL Server 2012 (UAFS) • Best Practices Analyzer for SQL Server 2012 • SQL Server Profiler 	

Decision	Factors	Notes
Ensure that servers meet SQL Server 2014 requirements	Ensure that servers meet minimum requirements for SQL Server 2014, including: <ul style="list-style-type: none"> • Sufficient processor, memory, and free disk space • Windows Server 2008 R2 SP1 or later • SQL Server 2005 SP4 or later (in-place upgrade) • SQL Server 2008 SP2 or later • SQL Server 2008 R2 SP1 or later • SQL Server 2012 SP1 or later 	
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 the requirements.	
Determine upgrade paths for editions	For each server, determine the target SQL Server 2014 edition: <ul style="list-style-type: none"> • Enterprise • Business Intelligence • Standard • 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: <ul style="list-style-type: none"> • Upgrade to support SQL Server 2014 • Changes to connectivity settings • Changes to authentication mode • 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: <ul style="list-style-type: none"> • SQL Server 2014 requires Windows Server 2008 R2 SP1 or later • Only SQL Server 2005 SP3 or later is supported on Windows Server 2008 R2 	
Select the appropriate upgrade strategy	For each server or database, choose the optimal upgrade strategy: <ul style="list-style-type: none"> • In-place upgrade • Side-by-side upgrade <ul style="list-style-type: none"> • Same physical server or VM • Separate physical server or VM 	

Decision	Factors	Notes
Developing an Upgrade Plan		
Upgrade as an IT project	Follow standard IT project practices for developing the upgrade project. Identify: <ul style="list-style-type: none"> • Team members • Stakeholders • Application/business owners Follow standard IT project procedures for the upgrade.	
Update DBA skills to SQL Server 2014	Ensure that DBA team members have SQL Server 2014 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 or enable .NET Framework 3.5 SP1.	
Establish performance baselines	Use tools such as 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.	

Decision	Factors	Notes
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: <ul style="list-style-type: none"> • Verify tests to ensure applications using the upgraded database servers will run as expected and required. • If available, enlist the support of the quality assurance (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: <ul style="list-style-type: none"> • Update statistics • Rebuild cubes • Reconfigure log shipping • Reconfigure 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 in future upgrade projects.	

Did this paper help you? Please give us your feedback. Tell us on a scale of 1 (poor) to 5 (excellent), how would you rate this paper and why have you given it this rating? For example:

- Are you rating it high due to having good examples, excellent screen shots, clear writing, or another reason?
- Are you rating it low due to poor examples, fuzzy screen shots, or unclear writing?

This feedback will help us improve the quality of white papers we release.

[Send feedback.](#)