Microsoft

Microsoft® SQL Server® 2012

William R. Stanek
Author and Series Editor

Pocket Consultant

PUBLISHED BY
Microsoft Press
A Division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

Library of Congress Control Number: 2011944612

ISBN: 978-0-7356-6376-3

Printed and bound in the United States of America.

Second Printing

Acquisitions Editor: Anne Hamiliton
Developmental Editor: Karen Szall
Project Editor: Karen Szall
Editorial Production: Christian Holdener, S4Carlisle Publishing Services
Technical Reviewer: boB Taylor
Copyeditor: Susan McClung
Indexer: Jean Skipp
Cover: Twist Creative · Seattle

[2012-04-13]

To my wife—for many years, through many books, many millions of words, and many thousands of pages, she's been there, providing support and encouragement and making every place we've lived a home.

*To my kids—for helping me see the world in new ways,
for having exceptional patience and boundless love,
and for making every day an adventure.*

To Karen, Martin, Lucinda, Juliana, Ben, and many others who've helped out in ways both large and small.

—William R. Stanek

Contents at a Glance

Contents

Introduction

Microsoft SQL Server 2012 Pocket Consultant is designed to be a concise and compulsively usable resource for Microsoft SQL Server 2012 administrators. It covers everything you need to know to perform the core administrative tasks for SQL Server and is the readable resource guide that you'll want on your desk at all times. Because the focus is on giving you maximum value in a pocket-sized guide, you don't have to wade through hundreds of pages of extraneous information to find what you're looking for. Instead, you'll find exactly what you need to get the job done.

This book is designed to be the one resource you turn to whenever you have questions about SQL Server administration. To this end, the book zeroes in on daily administration procedures, frequently used tasks, documented examples, and options that are representative while not necessarily inclusive. One of the key goals is to keep content concise enough that the book is compact and easy to navigate, while also ensuring that the book contains as much information as possible. Instead of a 1,000-page tome or a 100-page quick reference, you get a valuable resource guide that can help you quickly and easily perform common tasks, solve problems, and implement advanced SQL Server technologies such as replication, distributed queries, and multiserver administration.

Who Is This Book For?

Microsoft SQL Server 2012 Pocket Consultant covers the Standard, Business Intelligence, Enterprise, and Developer editions of SQL Server. The book is designed to be used in the daily administration of SQL Server and is written for:

- Current SQL Server database administrators
- Accomplished users who have some administrator responsibilities
- Administrators migrating to SQL Server 2012 from previous versions
- Administrators transitioning from other database architectures

To include as much information as possible, I had to assume that you have basic networking skills and a basic understanding of SQL Server. With this in mind, I don't devote entire chapters to understanding SQL Server architecture or running simple SQL queries. But I do cover SQL Server installation, configuration, enterprise-wide server management, performance tuning, optimization, maintenance, and much more.

I also assume that you're fairly familiar with SQL commands and stored procedures as well as the standard Windows user interface. If you need help learning SQL basics, you should read other resources (many of which are available from Microsoft Press).

How Is This Book Organized?

Speed and ease of reference are essential parts of this hands-on guide. The book has an expanded table of contents and an extensive index for finding answers to problems quickly. Many other quick reference features have been added to the book as well. These features include quick step-by-step procedures, lists, tables with fast facts, and cross-references.

The content is presented in four parts:

- Part I, "Microsoft SQL Server 2012 Essentials," discusses how to manage your SQL Server and SQL Server Services and clients.
- Part II, "Microsoft SQL Server 2012 Management and Security," dives into the details of implementing and configuring your SQL Server environment.
- Part III, "Microsoft SQL Server 2012 Data Management," focuses on data and the everyday tasks and best practices for managing your data.
- Part IV, "Microsoft SQL Server 2012 Optimization, Maintenance, and Recovery," addresses some of the more advanced topics that all administrators need to know.

What Is SQL Server 2012?

By functioning as a mission-critical data platform, allowing dynamic development, providing extensive business intelligence, and going beyond relational data, SQL Server 2012 provides the bedrock foundation on which small, medium, and large organizations can build their IT infrastructure. At the core of SQL Server 2012, you'll find the following:

- **Database Engine Services** Includes the core database, notification, and replication components. The core database—also known as the Database Engine—is the heart of SQL Server. Replication increases data availability by distributing data across multiple databases, allowing you to scale out the read workload across designated database servers.
- **Analysis Services** Delivers online analytical processing (OLAP) and data-mining functionality for business intelligence applications. Analysis Services enables your organization to aggregate data from multiple data sources, such as relational databases, and work with this data in a wide variety of ways.
- **Integration Services** Provides an enterprise data transformation and integration solution for extracting and transforming data from multiple data sources and moving it to one or more destination data sources. This functionality allows you to merge data from heterogeneous data sources, load data into data warehouses and data marts, and more.
- **Reporting Services** Includes Report Manager and Report Server, which provide a complete server-based platform for creating, managing, and distributing reports. Report Server is built on standard Microsoft Internet Information Services (IIS) and Microsoft .NET Framework technology, allowing you to combine the benefits of SQL Server and IIS to host and process reports.

- **Service Broker** Provides reliable queuing and messaging as a central part of the database. Queues can be used to stack work such as queries and other requests and perform the work as resources allow. Messaging allows database applications to communicate with each other. The Database Engine uses Service Broker to deliver notification messages.
- **Master Data Services** Provides a framework for creating business rules that ensure the quality and accuracy of your master data. Business rules can be used to start business processes that correct validation issues and handle workflows.
- **Data Quality Services** Provides a framework for creating a knowledge base repository of metadata that helps to improve the quality of your organization's data. Data cleansing processes can modify or remove data that is incomplete or incorrect. Data matching processes can identify and merge duplicates as appropriate.

System Requirements

Successful database server administration depends on three things: knowledgeable database administrators, strong database architecture, and appropriate hardware. The first two ingredients are covered: you're the administrator, you're smart enough to buy this book to help you through the rough spots, and you've implemented SQL Server 2012 to provide your high-performance database needs. This brings us to the issue of hardware. You should run SQL Server 2012 on a system with adequate memory, processing speed, and disk space. You also need an appropriate data and system protection plan at the hardware level.

Key guidelines for choosing hardware for SQL Server are as follows:

- **Memory** All editions of SQL Server 2012 except for Express require a minimum of 1 gigabyte (GB) of RAM. In most cases, you want to have at least 4 GB of RAM as a minimum starting point, even for development. The primary reason for having extra memory is performance. Additional database features—such as Analysis Services, Reporting Services, and Integration Services—increase the memory requirements. Also consider the number of user connections. Each user connection consumes about 24 KB. Data requests and other SQL Server processes use memory as well, and this memory usage is in addition to all other processes and applications running on the server.
- **Processor** The 64-bit versions run on the x64 family of processors from AMD and Intel, including AMD64 and Intel Extended Memory 64 Technology (Intel EM64T). Multicore Intel Xeon and AMD Opteron processors are recommended starting points. SQL Server 2012 supports symmetric multiprocessors and can process complex parallel queries. Parallel queries are valuable only when relatively few users are on a system and the system is processing large queries. On a dedicated system that runs only SQL Server and supports fewer than 100 simultaneous users who aren't running complex queries, a single multicore processor should suffice (although you should

always test with a representative workload). If the server supports more than 100 users or doesn't run on a dedicated system, you might consider adding processors (or using a system that can support additional processors as your needs grow). Keep in mind that the size of the queries and data sets being processed affects how well SQL Server scales. As the size of jobs being processed increases, you have increased memory and processor needs.

- **Disk drives** The amount of data storage capacity you need depends entirely on the number and size of the databases that the server supports. You need enough disk space to store all your data plus work space, indices, system files, virtual memory, and transaction logs. For log shipping and mirroring, you need space for the backup share and, in the case of a cluster, the quorum disk. I/O throughput is just as important as drive capacity. For the best I/O performance, Fibre Channel (FC) or Fibre Channel over Ethernet (FCoE) is the recommended choice for high-end storage solutions. Strongly consider solid state drives (SSDs) over spinning disks. Instead of using a single large drive, you should use several smaller drives, which allows you to configure fault tolerance with RAID. I recommend separating data and logs and placing them on separate spindles. This includes the backup share for log shipping and the quorum disk for clustering.
- **Data protection** You should add protection against unexpected drive failure by using RAID. For data, consider RAID 0 + 1 or RAID 5 as a starting point. For logs, consider RAID 1 as a starting point. RAID 0 (disk striping without parity) offers good read/write performance, but the effect of any failed drive is that SQL Server can't continue operation on an affected database until the drive is replaced and data is restored from backup. RAID 1 (disk mirroring) creates duplicate copies of data on separate drives, and you can rebuild the RAID unit to restore full operations. RAID 5 (disk striping with parity) offers good protection against single drive failure but has poor write performance. For best performance and fault tolerance, RAID 0 + 1 is recommended. This configuration consists of disk mirroring and disk striping without parity.
- **Uninterruptible power supply (UPS)** SQL Server is designed to maintain database integrity at all times and can recover information by using transaction logs. However, this does not protect the server hardware from sudden power loss or power spikes. Both of these events can seriously damage hardware. To prevent this, get a UPS that conditions the power. A UPS system gives you time to shut down the system properly in the event of a power outage, and it is also important in maintaining database integrity when the server uses write-back caching controllers.

If you follow these hardware guidelines, you will be well on your way to success with SQL Server 2012.

Conventions Used in This Book

I've used a variety of elements to help keep the text clear and easy to follow. You'll find code terms and listings in `monospace` type, except when I tell you to actually type a command. In that case, the command appears in **bold** type. When I introduce and define a new term, I put it in *italics*.

Other conventions include the following:

- **Best Practices** To examine the best technique to use when working with advanced configuration and administration concepts
- **Caution** To warn you about potential problems you should look out for
- **More Info** To provide more information on a subject
- **Note** To provide additional details on a particular point that needs emphasis
- **Real World** To provide real-world advice when discussing advanced topics
- **Security Alert** To point out important security issues
- **Tip** To offer helpful hints or additional information

I truly hope you find that *Microsoft SQL Server 2012 Pocket Consultant* provides everything you need to perform the essential administrative tasks for SQL Server as quickly and efficiently as possible. You are welcome to send your thoughts to me at *williamstanek@aol.com* or follow me at *www.twitter.com/WilliamStanek*. Thank you.

Other Resources

No single magic bullet for learning everything you'll ever need to know about SQL Server 2012 exists. While some books are offered as all-in-one guides, there's simply no way one book can do it all. With this in mind, I hope you use this book as it is intended to be used—as a concise and easy-to-use resource. It covers everything you need to perform core administration tasks for SQL Server, but it is by no means exhaustive.

Your current knowledge will largely determine your success with this or any other SQL Server resource or book. As you encounter new topics, take the time to practice what you've learned and read about. Seek out further information as necessary to get the practical hands-on know-how and knowledge you need.

I recommend that you regularly visit the SQL Server site (*www.microsoft.com/sqlserver/*) and Microsoft's support site (*www.support.microsoft.com*) to stay current with the latest changes. To help you get the most out of this book, you can visit my corresponding website at *www.williamstanek.com/sqlserver*. This site contains information about SQL Server 2012 and updates to the book.

Support and Feedback

This section provides useful information about accessing any errata for this title, reporting errors and finding support, as well as providing feedback and contacting Microsoft Press.

Errata

We’ve made every effort to ensure the accuracy of this book and its companion content. Any errors that have been reported since this book was published are listed on our Microsoft Press site:

http://www.microsoftpressstore.com/title/ 9780735663763

If you find an error that is not already listed, you can report it to us through the same page.

If you need additional support, email Microsoft Press Book Support at *mspinput@microsoft.com*.

Please note that product support for Microsoft software is not offered through the addresses above.

We Want to Hear from You

At Microsoft Press, your satisfaction is our top priority, and your feedback our most valuable asset. Please tell us what you think of this book at:

http://www.microsoft.com/learning/booksurvey

The survey is short, and we read *every one* of your comments and ideas. Thanks in advance for your input!

Stay in Touch

Let us keep the conversation going! We are on Twitter: *http://twitter.com/MicrosoftPress*

PART I

Microsoft SQL Server 2012 Essentials

CHAPTER 1

Managing Your SQL Servers

Microsoft SQL Server Management Studio is the primary tool you use to manage databases and servers. Other tools available to manage local and remote servers include SQL Server PowerShell, SQL Server Configuration Manager, Database Engine Tuning Advisor, and SQL Server Profiler. You use SQL Server Configuration Manager to manage SQL Server services, networking, and client configurations. Database Engine Tuning Advisor is available to help optimize indexes, indexed views, and partitions, and SQL Server Profiler lets you examine events generated by SQL Server, which can provide helpful details for troubleshooting. In this chapter, you will learn how to use SQL Server Management Studio. SQL Server Configuration Manager is discussed in Chapter 2, "Managing SQL Server Services and Clients." For details on tuning and tracing, see Chapter 12, "SQL Server 2012 Profiling and Monitoring."

Whenever you're working with databases and servers, keep in mind these concepts to help ensure your success:

- **Contained databases** These databases are fully or partially isolated databases that have no configuration dependencies on the instance of the SQL Server Database Engine where they are installed. A fully contained database does not allow any objects or functions that cross the boundary between the application model and the Database Engine instance. A partially contained database allows objects or functions that cross the boundary between the application model and the Database Engine instance. Contained database users with passwords are authenticated by the database. Authorized Microsoft Windows users and group members can connect directly to the database and do not need logins in the master database.

- **FileTable** Table structures act as virtual shares by storing FILESTREAM data and directory data as rows within tables. Even though the Database Engine manages the data at all times, a FileTable appears as a Windows share for non-transactional file access, allowing you to use MOVE, XCOPY, and other standard commands to load files when you are working with the command line or a batch script. The root of the hierarchy is established when you create the FileTable. A FileTable cannot be replicated or selected into like other tables.
- **Indirect checkpoints** Checkpoints are triggered based on the targeted recovery time you specify for a database, as opposed to automatic checkpoints, which are based on the maximum number of log records that can be processed in a particular recovery interval. A database that has a targeted recovery time does not use automatic checkpoints. Although indirect checkpoints can reduce read/write spikes by continually writing in the background, this continuous writing increases the total write load for the server instance, which may degrade performance for online transactional workloads.

You also should be aware of changes to the way the Database Engine works. While there are many discontinued and deprecated features, remember these important changes:

- Databases must be set to at least compatibility level 90. Level 90 is for Microsoft SQL Server 2005. Any earlier database is updated automatically when you install Microsoft SQL Server 2012.
- Indexes containing *varchar(max), nvarchar(max),* and *varbinary(max)* columns can now be rebuilt as an online operation.
- Re-create triggers that have WITH APPEND clauses, as these are no longer supported. Do the same for COMPUTE and COMPUTE BY, which must be rewritten by using the ROLLUP clause.
- Replace remote servers by using linked servers, and replace aliases with user accounts and database roles as appropriate.
- Replace the usage of SQL Mail with Database Mail and use ALTER DATABASE instead of sp_dboption.
- Use two-part table names following the syntax *schema.object* with ALTER TABLE, rather than four-part names, such as *server.database.schema.table*.

Using SQL Server Management Studio

The SQL Server Management Studio graphical point-and-click interface makes server, database, and resource management easy to perform. Using SQL Server Management Studio, you can manage local and remote server instances by establishing a connection to a SQL Server instance and then administering its resources. If you have disabled remote server connections to a particular server, you can work only with the server locally (by logging in to the system at the keyboard or by establishing a remote Terminal Server session in Windows and then running the local management tools).

Getting Started with SQL Server Management Studio

When you start working with SQL Server Management Studio, you see the Object Explorer view, shown in Figure 1-1. If this view is not displayed, you can access it (and other views) from the View menu. The following descriptions explain how to use each view:

- **Object Explorer** Allows you to view and connect to instances of SQL Server, Analysis Services, Integration Services, and Reporting Services. Once you have connected to a particular server, you can view its components as an object tree and expand nodes to work your way to lower levels of the tree.
- **Registered Servers** Shows the currently registered servers. Use Registered Servers to preserve login information for servers that you access frequently. The top bar of the view allows you to switch quickly between servers of a particular type (SQL Server, Analysis Server, Integration Server, or Report Server).
- **Template Explorer** Provides quick access to the default Query Editor templates, organized by action, and any custom templates you create. You can create templates in any script language supported by SQL Server Management Studio, SQL Server, and Analysis Server.
- **Solution Explorer** Provides quick access to existing SQL Server and Analysis Server projects. A project details the connections, queries, and other functions that are performed when the project is executed.

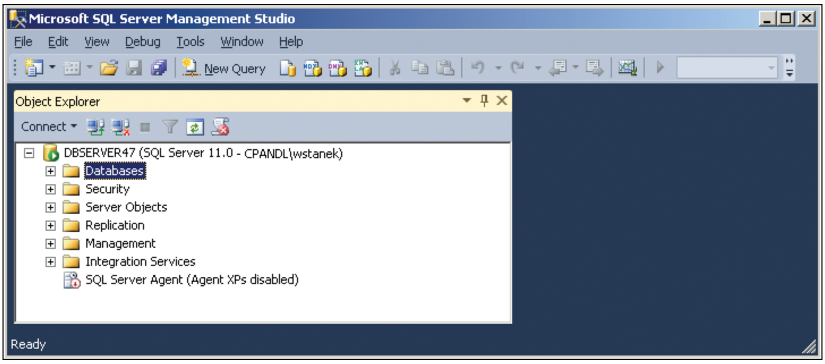

Microsoft SQL Server Management Studio
File Edit View Debug Tools Window Help
New Query
Object Explorer
Connect
DBSERVER47 (SQL Server 11.0 - CPANDL\wstanek)
Databases
Security
Server Objects
Replication
Management
Integration Services
SQL Server Agent (Agent XPs disabled)
Ready

FIGURE 1-1 Use SQL Server Management Studio to perform core administration tasks.

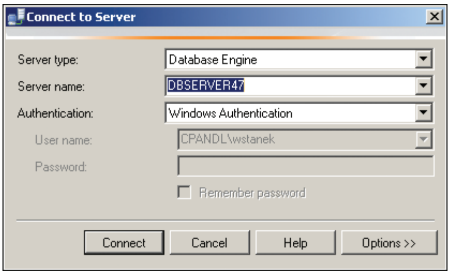
To run SQL Server Management Studio, click Start, type **ssms.exe** in the Search box, and then press Enter. Alternatively, select the related option on the Microsoft SQL Server 2012 menu. Next, you must connect to the server you want to work with. There are several ways to do this:

- Connect using a standard login to a server instance.
- Connect using a login to a specific database.
- Connect using server groups and registered servers.

Connecting to a server instance allows you to work with that particular server and its related components. (See Figure 1-2.) Typically, you want to connect to a server's Database Engine. The Database Engine gives you access to the following components and features:

- **Databases** Manage system databases, including the *master* and *model* databases, as well as user databases and database snapshots. If you've installed Reporting Services, you also can access the *ReportServer* and *Report ServerTempDB* databases under this node.
- **Security** Manage SQL Server logins, server roles, stored credentials, cryptographic providers, and auditing.
- **Server objects** Configure backup devices, HTTP endpoints, linked servers, and server triggers.
- **Replication** Configure distribution databases, update replication passwords, and launch Replication Monitor.
- **Management** View SQL Server logs, create, view, and manage maintenance plans, Microsoft Distributed Transaction Coordinator (MSDTC), and Database Mail. Configure data collection, Resource Governor, and Policy-Based Management policies.
- **SQL Server Agent** Configure SQL Server Agent jobs, alerts, operators, proxies, and error logs.

You store server and login information by using the Registered Servers feature. Registered servers can be organized using server groups and then can be accessed quickly in the Registered Servers view. Methods to manage server groups and register servers are discussed in the "Managing SQL Server Groups" and "Managing Servers" sections later in this chapter.


Connect to Server

Server type: Database Engine
Server name: DBSERVER47
Authentication: Windows Authentication
User name: CPANDL\wstanek
Password:
Remember password

FIGURE 1-2 Use the Database Engine to access core SQL Server components and features.

Connecting to a Specific Server Instance

To connect to a specific server instance by using a standard login, follow these steps:

1. Start SQL Server Management Studio. In the Connect To Server dialog box, use the Server Type list to select the database component you want to connect to, such as Database Engine. (If you exited the Connect To Server dialog box, you can display the Connect To Server dialog box by clicking File, Connect Object Explorer in SQL Server Management Studio.)
2. In the Server Name box, type the fully qualified domain name (FQDN) or host name of the server on which SQL Server is running, such as EngDBSrv12.cpandl.com or EngDBSrv12, or select Browse For More in the related drop-down list. In the Browse For Servers dialog box, select the Local Servers or Network Servers tab as appropriate. After the instance data has been retrieved, expand the nodes provided, select the server instance, and then click OK.

   TIP **The list in the Browse For Servers dialog box is populated by the SQL Server Browser service running on the database servers. There are several reasons that a SQL Server instance you want to work with might not be listed. The SQL Server Browser service might not be running on the computer running SQL Server. A firewall might be blocking User Datagram Protocol (UDP) port 1434, which is required for browsing. Or the HideInstance flag might be set on the SQL Server instance.**

3. Use the Authentication list to choose the option for authentication type, which is either Windows Authentication or SQL Server Authentication (based on the authentication types selected when you installed the server). Provide a SQL Server login ID and password as necessary.
   - **Windows Authentication** Uses your current domain account and password to establish the database connection. This authentication type works only if Windows authentication is enabled and you have appropriate privileges.
   - **SQL Server Authentication** Allows you to specify a SQL Server login ID and password. To save the password so that you do not have to reenter it each time you connect, select Remember Password.
4. Click Connect. Now you can use the Object Explorer view to work with this server.

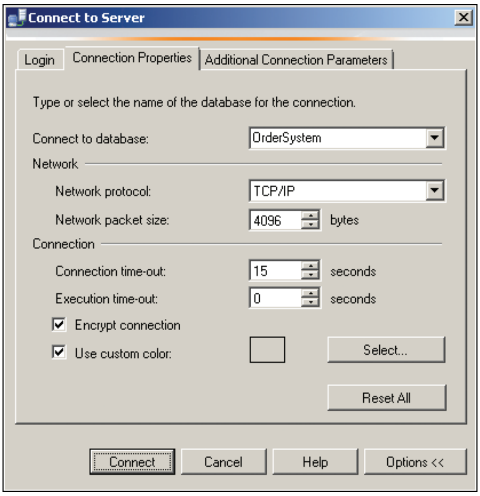
Connecting to a Specific Database

To connect to a specific database by using a standard login, follow these steps:

1. Start SQL Server Management Studio. In the Connect To Server dialog box, use the Server Type list to select the database component you want to connect to, such as Database Engine, and then, in the Server Name box, type the FQDN or host name of the server on which SQL Server is running,

such as EngDBSrv12.cpandl.com or EngDBSrv12. (If you exited the Connect To Server dialog box, you can display the Connect To Server dialog box by clicking File, Connect Object Explorer in SQL Server Management Studio.)

2. Use the Authentication list to choose the option for authentication type, which is either Windows Authentication or SQL Server Authentication (based on the authentication types selected when you installed the server). Provide a SQL Server login ID and password as necessary.
3. Click Options to display the advanced view of the Connect To Server dialog box. Select the Connection Properties tab, shown in Figure 1-3.



Type or select the name of the database for the connection.

Connect to database: OrderSystem

Network

Network protocol: TCP/IP

Network packet size: 4096 bytes

Connection

Connection time-out: 15 seconds

Execution time-out: 0 seconds

☑ Encrypt connection

☑ Use custom color:

FIGURE 1-3 Connect to a specific database.

4. In the Connect To Database box, type the name of the database you want to connect to, such as Personnel, or select Browse Server in the related drop-down list. When prompted, click Yes to establish a connection to the previously designated server. In the Browse Server For Database dialog box, select the database you want to use, and then click OK.
5. Using the Network Protocol list, select the network protocol and any other connection properties if you are prompted to do so. Shared Memory is the default network protocol for local connections. TCP/IP is the default for remote connections. If you want, establish a secure connection by selecting the Encrypt Connection check box.
6. Click Connect. You are now able to work with the specified database in the Object Explorer view.

Managing SQL Server Groups

You use SQL Server groups to organize sets of computers running SQL Server. You define these server groups, and you can organize them by function, department, or any other criteria. Creating a server group is easy. You can even create subgroups within a group, and if you make a mistake, you can delete a group as well.

> ***MORE INFO*** **Centrally managed servers also can be organized into server groups. For more information, see the "Configuring Central Management Servers" section in Chapter 3, "Implementing Policy-Based Management."**

Introducing SQL Server Groups and the Registered Servers View

In SQL Server Management Studio, you use the Registered Servers view to work with server groups. To use this view, or to display it if it is hidden, press Ctrl+Alt+G.

The top-level groups are already created for you, based on the SQL Server instances. Use the Registered Servers toolbar to switch between the various top-level groups. These groups are organized by SQL Server instance as follows:

- Database Engine
- Analysis Services
- Reporting Services
- SQL Server Compact Edition
- Integration Services

Although you can add registered servers directly to the top-level groups (as explained in the "Managing Servers" section later in this chapter), in a large enterprise with many SQL Server instances, you probably want to create additional levels in the server group hierarchy. These additional levels make it easier to access and work with your servers. You can use the following types of organizational models:

- **Division or business unit model** In this model, group names reflect the divisions or business units to which the computers running SQL Server belong or in which they are located. For example, you could have server groups such as Engineering, IS, Operations, and Support.
- **Geographic location model** In this model, group names reflect the geographic location of your servers, such as North America and Europe. You could have additional levels under North America for USA, Canada, and Mexico, for example, and levels under Europe could include UK, Germany, and Spain.

Figure 1-4 shows an example of using server groups. As the figure shows, subgroups are organized under their primary group. Under Database Engine, you might have Corporate Customers, Engineering, and Enterprise Data groups. Within Engineering, you might have Dev, Test, and Core subgroups.

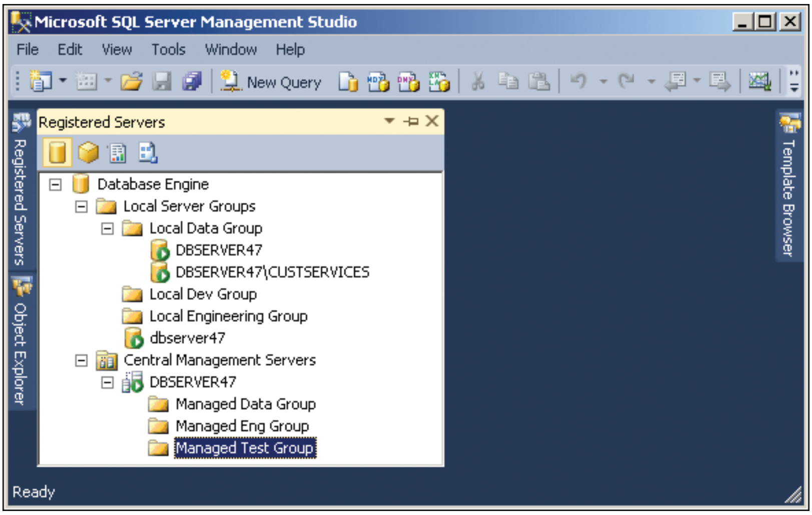
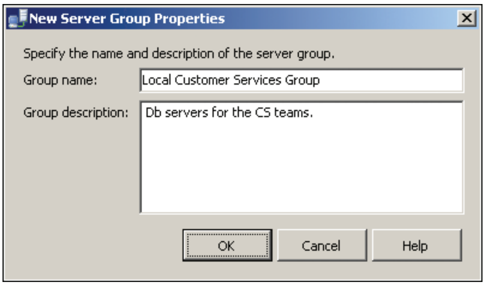


FIGURE 1-4 Use server groups to organize SQL Server deployments.

Creating a Server Group

You can create a server group or a subgroup by completing the following steps:

1. In SQL Server Management Studio, display the Registered Servers view by pressing Ctrl+Alt+G. If the view was previously hidden, this step also displays the view.
2. Use the Registered Servers toolbar to select the top-level group. For example, if you want to create a second-level or third-level group for Database Engine instances, select Database Engine.
3. As necessary, expand the top-level group node and the Local Server Groups nodes by double-clicking each in turn. You will see the names of the top-level server group and any second-level server groups that you created. You can now do the following:
   - Add a server group to one of the top-level or second-level groups by right-clicking the group name and choosing New Server Group.
   - Add a server group to a lower-level group by expanding the server group entries until the group you want to use is displayed. Right-click the group name, and then choose New Server Group.
4. In the New Server Group Properties dialog box, shown in Figure 1-5, type a name and description for the new group in the boxes provided. Click OK.



Specify the name and description of the server group.

Group name: Local Customer Services Group

Group description: Db servers for the CS teams.

FIGURE 1-5 Enter a name and description in the New Server Group Properties dialog box.

Deleting a Server Group

You can delete a group or subgroup by completing the following steps:

1. In SQL Server Management Studio, display the Registered Servers view by pressing Ctrl+Alt+G. If the view was previously hidden, this step also displays the view.
2. Use the Registered Servers toolbar to select the top-level group in which the group you want to delete is located. For example, if you want to delete a second- or third-level group for Database Engine instances, select Database Engine.
3. Click the plus sign (+) next to the group or subgroup you want to delete. If the group has servers registered in it, move them to a different group. (The steps involved in moving servers to a new group are explained in the "Moving a Server to a New Group" section later in this chapter.)
4. Select the group or subgroup entry.
5. Press Delete. When prompted to confirm the action, click Yes.

Editing and Moving Server Groups

Server groups have several key properties that you can edit: the name, the description, and the location in the Registered Server hierarchy. To edit a group's name or description, follow these steps:

1. Right-click the group in the Registered Servers view, and then select Properties.
2. In the Edit Server Group Properties dialog box, enter the new group name and description. Click OK.

To move a group (and all its associated subgroups and servers) to a new level in the server group hierarchy, follow these steps:

1. Right-click the group in the Registered Servers view, point to Tasks, and then select Move To.

2. In the Move Server Registration dialog box, you can now do the following:
   - Move the group to the top-level group by selecting the top-level group. This makes the group a second-level group.
   - Move the group to a different level by selecting a subgroup into which you want to place the group.
3. Click OK.

Adding SQL Servers to a Group

When you register a computer running SQL Server for use with SQL Server Management Studio, you can choose the group in which you want to place the server. You can even create a new group specifically for the server. The next section covers the topic of server registration.

Managing Servers

Servers and databases are the primary resources you manage in SQL Server Management Studio. When you select a top-level group in the Registered Servers view, you can see the available server groups. If you expand the view of these groups by double-clicking the group name, you can see the subgroups or servers assigned to a particular group. Local servers are registered automatically (in most cases). If a local server is not shown, you need to update the local registration information. If the remote server you want to manage is not shown, you need to register it.

Registration saves the current connection information and assigns the server to a group for easy future access using the Registered Servers view. After you register a server, you can connect to the server to work with it and then disconnect when you have finished simply by double-clicking the server entry in the Registered Servers view. If you are not automatically connected, you can force a connection by right-clicking the server entry and then selecting New Query (if you want to create an SQL query) or Object Explorer (if you want to view and manage the server).

You can start the registration process by using either of the following techniques:

- Register a server to which you are connected in Object Explorer.
- Register a new server in the Registered Servers view.

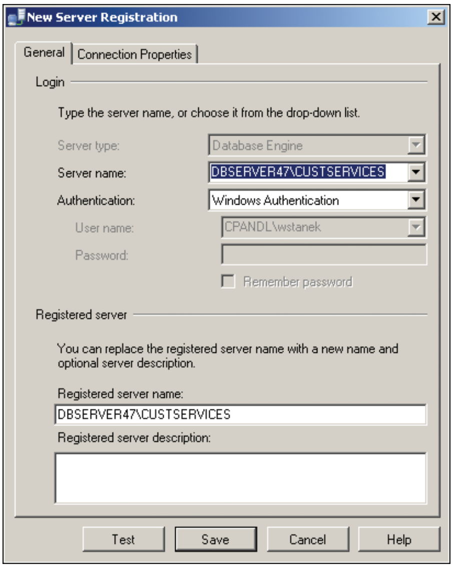
You can manage previous registrations in a variety of ways:

- Import registration information on previously registered SQL Server 2000 servers.
- Update registration information for local servers.
- Copy registration information from one computer to another by importing and exporting the information.

Registering a Connected Server

Any server to which you have connected in Object Explorer can be registered easily. Registration saves the current connection information and assigns the server to a group for easy future access using the Registered Servers view. To register a connected server, follow these steps:

1. In Object Explorer view, right-click any server to which you are currently connected, and then choose Register to display the New Server Registration dialog box, shown in Figure 1-6.
2. On the General tab, the current values for the server name and authentication type are filled in for you. Although the Registered Server Name option is set to the same value as the server name, you can modify this name and add a description.
3. On the Connection Properties tab, you can specify the database to which you want to connect and set options for networking and connections. If you want to encrypt the connection, select the Encrypt Connection check box.
4. To test your settings before you save the registration settings, click Test. If the test is unsuccessful, verify the settings and then make changes as necessary. As discussed in Chapter 2, SQL Server doesn't allow remote connections by default, so you must change the configuration settings to allow remote connections.
5. Click Save to save the server registration.



Type the server name, or choose it from the drop-down list.

Server type: Database Engine
Server name: DBSERVER47\CUSTSERVICES
Authentication: Windows Authentication
User name: CPANDL\wstanek
Password:
Remember password

Registered server

You can replace the registered server name with a new name and optional server description.

Registered server name: DBSERVER47\CUSTSERVICES
Registered server description:

FIGURE 1-6 The New Server Registration dialog box.

By default, the server is added to the top-level group. To move the server to a new level in the server group hierarchy, follow these steps:

1. Right-click the server in the Registered Servers view, point to Tasks, and then select Move To.
2. In the Move Server Registration dialog box, you can move the server to a different level by selecting the subgroup into which you want to place the server.
3. Click OK.

Registering a New Server in the Registered Servers View

You do not have to connect to a server in Object Explorer to register the server. You can register new servers directly in the Registered Servers view by following these steps:

1. In the Registered Servers view, use the toolbar to select the type of server you want to connect to, such as Database Engine.
2. Expand the available groups as necessary. In the Registered Servers view, right-click the group into which you want to register the server, and then select New Server Registration to display the New Server Registration dialog box, shown previously in Figure 1-6.
3. In the Server Name box, type the FQDN or host name of the server on which SQL Server is running, such as EngDBSrv12.cpandl.com or EngDBSrv12.
4. Use the Authentication list to choose the option for authentication type, which is either Windows Authentication or SQL Server Authentication (based on the authentication types selected when you installed the server). Provide a SQL Server login ID and password as necessary.
   - **Windows Authentication** Uses your current domain account and password to establish the database connection. This authentication type works only if Windows authentication is enabled and you have appropriate privileges.
   - **SQL Server Authentication** Allows you to specify a SQL Server login ID and password. To save the password so that you do not have to reenter it each time you connect, select Remember Password.
5. You also can specify connection settings by using the options on the Connection Properties tab. These options allow you to connect to a specific database instance and to set the network configuration. If you want to encrypt the connection, select the Encrypt Connection check box.
6. The registered server name is filled in for you based on the previously entered server name. Change the default name only if you want SQL Server Management Studio to use an alternate display name for the server.
7. To test the settings, click Test. If you successfully connect to the server, you see a prompt confirming this. If the test fails, verify the information you provided, make changes as necessary, and then test the settings again.
8. Click Save.

Registering Previously Registered SQL Server 2000 Servers

Registration details for servers registered by SQL Server 2000 can be imported into SQL Server Management Studio. This makes it easier to work with existing SQL Server 2000 installations. If the SQL Server 2000 installations were previously registered on the computer, you can import the registration details into a specific server group by completing the following steps:

1. In the Registered Servers view, use the toolbar to select the type of server you are registering, such as Database Engine.
2. Right-click the Local Server Groups entry, point to Tasks, and then select Previously Registered Servers.
3. Available registration information for SQL Server 2000 servers will be imported. If an error prompt is displayed, you might not be logged in locally to the computer on which the servers were registered previously.

Updating Registration for Local Servers

Local servers are registered automatically (in most cases). If you have added or removed SQL Server instances on the local computer and those instances are not displayed, you need to update the local server registration. Updating the registration information ensures that all currently configured local server instances are shown in SQL Server Management Studio.

To update registration details for local servers, follow these steps:

1. In the Registered Servers view, use the toolbar to select the type of servers you are registering, such as Database Engine.
2. Right-click the Local Server Groups entry, point to Tasks, and then select Register Local Servers.

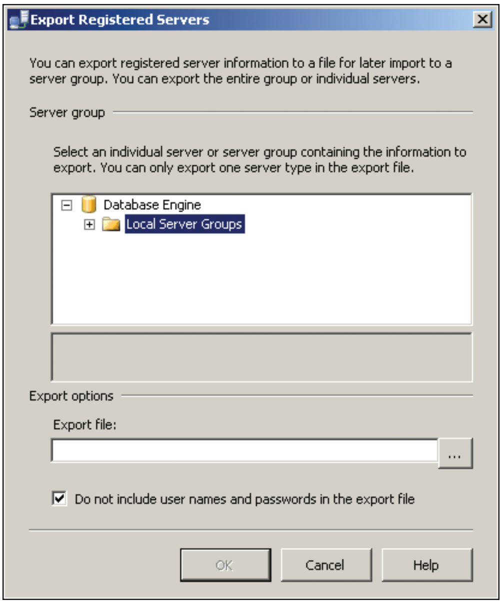
Copying Server Groups and Registration Details from One Computer to Another

After you register servers in SQL Server Management Studio and place the servers into a specific group hierarchy, you might find that you want to use the same registration information and server group structure on another computer. SQL Server Management Studio allows you to copy registration information from one computer to another by using an import/export process. You can copy the registration details with or without the user names and passwords.

To export the registration and group information to a file on one computer and then import it onto another computer, complete the following steps:

1. Start SQL Server Management Studio on the computer with the registration and group structure details that you want to copy.
2. Select the Registered Servers view by pressing Ctrl+Alt+G.

3. In the Registered Servers view, use the toolbar to select the type of servers you want to work with, such as Database Engine.
4. Right-click the Local Server Groups entry, point to Tasks, and then select Export to display the Export Registered Servers dialog box, shown in Figure 1-7.
5. Under Server Group, select the point from which the export process will begin. You can start copying registration information at any level in the group structure:
   - To copy the structure for a top-level group, all its subgroups, and all registration details for all related servers, select the Local Server Groups entry.
   - To copy the structure for a subgroup, its subgroups (if any), and all registration details for all related servers, select a subgroup.
   - To copy the registration details for a single server, select the server.
6. The server group structure and registration details are exported to a registration server file with the .regsrvr extension. By default, this file is created in the currently logged in user's Documents folder. Under Export Options, type a name for the registration server file, such as CurrentDBConfig.


Export Registered Servers

You can export registered server information to a file for later import to a server group. You can export the entire group or individual servers.

Server group

Select an individual server or server group containing the information to export. You can only export one server type in the export file.

Database Engine
Local Server Groups

Export options

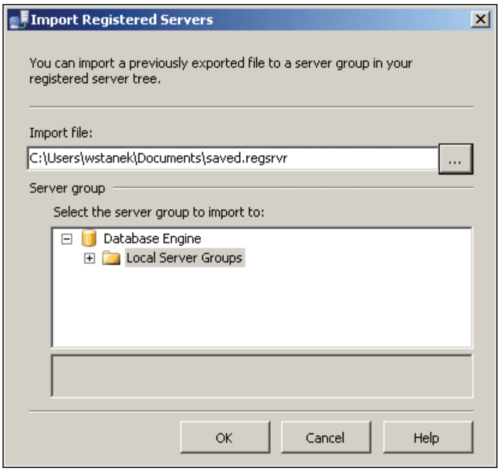
Export file:

[✓] Do not include user names and passwords in the export file

FIGURE 1-7 The Export Registered Servers dialog box.

TIP **If you place the registration server file on a secure network share, you can access it on the computer to which you want to copy the registration information. Otherwise, you need to copy this file to the destination computer later.**

7. By default, the current authentication details for server connections are not exported into the saved file. If you want to export user names and passwords, clear the Do Not Include User Names And Passwords In The Export File check box.
8. Click OK. If the export is successful, you see a dialog box confirming this. Click OK in the dialog box. If there is a problem, note and correct the problem.
9. Start SQL Server Management Studio on the computer to which you want to copy the server group and registration details. If you did not place the registration server file on a secure network share, you need to copy the file to this computer now.
10. Select the Registered Servers view by pressing Ctrl+Alt+G.
11. In the Registered Servers view, use the toolbar to select the type of server you want to work with, such as Database Engine.
12. Right-click the Local Server Groups entry, point to Tasks, and then select Import to display the Import Registered Servers dialog box, shown in Figure 1-8.


Import Registered Servers

You can import a previously exported file to a server group in your registered server tree.

Import file: C:\Users\wstanek\Documents\saved.regsrvr

Server group

Select the server group to import to:

Database Engine
Local Server Groups

FIGURE 1-8 The Import Registered Servers dialog box.

13. In the dialog box, click the button to the right of the Import File text box, and then use the Open dialog box that appears to select the registration server file you want to import.
14. Under Server Group, select the server group under which you want the imported groups and servers to be created.
15. Click OK. If the import is successful, you see a dialog box confirming this. Click OK in the dialog box. If there is a problem, note and correct it.

Editing Registration Properties

You can change a server's registration properties at any time by right-clicking the server entry in the Registered Servers view in SQL Server Management Studio and then selecting Properties. Use the Edit Server Registration Properties dialog box to make changes. The only property you cannot change is the server type. Be sure to test the settings before saving them.

Connecting to a Server

After you register a server, connecting to it is easy. Right-click the server entry in the Registered Servers view in SQL Server Management Studio, and then select New Query (if you want to create an SQL query) or Object Explorer (if you want to view and manage the server). You also can double-click the server entry to establish a connection and then work with the server in the Object Explorer view.

> ***NOTE*** **SQL Server Management Studio connects to other servers that are running SQL Server by using the network protocol set in the registration properties. If you have disabled the network protocol or remote access entirely for a server, you won't be able to connect to that server in SQL Server Management Studio. You need to make the appropriate changes in the registration properties or in the surface area configuration. Chapter 2 discusses surface area configuration.**

Disconnecting from a Server

When you have finished working with a server, you can disconnect from it. This eliminates the back-and-forth communications to the server. To disconnect, right-click the server's entry in the Object Explorer view in SQL Server Management Studio, and then select Disconnect from the shortcut menu.

Moving a Server to a New Group

To move the server to a new group, complete the following steps:

1. Right-click the server you want to move in the Registered Servers view, point to Tasks, and then select Move To from the shortcut menu to display the Move Server Registration dialog box.
2. In the Move Server Registration dialog box, expand the Local Server Groups entry to see a list of subgroups. Expand the subgroups as necessary. You can now do the following:
   - Move the server to the top-level group by selecting the top-level group. This makes the server a member of the top-level group.
   - Move the server to a different level by selecting the subgroup into which you want to place the server.
3. Click OK.

Deleting a Server Registration

If you change a server name or remove a server, you might want to delete the server registration in SQL Server Management Studio so that SQL Server Management Studio no longer tries to connect to a server that cannot be accessed. Right-click the server entry in the Registered Servers view and then select Delete. When prompted to confirm the action, click Yes to delete the server registration details.

Using Windows PowerShell for SQL Server Management

The graphical management tools provide just about everything you need to work with SQL Server. Still, there are many times when you might want to work from the command line, such as when you are working on a Windows Server 2008 R2 Core installation. To help with all your command-line needs, SQL Server 2012 includes the SQL Server provider for Windows PowerShell (also known as "SQL Server PowerShell"). To work with SQL Server via Windows PowerShell, you must first open a Command Prompt window or Windows PowerShell prompt and then start SQL Server PowerShell by typing **sqlps** at the command line.

Windows PowerShell introduces the concept of a *cmdlet* (pronounced "commandlet"). A cmdlet is the smallest unit of functionality in Windows PowerShell. Cmdlet names are not case-sensitive. SQL Server PowerShell cmdlets include the following:

- **Backup-SQLDatabase** Performs backup operations on SQL Server databases.
- **Convert-UrnToPath** Converts a SQL Server Management Object Uniform Resource Name (URN) to a SQL Server provider path. The URN indicates a management object's location within the SQL Server object hierarchy. If the URN path has characters not supported by Windows PowerShell, the characters are encoded automatically.
- **Decode-SQLName** Returns an unencoded SQL Server identifier when given an identifier that has been encoded.
- **Encode-SQLName** Encodes special characters in SQL Server identifiers and name paths to formats that are usable in Windows PowerShell paths. The characters encoded by this cmdlet include \:/%<>*?[]|. If you don't encode these characters, you must escape them by using the single quotation mark (') character.
- **Invoke-PolicyEvaluation** Evaluates management policies applied to SQL Server instances. By default, this cmdlet reports compliance but does not enforce compliance. To enforce compliance, set –AdHocPolicyEvaluationMode to Configure.

- **Invoke-Sqlcmd** Runs a Transact-SQL (T-SQL) or XQuery script containing commands supported by the SQLCMD utility. By default, this cmdlet doesn't set any SQLCMD variables or return message output; only a subset of SQLCMD commands can be used.
- **Restore-SQLDatabase** Performs restore operations on SQL Server databases.

To get detailed information about a cmdlet, type **get-help *cmdletname* –detailed,** where *cmdletname* is the name of the cmdlet you want to examine. To get detailed information about the SQL Server provider, which provides SQL Server functionality for Windows PowerShell, type **get-help sqlserver | more**.

REAL WORLD **You can use the sqlps utility on any computer where you've installed SQL Server or the command-line management tools. The sqlps utility starts a Windows PowerShell session with the SQL Server PowerShell provider imported so that you can use its cmdlets and work with instances of SQL Server. When you are working with Windows PowerShell or scripts, you can import the SQLPS module to load the SQL Server provider, which automatically loads the required assemblies and initializes the environment. While you previously needed to use an initialization script, this is no longer required so long as you import the SQLPS module prior to trying to access the SQL Server instance. For best results, import the SQLPS module using the following command:**

```
Import-Module "sqlps" –DisableNameChecking
```

You can work with cmdlets by executing commands directly at the shell prompt or by running commands from scripts. You can enter any command or cmdlet that you can run at the Windows PowerShell command prompt into a script by copying the related command text to a file and saving the file with the *.ps1* extension. You can then run the script in the same way that you would any other command or cmdlet. However, when you are working with Windows PowerShell, the current directory might not be part of the environment path. For this reason, you might need to use the *./* notation when you run a script in the current directory, such as the following:

```
./runtasks
```

The current execution policy for SQL Server PowerShell controls whether and how you can run scripts. Although the default configuration depends on which operating system and edition you've installed, you can quickly determine the execution policy by entering **get-executionpolicy** at the Windows PowerShell prompt.

To set the execution policy to require that all scripts have a trusted signature to execute, enter the following command:

```
set-executionpolicy allsigned
```

To set the execution policy so that scripts downloaded from the web execute only if they are signed by a trusted source, enter:

```
set-executionpolicy remotesigned
```

To set the execution policy to run scripts regardless of whether they have a digital signature and work in an unrestricted environment, you can enter the following command:

```
set-executionpolicy unrestricted
```

For administration at the Windows PowerShell prompt, you use Invoke-Sqlcmd to run T-SQL or XQuery scripts containing commands supported by the SQLCMD utility. Invoke-Sqlcmd fully supports T-SQL and the XQuery syntax supported by the Database Engine, but it does not set any scripting variables by default. Invoke-Sqlcmd also accepts the SQLCMD commands listed in Table 1-3, later in this chapter. By default, results are formatted as a table, with the first result set displayed automatically and subsequent result sets displayed only if they have the same column list as the first result set.

The basic syntax you use most often with Invoke-Sqlcmd follows:

```
Invoke-Sqlcmd [-ServerInstance ServerStringOrObject]
[-Database DatabaseName] [-EncryptConnection ]
[-Username UserName] [-Password Password] [[-Query] QueryString]
[-DedicatedAdministratorConnection]

[-InputFile FilePath] [ | Out-File –filepath FilePath]
```

The command's parameters are used as follows:

- **–Database** Specifies the name of the database that you want to work with. If you don't use this parameter, the database that is used depends on whether the current path specifies both the SQLSERVER:\SQL folder and a database name. If both are specified, Invoke-Sqlcmd connects to the database that is specified in the path. Otherwise, Invoke-Sqlcmd connects to the default database for the current login ID.

  NOTE **Use–IgnoreProviderContext to force a connection to the database that is defined as the default for the current login ID.**

- **–DedicatedAdministratorConnection** Ensures that a dedicated administrator connection (DAC) is used to force a connection when one might not be possible otherwise.
- **–EncryptConnection** Enables Secure Sockets Layer (SSL) encryption for the connection.
- **–InputFile** Provides the full path to a file that should be used as the query input. The file can contain T-SQL statements, XQuery statements, SQLCMD commands, and scripting variables. Spaces are not allowed in the file path or file name.

- **–Password** Sets the password for the SQL Server Authentication login ID that is specified in –Username.
- **–Query** Defines one or more queries to be run. The queries can be T-SQL queries, XQuery statements, or SQLCMD commands. Separate multiple queries with semicolons.

  TIP **You do not need to use the SQLCMD GO command. Escape any double quotation marks included in the string and consider using bracketed identifiers such as [EmpTable] instead of quoted identifiers such as "EmpTable". To ensure that the message output is returned, add the –Verbose parameter. –Verbose is a parameter common to all cmdlets.**

- **–ServerInstance** Specifies the name of an instance of the Database Engine that you want to work with. For default instances, specify only the computer name, such as DbServer23. For named instances, use the format "ComputerName\InstanceName", such as DbServer23\EmployeeDb.
- **–Username** Sets the login ID for making a SQL Server authentication connection to an instance of the Database Engine. You also must set the password for the login ID.

NOTE **By default, Invoke-Sqlcmd attempts a Windows authentication connection by using the Windows account running the Windows PowerShell session. Windows authentication connections are preferred. To use a SQL Server authentication connection instead, specify the user name and password for the SQL login ID that you want to use.**

With this in mind, you could replace the following T-SQL statements:

```
USE OrderSystem;
GO
SELECT * FROM Inventory.Product
ORDER BY Name ASC
GO
```

with the following Windows PowerShell command:

```
Invoke-Sqlcmd -Query "SELECT * FROM Inventory.Product; ORDER BY Name ASC"
-ServerInstance "DbServer23\OrderSystem"
```

You also could read the commands from a script, as shown in Sample 1-1.

SAMPLE 1-1 Example SQL Command Script.

Contents of SqlCmd.sql Script.

```
SELECT * FROM Inventory.Product
ORDER BY Name ASC
```

Command to Run the Script

```
Invoke-Sqlcmd -InputFile "C:\Scripts\SqlCmd.sql"
```

When you work with Windows PowerShell, don't overlook the importance of SQL Server support being implemented through a provider. The data that providers expose appears as a drive that you can browse. One way to browse is to get or set the location with respect to the SqlServer: provider drive. The top of the hierarchy exposed is represented by the SQL folder, then there is a folder for the machine name, and finally, there is a folder for the instance name. Following this, you could navigate to the top-level folder for the default instance by entering

```
Set-Location SQLSERVER:\SQL\DbServer23\Default
```

You could then determine the available database structures by entering Get-ChildItem (or one of its aliases, such as ls or dir). To navigate logins, triggers, endpoints, databases, and any other structures, you set the location to the name of the related folder. For example, you could use Set-Location Databases and then enter **Get-ChildItem** to list available databases for the selected instance. Of course, if you know the full path you want to work with in the first place, you also can access it directly, as shown in the following example:

```
Set-Location SQLSERVER:\SQL\DbServer23\Default\Databases\OrderSystem
```

Here, you navigate to the structures for the OrderSystem database on DbServer23's default instance. If you then want to determine what tables are available for this database, you could enter:

```
Get-ChildItem Tables
```

Or you could enter:

```
Set-location Tables
Get-ChildItem
```

To manage SQL Server 2012 from a computer that isn't running SQL Server, you need to install the management tools. In the SQL Server Installation Center, select Installation, and then click the New Installation Or Add Features To An Existing Installation option. When the wizard starts, follow the prompts. On the Feature Selection page, select the Management Tools—Basic option to install Management Studio, SQLCMD, and the SQL Server provider for Windows PowerShell.

For remote management via Windows PowerShell, you need to ensure that Windows Remote Management (WinRM) and Windows PowerShell are both installed and made available by using the Add Features Wizard. You also need to enable remote commands on both your management computer and the server running SQL Server.

You can verify the availability of WinRM and configure Windows PowerShell for remoting by following these steps:

1. Click Start, All Programs, Accessories, and Windows PowerShell. Then start Windows PowerShell as an administrator by right-clicking the Windows PowerShell shortcut and selecting Run As Administrator.

2. The WinRM service is configured for manual startup by default. You must change the startup type to Automatic and start the service on each computer you want to work with. At the PowerShell prompt, you can verify that the WinRM service is running by using the following command:

   ```
   get-service winrm
   ```

   As shown in the following example, the value of the Status property in the output should be Running:

   ```
   Status   Name               DisplayName
   ------   ----               -----------
   Running  WinRM              Windows Remote Management
   ```

   If the service is stopped, enter the following command to start the service and configure it to start automatically in the future:

   ```
   set-service –name winrm –startuptype automatic –status running
   ```

3. To configure Windows PowerShell for remoting, type the following command:

   ```
   Enable-PSRemoting –force
   ```

   You can enable remoting only when your computer is connected to a domain or private network. If your computer is connected to a public network, you need to disconnect from the public network and connect to a domain or private network and then repeat this step. If one or more of your computer's connections has the Public connection type but you are actually connected to a domain or private network, you need to change the network connection type in the Network And Sharing Center and then repeat this step.

In many cases, you can work with remote computers in other domains. However, if the remote computer is not in a trusted domain, the remote computer might not be able to authenticate your credentials. To enable authentication, you need to add the remote computer to the list of trusted hosts for the local computer in WinRM. To do so, type the following:

```
winrm s winrm/config/client '@{TrustedHosts="RemoteComputer"}'
```

where *RemoteComputer* is the name of the remote computer, such as

```
winrm s winrm/config/client '@{TrustedHosts="DbServer23"}'
```

When you are working with computers in workgroups or homegroups, you must use HTTPS as the transport or add the remote machine to the TrustedHosts configuration settings. If you cannot connect to a remote host, you can verify that the service on the remote host is running and is accepting requests by running the following command on the remote host:

```
winrm quickconfig
```

This command analyzes and configures the WinRM service. If the WinRM service is set up correctly, you see output similar to the following:

```
WinRM already is set up to receive requests on this machine.
WinRM already is set up for remote management on this machine.
```

If the WinRM service is not set up correctly, you see errors and need to respond affirmatively to several prompts that allow you to configure remote management automatically. When this process is complete, WinRM should be set up correctly. Don’t forget that you need to enable remote management on the database server as well as your management computer.

Starting, Stopping, and Configuring SQL Server Agent

SQL Server Agent runs as a service and is used to schedule jobs, alerts, and other automated tasks. After you have scheduled automated tasks, you usually want SQL Server Agent to start automatically when the system starts. This configuration ensures that the scheduled tasks are performed as expected. Using SQL Server Service Manager, you can control the related SQL Server Agent (*InstanceName*) service just as you do the SQL Server service. For details, see the “Configuring SQL Server Services” section in Chapter 2.

You use SQL Server Management Studio to configure SQL Server Agent. Chapter 10, “Automating and Maintaining SQL Server 2012,” covers the agent configuration in detail, but the basic steps are as follows:

1. Connect to the Database Engine on the server you want to configure. You can do this in the Registered Servers view by double-clicking the server entry, or you can use the Object Explorer view. In the Object Explorer view, click Connect, and then select Database Engine to display the Connect To Server dialog box, which you can use to connect to the server.
2. Right-click the SQL Server Agent node, and then select Properties from the shortcut menu. You can now configure SQL Server Agent. Keep in mind that if the service is not running, you need to start it before you can manage its properties.
3. The SQL Server Agent shortcut menu also lets you manage the SQL Server Agent service. Select Start, Stop, or Restart as desired.

Starting, Stopping, and Configuring MSDTC

Microsoft Distributed Transaction Coordinator (MSDTC) is a transaction manager that makes it possible for client applications to work with multiple sources of data in one transaction.

When a distributed transaction spans two or more servers, the servers coordinate the management of the transaction by using MSDTC. When a distributed transaction spans multiple databases on a single server, SQL Server manages the transaction internally.

SQL Server applications can call MSDTC directly to start an explicit distributed transaction. Distributed transactions can also be started implicitly by using one of the following methods:

- Calling stored procedures on remote servers running SQL Server
- Updating data on multiple OLE DB data sources
- Enlisting remote servers in a transaction

If you work with transactions under any of these scenarios, you should have MSDTC running on the server, and you should set MSDTC to start automatically when the server starts. As with SQL Server itself, MSDTC runs as a service. Unlike the SQL Server service, only one instance of the MSDTC service runs on a computer, regardless of how many database server instances are available. This means that all instances of SQL Server running on a computer use the same transaction coordinator.

You can view the current state of MSDTC in SQL Server Management Studio by connecting to the server's Database Engine. In Object Explorer, expand the server and Management nodes. If the service is running, you see a green circle with a right-facing triangle in it (similar to a Play button). If the service is stopped, you see a red circle with a square in it (similar to a Stop button). You can control the MSDTC service with Computer Management. Follow these steps:

1. Start Computer Management by clicking Start, pointing to All Programs, Administrative Tools, and then selecting Computer Management.
2. By default, you are connected to the local computer. To connect to a remote computer, right-click the Computer Management node, and then select Connect To Another Computer. In the Select Computer dialog box, choose Another Computer, and then type the name of the computer. The name can be specified as a host name, such as EngDBSrv12, or as an FQDN, such as EngDBSrv12.cpandl.com.
3. Expand Services And Applications, and then select Services. Right-click Distributed Transaction Coordinator, and then choose Properties. You can now manage MSDTC.

Managing SQL Server Startup

The Database Engine has two modes of operation. It can run as a service or as a command-line application (SQLServr.exe). You normally run SQL Server as a service. Use the command-line application when you need to troubleshoot problems or modify configuration settings in single-user mode.

Enabling or Preventing Automatic SQL Server Startup

In Chapter 2, you'll see how SQL Server Configuration Manager is used to manage the SQL Server (MSSQLSERVER) service, related services for other Database Engine instances, and other SQL Server–related services. Any of these services can be configured for automatic startup or can be prevented from starting automatically. To enable or prevent automatic startup of a service, follow these steps:

1. Start SQL Server Configuration Manager by using one of the following techniques:
   - Log in to the database server through a local or remote login, and then start SQL Server Configuration Manager by clicking Start; pointing to All Programs, Microsoft SQL Server 2012, Configuration Tools; and then selecting SQL Server Configuration Manager.
   - In SQL Server Management Studio, open the Registered Servers view by pressing Ctrl+Alt+G. Use the Registered Servers toolbar to select the top-level group, and then expand the group nodes by double-clicking them. Right-click the server entry, and then select SQL Server Configuration Manager.
2. Select the SQL Server Services node. Right-click the SQL Server service that you want to start automatically, and then select Properties. You can now do the following:
   - **Enable automatic startup** On the Service tab, set the Start Mode to Automatic. If the server state is Stopped, click Start on the Log On tab to start the service.
   - **Prevent automatic startup** On the Service tab, set the Start Mode to Manual.
3. Click OK.

You can also use Computer Management to configure services. To configure automatic startup of a service by using Computer Management, follow these steps:

1. Click Start, point to All Programs, Administrative Tools, and then select Computer Management.
2. By default, you are connected to the local computer. To connect to a remote computer, right-click the Computer Management node and select Connect To Another Computer. In the Select Computer dialog box, select Another Computer, and then type the name of the computer. The name can be specified as a host name, such as EngDBSrv12, or as an FQDN, such as EngDBSrv12.cpandl.com.
3. Expand Services And Applications, and then select Services.
4. Right-click the SQL Server service that you want to start automatically, and then select Properties.

5. Now you can do the following:
   - **Enable automatic startup** On the General tab, set the Startup Type to Automatic. If the Service Status reads Stopped, click Start.
   - **Prevent automatic startup** On the General tab, set the Startup Type to Manual.
6. Click OK.

Setting Database Engine Startup Parameters

Startup parameters control how the SQL Server Database Engine starts and which options are set when it does. You can configure startup options by using SQL Server Configuration Manager or Computer Management. SQL Server Configuration Manager is the recommended tool for this task because it provides the current default settings and allows you to make modifications easily.

TIP **You can pass startup parameters to the command-line utility SQLServr.exe as well. Passing the –c option to this utility starts SQL Server without using a service. You must run SQLServr.exe from the Binn directory that corresponds to the instance of the SQL Server Database Engine that you want to start. For the default instance, the utility is located in MSSQL11.MSSQLSERVER\MSSQL\Binn. For named instances, the utility is located in MSSQL11.*InstanceName*\MSSQL\Binn.**

Adding Startup Parameters

You can add startup parameters by completing the following steps:

1. Start SQL Server Configuration Manager by using one of the following techniques:
   - Log in to the database server through a local or remote login, and then start SQL Server Configuration Manager. On the Microsoft SQL Server 2012 menu, the related option is found under Configuration Tools.
   - In SQL Server Management Studio, open the Registered Servers view by pressing Ctrl+Alt+G. Use the Registered Servers toolbar to select the top-level group, and then expand the group nodes by double-clicking them. Right-click the server entry, and then select SQL Server Configuration Manager.
2. Select the SQL Server Services node. Right-click the SQL Server service that you want to modify, and then select Properties.
3. On the Advanced tab, click in the Startup Parameters box, and then press End to go to the end of the currently entered parameters. The –d, –e, and –l parameters are set by default. Be careful not to modify these or other existing parameters accidentally.

4. Each parameter is separated by a semicolon. Type a semicolon and then a hyphen followed by the letter and value of the parameter you are adding, such as `;–g512`.
5. Repeat steps 3 and 4 as necessary to specify additional parameters and values.
6. Click Apply to save the changes. The parameters are applied the next time the SQL Server instance is started. To apply the parameters right away, you must stop and then start the service by clicking Restart on the Log On tab.

Removing Startup Parameters

You can remove startup parameters by completing the following steps:

1. Start SQL Server Configuration Manager by using one of the following techniques:
   - Log in to the database server through a local or remote login, and then start SQL Server Configuration Manager by clicking Start, pointing to All Programs, Microsoft SQL Server 2012, Configuration Tools, and then selecting SQL Server Configuration Manager.
   - In SQL Server Management Studio, open the Registered Servers view by pressing Ctrl+Alt+G. Use the Registered Servers toolbar to select the top-level group, and then expand the group nodes by double-clicking them. Right-click the server entry, and then select SQL Server Configuration Manager.
2. Select the SQL Server Services node. Right-click the SQL Server service that you want to modify, and then select Properties.
3. On the Advanced tab, click in the Startup Parameters box. Each parameter is specified with a hyphen, parameter letter, and parameter value. A semicolon is used to separate parameter values, as shown in the following example:

   ```
   -g512;
   ```
4. Remove the parameter by deleting its entry.
5. The change is applied the next time the SQL Server instance is started. To apply the change right away, you must stop and then start the service by clicking Restart on the Log On tab.

Common Startup Parameters

Table 1-1 shows the startup parameters in SQL Server and how they are used. The first three parameters (–d, –e, and –l) are the defaults for SQL Server. The remaining parameters allow you to configure additional settings.

TABLE 1-1 Startup Parameters for SQL Server

PARAMETER	DESCRIPTION
–d<*path*>	Sets the full path for the *master* database. If this parameter is omitted, the registry values are used. *Example:* –dC:\Program Files\Microsoft SQL Server\MSSQL11 .MSSQLSERVER\MSSQL\DATA\Master.mdf
–e<*path*>	Sets the full path for the error log. If this parameter is omitted, the registry values are used. *Example:* –eC:\Program Files\Microsoft SQL Server\MSSQL11 .MSSQLSERVER\MSSQL\LOG\ERRORLOG
–l<*path*>	Sets the full path for the *master* database transaction log. If this parameter is omitted, the registry values are used. *Example:* –lC:\Program Files\Microsoft SQL Server\MSSQL11 .MSSQLSERVER\MSSQL\DATA\Mastlog.ldf
–B	Sets a breakpoint in response to an error; used with the –y option when debugging.
–c	Prevents SQL Server from running as a service. This setting makes startup faster when you are running SQL Server from the command line.
-E	Increases the number of extents that are allocated for each file in a file group. Useful for data warehouse applications with a limited number of users.
–f	Starts SQL Server with minimal configuration. This setting is useful if a configuration value has prevented SQL Server from starting.
–g *number*	Specifies the amount of virtual address space memory in megabytes to reserve for SQL Server. This memory is outside the SQL Server memory pool and is used by the extended procedure dynamic-link libraries (DLLs), the OLE DB providers referenced in distributed queries, and the automation object referenced in T-SQL. The default value is 256. *Example:* –g256
–K	Forces regeneration of the service master key if it exists.
–k *number*	Sets the checkpoint speed in megabytes per second. Use a decimal value. *Example:* –k25

PARAMETER	DESCRIPTION
–m	Starts SQL Server in single-user mode. Only a single user can connect, and the checkpoint process is not started. Enables the sp_configure allow updates option, which is disabled by default.
–n	Tells SQL Server not to log errors in the application event log. Use with –e.
–s *instance*	Starts the named instance of SQL Server. You must be in the relevant Binn directory for the instance. *Example:* –sdevapps
–T<*tnum*>	Sets a trace flag. Trace flags set nonstandard behavior and are often used in debugging or diagnosing performance issues. *Example:* –T237
–t<*tnum*>	Sets an internal trace flag for SQL Server. Used only by SQL Server support engineers. *Example:* –t8837
–x	Disables statistics tracking for CPU time and cache-hit ratio. Allows maximum performance.
–y *number*	Sets an error number that causes SQL Server to dump the stack. *Example:* –y1803

Managing Services from the Command Line

You can start, stop, and pause SQL Server as you would any other service. On a local system, you can type the necessary command at a standard command prompt. You also can connect to a system remotely and then issue the necessary command. To manage the default database server instance, use these commands:

- **NET START MSSQLSERVER** Starts SQL Server as a service.
- **NET STOP MSSQLSERVER** Stops SQL Server when running as a service.
- **NET PAUSE MSSQLSERVER** Pauses SQL Server when running as a service.
- **NET CONTINUE MSSQLSERVER** Resumes SQL Server when it is running as a service.

To manage named instances of SQL Server, use the following commands:

- **NET START MSSQL$*instancename*** Starts SQL Server as a service; *instancename* is the actual name of the database server instance.
- **NET STOP MSSQL$*instancename*** Stops SQL Server when it is running as a service; *instancename* is the actual name of the database server instance.

- **NET PAUSE MSSQL$*instancename*** Pauses SQL Server when it is running as a service; *instancename* is the actual name of the database server instance.
- **NET CONTINUE MSSQL$*instancename*** Resumes SQL Server when it is running as a service; *instancename* is the actual name of the database server instance.

You can add startup options to the end of NET START MSSQLSERVER or NET START MSSQL$*instancename* commands. Use a slash (/) instead of a hyphen (–), as shown in these examples:

```
net start MSSQLSERVER /f /m
net start MSSQL$CUSTDATAWAREHOUS /f /m
```

REAL WORLD **Instead of referencing MSSQLSERVER or MSSQL$*instancename*, you also can refer to the service by its display name. For the default instance, you use** `"SQL Server (MSSQLSERVER)"` **with** `NET START`, `NET STOP`, `NET PAUSE`, **and** `NET CONTINUE`**. For a named instance, you use** `net start "SQL Server (InstanceName)"`, **where *InstanceName* is the name of the instance, such as** `net start "SQL Server (CUSTDATAWAREHOUS)"`**. In both usages, the quotation marks are required as part of the command text.**

Managing the SQL Server Command-Line Executable

The SQL Server command-line executable (SQLServr.exe) provides an alternative to the SQL Server service. You must run SQLServr.exe from the Binn directory that corresponds to the instance of the SQL Server Database Engine that you want to start. For the default instance, the utility is located in MSSQL11.MSSQLSERVER\MSSQL\Binn. For named instances, the utility is located in MSSQL11.*InstanceName*\MSSQL\Binn.

When SQL Server is installed on a local system, start SQL Server by changing to the directory where the instance of SQL Server you want to start is located and then typing **sqlservr** at the command line. On a remote system, connect to the system remotely, change to the appropriate directory, and then issue the startup command. Either way, SQL Server reads the default startup parameters from the registry and starts execution.

You also can enter startup parameters and switches that override the default settings. (The available parameters are summarized in Table 1-1.) You still can connect SQL Server Management Studio and SQL Server Configuration Manager to the server. However, when you do, these programs show an icon indicating that the SQL Server service is stopped because you aren't running SQL Server via the related service. You also will be unable to pause, stop, or resume the instance of SQL Server as a Windows service.

When you are running SQL Server from the command line, SQL Server runs in the security context of the user, not the security context of the account assigned to the SQL Server service. You should not minimize the command console in which SQL Server is running because doing so causes Windows to remove nearly all the resources from SQL Server.

In addition, when you are running SQL Server from the command line, you can make configuration changes that might be necessary for diagnosing and resolving problems, and you also can perform tasks that you can accomplish only when SQL Server is running in single-user mode. However, you should be careful when creating databases, changing data file locations, or making other similar types of changes. If you are logged in as an administrator and create a new database or change the location of a data file, SQL Server might not be able to access the database or data file when it runs later under the default account for the SQL Server service.

You must shut down the instance of SQL Server before logging off Windows. To stop an instance of SQL Server started from the command line, complete the following steps:

1. Press Ctrl+C to break into the execution stream.
2. When prompted, press Y to stop SQL Server.

Managing Server Activity

As a database administrator, your job is to be sure that SQL Server runs smoothly. To ensure that SQL Server is running optimally, you can actively monitor the server to do the following:

- Keep track of user connections and locks.
- View processes and commands that active users are running.
- Check the status of locks on processes and objects.
- See blocked or blocking transactions.
- Ensure that processes complete successfully, and detect errors if they do not.

When problems arise, you can terminate a process if necessary.

NOTE **For more coverage of monitoring SQL Server, see Chapter 12. In that chapter, you will learn how to use Performance Monitor and SQL Server Profiler to keep track of SQL Server activity, performance, and errors.**

Examining Process Information

Process information provides details about the status of processes, current user connections, and other server activity. You can view process information by completing the following steps:

1. Start SQL Server Management Studio, and then connect to a server.
2. Use the Object Explorer view to access an instance of the Database Engine.
3. Right-click the Database Engine instance and then select Activity Monitor.

In Activity Monitor, shown in Figure 1-9, you should see a graphical overview of the server activity, as well as an activity summary by processes, resource waits, data file I/O, and recent expensive queries. The overview and the summaries are provided in separate panels that you can expand to display or shrink to hide.

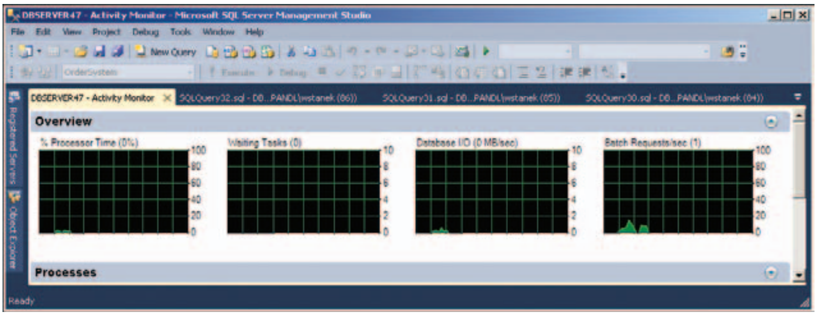

DBSERVER47 - Activity Monitor - Microsoft SQL Server Management Studio
File Edit View Project Debug Tools Window Help
New Query
OrderSystem
DBSERVER47 - Activity Monitor
SQLQuery32.sql - DB...PANDL\wstanek (86))
SQLQuery31.sql - DB...PANDL\wstanek (85))
SQLQuery30.sql - DB...PANDL\wstanek (84))
Registered Servers
Object Explorer
Overview
% Processor Time (0%)
Waiting Tasks (0)
Database I/O (0 MB/sec)
Batch Requests/sec (1)
Processes
Ready

FIGURE 1-9 Activity Monitor.

The Overview panel has graphs depicting processor time, waiting tasks, database I/O, and batch requests. By default, the graphs are updated every 10 seconds. You can specify a different refresh interval by right-clicking in the panel, pointing to Refresh Interval, and then selecting an interval, such as 30 seconds.

In the Processes panel, processes are sorted by process ID by default, but you can arrange them by any of the available information categories summarized in Table 1-2. Click a category header to sort processes based on that category. Click the same category header again to perform a reverse sort on the category.

TABLE 1-2 Process Information Used in Database Administration

CATEGORY	DESCRIPTION
Session ID	Provides the session ID of the process on the server.
User Process	Flags the process as being either a user process (flag=1) or a server process (flag=0).
Login	Shows which user is running the process by SQL Server ID, service name, or domain account.
Database	Indicates the database with which the process is associated.
Task State	Shows the status of the process. A running process is active and currently performing work. A runnable process has a connection but currently has no work to perform. A sleeping process is waiting for something, such as user input or a lock. A background process is running in the background, periodically performing tasks. A suspended process has work to perform but has stopped.

CATEGORY	DESCRIPTION
Command	Displays the command being executed or the last command executed.
Application	Shows the application or SQL Server component (such as a report server) connecting to the server and running the process.
Wait Time	Indicates the elapsed wait time in milliseconds.
Wait Type	Specifies whether the process is waiting or not waiting.
Wait Resource	Displays the resource that the process is waiting for (if any).
Blocked By	Displays the process ID blocking this process.
Head Blocker	Shows 1 if the session ID is the head blocker in the blocking chain. Otherwise, it shows 0.
Memory Use	Displays the amount of memory the process is using (in kilobytes).
Host Name	Displays the host from which the connection originated.
Workload Group	Displays the name of the Resource Governor workload group for the query.

Tracking Resource Waits and Blocks

When you are diagnosing performance issues, you should look closely at the Wait Time, Wait Type, Wait Resource, and Blocked By values for each process. Most of the process information is gathered from data columns returned by various dynamic management views, including the following:

- **sys.dm_os_tasks** Returns information about each task that is active in the instance of SQL Server.
- **sys.dm_os_waiting_tasks** Returns information about each task that is waiting on some resource.
- **sys.dm_exec_requests** Returns information about each request that is executing within SQL Server.
- **sys.dm_exec_sessions** Returns information about each authentication session within SQL Server.

Although Activity Monitor provides a good overview, you might need to use these dynamic management views to get more detailed information about processes, resource waits, and resource blocks.

The Resource Waits panel provides additional information about resource waits. Each wait category combines the wait time for closely related wait types, such as buffer I/O or network I/O. Keep the following in mind:

- **Wait Time** Shows the accumulated wait time per second. Here, a rate of 3,000 ms indicates that three tasks on average were waiting with this wait category.
- **Recent Wait Time** Shows the wait average of accumulated wait time per second. This combines all the wait times over the last several minutes and averages them for this wait category.
- **Average Waiter Count** Shows the average number of waiting tasks per second for this wait category.
- **Cumulative Wait Time** Shows the total amount of wait time for this wait category since SQL Server was started or the wait statistics were reset.

TIP **You can reset wait statistics using DBCC SQLPERF, as shown in this example:**

```
DBCC SQLPERF("sys.dm_os_wait_stats",CLEAR);
```

To get a clearer picture of resource waits and blocks, you can use the sys.dm_tran_locks view. Table 1-3 summarizes the information returned with this view. Actual values are in parentheses, preceded by a general category name.

TABLE 1-3 Lock-Related Information Used in Database Administration

CATEGORY	TYPE	DESCRIPTION
Process ID (request_session_id)		The process ID of the related user process within SQL Server.
Object ID (resource_associated_entity_id)		The ID of the entity with which a resource is associated.
Context (request_exec_context_id)		The ID of the thread associated with the process ID.
Batch ID (request_request_id)		The batch ID associated with the process ID.
Type (resource_type)	RID	Row identifier; used to lock a single row within a table.
	KEY	A row lock within an index; used to protect key ranges.
	PAGE	A lock on a data or index page.
	EXTENT	A lock on a contiguous group of eight data or index pages.

CATEGORY	TYPE	DESCRIPTION
	TABLE	A lock on an entire table, including all data and indexes.
	DATABASE	A lock on an entire database.
	METADATA	A lock on descriptive information about the object.
	ALLOCATION_ UNIT	A lock on allocation unit page count statistics during deferred drop operations.
	HOBT	A lock on basic access path structures for heap or index reorganization operations or heap-optimized bulk loads.
Subtype (resource_subtype)		The lock subtype, frequently used with METADATA locks to identify metadata lock activity.
Description (resource_description)		Gives optional descriptive information.
Request Mode (request_mode)	S	Shared; used for read-only operations, such as a SELECT statement.
	U	Update; used when reading/ locking an updatable resource; prevents some deadlock situations.
	X	Exclusive; allows only one session to update the data; used with the modification operations, such as INSERT, DELETE, and UPDATE.
	I	Intent; used to establish a lock hierarchy.
	Sch-S	Schema stability; used when checking a table's schema.
	Sch-M	Schema modification; used when modifying a table's schema.

CATEGORY	TYPE	DESCRIPTION
	BU	Bulk update; used when bulk copying data into a table and the TABLOCK hint is specified.
	RangeS_S	Serializable range scan; used with shared resource locks on shared ranges.
	RangeS_U	Serializable update; used for updating resource locks on shared ranges.
	RangeI_N	Insert range with a null resource lock; used to test ranges before inserting a new key into an index.
	RangeX_X	Exclusive range with an exclusive lock; used when updating a key in a range.
Request Type (request_type)		The type of object requested.
Request Status (request_status)	GRANT	The lock was obtained.
	WAIT	The lock is blocked by another process.
	CNVT	The lock is being converted—that is, it is held in one mode but waiting to acquire a stronger lock mode.
Owner Type (request_owner_type)	CURSOR	The lock owner is a cursor.
	SESSION	The lock owner is a user session.
	TRANSACTION	The lock owner is a transaction.
	SHARED_ TRANSACTION _WORKSPACE	The lock owner is the shared portion of the transaction workspace.
	EXCLUSIVE_ TRANSACTION _WORKSPACE	The lock owner is the exclusive portion of the transaction workspace.

CATEGORY	TYPE	DESCRIPTION
Owner ID (request_owner_id)		The owner ID associated with the lock.
Owner GUID (request_owner_guid)		The globally unique identifier (GUID) of the owner associated with the lock.
Database (resource_database_id)		The database containing the lock.
Object (resource_associated_ entity_id)		The name of the object being locked.

Troubleshooting Deadlocks and Blocking Connections

Two common problems you might encounter are deadlocks and blocking connections. Deadlocks and blocking connections, as described in the following list, can occur in almost any database environment, especially when many users are making connections to databases:

- Deadlocks can occur when two users have locks on separate objects and each wants a lock on the other's object. Each user waits for the other user to release the lock, but this does not happen.
- Blocking connections occur when one connection holds a lock and a second connection wants a conflicting lock type. This forces the second connection to wait or to block the first.

Both deadlocks and blocking connections can degrade server performance.

Although SQL Server can detect and correct deadlocks, you can help speed up this process by identifying potential problems and taking action when necessary. Clearing blocks is a manual step; you must kill the blocking process.

Process information can tell you when deadlocks or blocking connections occur. Examine these process information columns: Wait Time, Wait Type, Resource, Blocking, and Blocked By. When you have a deadlock or blocking situation, take a closer look at the locks on the objects that are causing problems. Refer to the "Tracking Resource Waits and Blocks" section earlier in this chapter for details. You also might want to stop the offending processes, and you can do this by following the steps described in the "Killing Server Processes" section later in this chapter.

You can also use the sys.dm_tran_locks view to obtain information about active locks. Each row in the results returned by this view represents a currently active request to the lock manager for a lock that has been granted or is waiting to be granted. The following example returns a list of locks in the Customer database:

T-SQL

```
USE customer;
GO
SELECT * FROM sys.dm_tran_locks
```

PowerShell

```
Invoke-Sqlcmd -Query "USE customer; SELECT * FROM sys.dm_tran_locks"
-ServerInstance "DbServer25"
```

In the result set, the results are organized in two main groups. Columns that begin with *resource_* describe the resource on which the lock request is being made. Columns that begin with *request_* describe the lock request itself. (Table 1-3 lists the correlation between the columns in the results and the categories listed in Activity Monitor.) While Activity Monitor returns the actual database name, the resource_database_id column returns the database_id as set in the sys.databases view. In SQL Server, database IDs are set on a per-server basis. You can determine the database name for a particular database ID on a particular server by using the following statement:

```
SELECT name, database_id FROM sys.databases
```

In the results returned by sys.dm_tran_locks, request_session_id tracks process IDs. Process IDs tracked internally by SQL Server do not correspond to process IDs tracked by the operating system. You can determine the association between the SQL Server process IDs and Windows thread IDs by using the following query:

```
SELECT ServerTasks.session_id, ServerThreads.os_thread_id
    FROM sys.dm_os_tasks AS ServerTasks
    INNER JOIN sys.dm_os_threads AS ServerThreads
        ON ServerTasks.worker_address = ServerThreads.worker_address
    WHERE ServerTasks.session_id IS NOT NULL
    ORDER BY ServerTasks.session_id;
GO
```

While you are connected to the database that contains the locking object, you get more information about the locking object and blocking information. Use the following query, where <*resource_associated_entity_id*> is the value in the related column, to get information about the locking object:

```
SELECT object_name(object_id), *
    FROM sys.partitions
    WHERE hobt_id=<resource_associated_entity_id>
```

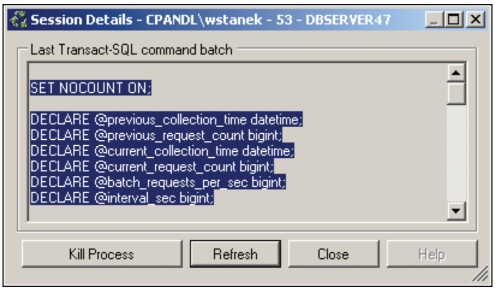
Use the following query to get blocking information:

```
SELECT
        tr1.resource_type,
        tr1.resource_subtype,
        tr1.resource_database_id,
        tr1.resource_associated_entity_id,
        tr1.request_mode,
        tr1.request_type,
        tr1.request_status,
        tr1.request_session_id,
        tr1.request_owner_type,
        tr2.blocking_session_id
    FROM sys.dm_tran_locks as tr1
    INNER JOIN sys.dm_os_waiting_tasks as tr2
        ON tr1.lock_owner_address = tr2.resource_address;
```

Tracking Command Execution in SQL Server

Sometimes you want to track the commands that users are executing. You can do this by using Activity Monitor as follows:

1. In SQL Server Management Studio, use the Object Explorer view to access an instance of the Database Engine.
2. Right-click the Database Engine instance and then select Activity Monitor.
3. Expand the Processes panel by clicking Options. The entries in the Session ID, User Process, and Login columns can help you track user sessions and the processes they are using.
4. Right-click a process and then select Details to display the Session Details dialog box, as shown in Figure 1-10. This dialog box shows the last command batch executed by the user.


Session Details - CPANDL\wstanek - 53 - DBSERVER47
Last Transact-SQL command batch
Kill Process
Refresh
Close
Help

```
SET NOCOUNT ON;

DECLARE @previous_collection_time datetime;
DECLARE @previous_request_count bigint;
DECLARE @current_collection_time datetime;
DECLARE @current_request_count bigint;
DECLARE @batch_requests_per_sec bigint;
DECLARE @interval_sec bigint;
```

FIGURE 1-10 The Session Details dialog box displays the user's most recent command batch.

5. To track current commands being executed by the user, click Refresh periodically.
6. To end the process, click Kill Process. Then, when prompted, choose Yes.

Killing Server Processes

Sometimes you might need to stop processes that are blocking connections or using too much CPU time. To do this, complete the following steps:

1. In SQL Server Management Studio, use the Object Explorer view to access an instance of the Database Engine.
2. Right-click the Database Engine instance and then select Activity Monitor.
3. Expand the Processes panel by clicking Options.
4. Right-click the process you want to stop, and then choose Kill Process. When prompted to confirm, click Yes.

NOTE **Rather than kill a process in SQL Server, you may want to stop the application that is connected to the process and restart it to check if the issue is resolved.**

CHAPTER 3

Implementing Policy-Based Management

Policy-Based Management is an extensible and scalable configuration framework that you can use to manage servers, databases, and other objects in your data environments. As an administrator, you need to be very familiar with how Policy-Based Management technology works, and that's exactly what this chapter is about. If you haven't worked with Policy-Based Management technology before, one thing you'll notice immediately is that the technology is fairly advanced and has many features. To help you manage this complex technology, I'll start with an overview of Policy-Based Management and then explore its components.

Introducing Policy-Based Management

Just about every administrative task you perform can be affected by the policy-based framework in some way. The policy-based framework provides the ability to define policies that apply to server instances, databases, and other objects in your data environments. You use these policies to help you control and manage the configuration of data services throughout the enterprise. Through intelligent monitoring and proactive responses, you can prevent changes that deviate from the configurations you specify and want. You also can scale management across multiple servers, which makes enforcing consistent configuration policies easier.

Within the policy-based framework, you use the following objects to configure policy management:

- **Facet** Defines a management area within the policy-based framework. Each management area has a fixed set of related properties that you can configure. For example, the Backup Device facet has the following properties: BackupDeviceType, Name, PhysicalLocation, and SkipTapeLabel.
- **Condition** Defines the permitted states for one or more properties of a single facet. For example, you can create a condition called Limit Backup Devices to specify that for the Backup Device facet, BackupDeviceType can be set to hard disk or tape and SkipTapeLabel should always be set to True.
- **Policy** Contains a single condition that you want to enforce. For example, you can create a policy named Standard Backup Device Policy that assigns the Limit Backup Devices condition.
- **Category** Contains one or more policies that you want to enforce together. For example, you can create a category named Standard DB Policies that contains all the standard policies that you want to enforce within your Microsoft SQL Server databases.
- **Target** Defines the servers, databases, or other database objects to which policies are applied. For example, a target set could include all the databases on an instance of SQL Server.

MORE INFO **Put another way, policy-based management is explicit declarative management that you configure using facets, conditions, categories, and targets. *Facets* are fixed lists of things you can set policy on. *Conditions* determine when policy applies. *Categories* group policies together for enforcement. *Targets* determine to which objects policies apply.**

You can create and manage policies in several ways. In SQL Server Management Studio, you can create policies from scratch or import existing policy files. The policy creation process includes the following steps:

1. Select a facet that contains the properties you want to configure.
2. Define a condition that specifies the permitted states of the facet.
3. Define a policy that contains the condition and sets one of the evaluation modes listed in Table 3-1.
4. Determine whether an instance of SQL Server is in compliance with the policy, and then take appropriate action.

For scripting, the Microsoft.SqlServer.Management.Dmf namespace contains classes that represent policy-based management objects. You use the root of this namespace, the PolicyStore class, to work with policies. Consider the following example:

```
$comp = get-content c:\data\servers.txt
```

```
$cn = New-Object Microsoft.SqlServer.Management.Sdk.Sfc.SqlStoreConnection
("server='DbServer85';Trusted_Connection=True")

$ps = new-object Microsoft.SQLServer.Management.DMF.PolicyStore($cn)

foreach ($c in $comp) { foreach ($p in $ps.Policies) {
 #Invoke-PolicyEvaluation  }
}
```

Here, you get a list of servers that you want to work with from a text file and then configure a connection to SQL Server. Once you're connected to SQL Server, you access the policy store and work with the policies on each server in your list.

TABLE 3-1 Evaluation Modes for Policy-Based Management

POLICY EVALUATION MODE	DESCRIPTION	EXECUTION TYPE
On Demand	Evaluates the policy only when you directly execute the policy. Also referred to as *ad hoc* policy evaluation.	Manual
On Change: Log Only	Evaluates a policy when a relevant change is made and logs policy violations in the event logs.	Automatic
On Change: Prevent	When nested triggers are enabled, uses data definition language (DDL) triggers to prevent policy violations by detecting changes that violate a policy and rolling them back.	Automatic
On Schedule	Uses SQL Server Agent jobs to evaluate policies periodically. Logs policy violations in the event logs and generates a report.	Automatic

NOTE **All facets support the On Demand and On Schedule modes. Facets support the On Change: Log Only mode only if the change of the facet state can be captured by related system events. Facets support the On Change: Prevent mode only if there is transactional support for the DDL statements that change the facet state. Only automatic policies can be enabled or disabled.**

Policy categories apply to databases and servers. At the database level, database owners can subscribe a database to a set of policy categories, and those policies govern the database. By default, all databases implicitly subscribe to the default policy category. At the server level, you can apply policy categories to all databases.

You can mark categories as Active or Inactive at the server or database level. Although you can classify policies into different policy categories, a policy can belong only to one policy category.

All objects defined on a SQL Server instance form a target hierarchy. Within a policy, you define a target when you apply filters to the target hierarchy. For example, a target set with a large scope could include all the databases on an instance of SQL Server, while a target set with a small scope could include only the tables and indexes owned by the Sales schema in the Customers database.

The effective policies of a target are those policies that govern the target. For a policy to be effective with regard to a target, the policy must be enabled and the target must be subject to the policy. Within your data services environments, you enforce Policy-Based Management by using configuration servers. A designated configuration server is responsible for monitoring and enforcing policies as assigned. By default, each instance of SQL Server acts as its own configuration server. This means that each SQL Server instance normally handles its own policy monitoring and enforcement.

REAL WORLD **To be notified when messages from automatically executed policies are written to the event logs, you can create alerts to detect these messages and perform necessary actions. The alerts should detect the messages according to their message number. Look for message numbers 34050, 34051, 34052, and 34053. You can configure alerts as discussed in the "Managing Alerts" section in Chapter 10, "Automating and Maintaining SQL Server 2012."**

When policies are executed automatically, they execute as a member of the sysadmin role. This allows the policy to write entries to the event logs and raise an alert. When policies are evaluated on demand, they execute in the security context of the current user. To write to the event log, the user must have ALTER TRACE permissions or be a member of the sysadmin fixed server role; otherwise, Windows will not write to the event log and will not fire an alert.

Working with Policy-Based Management

You must be a member of the PolicyAdministratorRole role in the *msdb* database to configure Policy-Based Management settings. This role has complete control of all policies and can create policies and conditions, edit policies and conditions, and enable or disable policies.

When working with policies, keep the following in mind:

- A system administrator or database owner can subscribe a database to a policy or policy group.
- On-demand policy execution occurs in the security context of the user.
- Members of the PolicyAdministratorRole role can create policies that they do not have permission to execute on an ad hoc basis.

- Members of the PolicyAdministratorRole role can enable or disable policies.
- Policies that are in the On Schedule mode use SQL Server Agent jobs that are owned by the sa login.

Although you can manage policies for each instance of SQL Server, you'll likely reuse policies you've defined and then apply them to other instances of SQL Server. With Policy-Based Management, you can apply policies to multiple instances of SQL Server in several ways. As discussed in the "Importing and Exporting Policies" section later in this chapter, you can export the policies you've defined on a particular instance of SQL Server and then import the policies on another instance of SQL Server. During the import process, you can specify whether policies are enabled or disabled and whether to preserve the exported state of the policies.

Being able to export and import policies is useful. However, you don't necessarily need to move policies around to enforce the policies on multiple computers running SQL Server. Instead, you can manage policies by using a central management server. A central management server is a special type of configuration server that is responsible for monitoring and enforcing policy on any instance of SQL Server registered as a subordinate server. As discussed in the "Configuring Central Management Servers" section later in this chapter, you designate central management servers and their subordinates by using SQL Server Management Studio. Because the central management architecture is already an execution environment for Transact-SQL (T-SQL) statements related to policies, you can execute T-SQL statements on multiple instances of SQL Server at the same time from a central management server.

REAL WORLD **Generally, SQL Server does more validation in the graphical user interface (GUI) than in the application programming interface (API). As a result, you may be allowed to create a policy in T-SQL but be restricted from creating the same policy in the GUI. Why? Because SQL Server tries to prevent you from creating policies that might impact performance when working in the GUI, while making it possible for advanced users to be able to work around this.**

Because SQL Server stores policy-related data in the *msdb* database, you should back up *msdb* after you change conditions, policies, or categories. Policy history for policies evaluated in the current instance of the Database Engine is maintained in *msdb* system tables. Policy history for policies applied to other instances of the Database Engine or applied to Reporting Services or Analysis Services is not retained.

As summarized in Table 3-2, SQL Server 2012 includes several sets of predefined policies, including those for the Database Engine, Analysis Services, and Reporting Services. By default, the policies are stored as XML files in the following locations, and you must import them into SQL Server:

- Microsoft SQL Server\110\Tools\Policies\DatabaseEngine\1033
- Microsoft SQL Server\110\Tools\Policies\AnalysisServices\1033
- Microsoft SQL Server\110\Tools\Policies\ReportingServices\1033

NOTE **On 64-bit computers, policies are located under Program Files (x86) rather than Program Files when you install SQL Server in the default file system location. Surface area configuration is discussed in the "Managing SQL Server Component Feature Access" section in Chapter 2, "Managing SQL Server Services and Clients."**

TABLE 3-2 Important Predefined Policies for SQL Server 2012

PREDEFINED POLICY NAME	DESCRIPTION
Asymmetric Key Encryption Algorithm	Checks whether asymmetric keys were created by using 1024-bit or stronger encryption. As a best practice, you should use RSA 1024-bit or stronger encryption to create asymmetric keys for data encryption.
Backup And Data File Location	Checks whether database files are on devices separate from the backup files. As a best practice, you should put the database and backups on separate backup devices. This approach helps safeguard the data in case of device failure and also optimizes the I/O performance for both the production use of the database and the writing of backups.
CmdExec Rights Secured	Checks an instance of SQL Server 2000 to determine whether only members of the sysadmin server role can run CmdExec and ActiveX Script job steps, which is a recommended best practice.
Data And Log File Location	Checks whether data and log files are placed on separate logical drives. As a best practice, placing the files on separate drives allows the I/O activity to occur at the same time for both the data and log files.
Database Auto Close	Checks whether the AUTO_CLOSE option is set to OFF. When AUTO_CLOSE is set to ON, this option can cause performance degradation on frequently accessed databases because of the increased overhead of opening and closing the database after each connection. AUTO_CLOSE also flushes the procedure cache after each connection. As a best practice, you should set the AUTO_CLOSE option to OFF on a database that is accessed frequently.
Database Auto Shrink	Checks whether the AUTO_SHRINK database option is set to OFF. Because frequently shrinking and expanding a database can lead to fragmentation on the storage device, you should set the AUTO_SHRINK database option to OFF in most instances.

PREDEFINED POLICY NAME	DESCRIPTION
Database Collation	Checks whether user-defined databases are defined using a database collation that is the same as the collation for the *master* and *model* databases, which is a recommended best practice. Otherwise, collation conflicts can occur that might prevent code from executing. You can resolve collation conflicts by exporting the data from the user database, importing it into new tables that have the same collation as the *master* and *model* databases, and then rebuilding the system databases to use a collation that matches the user database collation. Or you can modify any stored procedures that join user tables to tables in *tempdb* to create the tables in *tempdb* by using the collation of the user database.
Database Page Status	Checks for user databases that have the database status set to Suspect. The Database Engine marks a database as Suspect when it reads a database page that contains an 824 error. Error 824 indicates that a logical consistency error was detected during a read operation, and it frequently indicates data corruption caused by a faulty I/O subsystem component. Resolve this situation by running DBCC CHECKDB.
Database Page Verification	Checks whether the PAGE_VERIFY database option is set to CHECKSUM. This recommended best practice helps provide a high level of data-file integrity by forcing the Database Engine to calculate a checksum over the contents of the whole page and store the value in the page header when a page is written to disk. When the page is read from disk, the checksum is recomputed and compared to the checksum value that is stored in the page header.
Guest Permissions	Checks whether the Guest user has permission to access a user database. As a best practice, you should revoke the Guest user permission to access non-system databases if it is not required. Although the Guest user cannot be dropped, the Guest user can be disabled by revoking its CONNECT permission. Execute REVOKE CONNECT FROM GUEST within any database other than *master* or *tempdb*.
Last Successful Backup Date	Checks to ensure that a database has recent backups. Scheduling regular backups protects a database against data loss. If there are no recent backups, you should schedule backups by using a database maintenance plan.

PREDEFINED POLICY NAME	DESCRIPTION
Public Not Granted Server Permissions	Checks whether the public server role has server permissions. Every login that is created on the server is a member of the public server role and has server permissions. As a best practice, however, do not grant server permissions directly to the public server role.
Read-Only Database Recovery Model	Checks for read-only user databases that have recovery set to Full. As a best practice, these databases should use the Simple recovery model because they aren't updated regularly.
SQL Server 32-Bit Affinity Mask Overlap	Checks whether the 32-bit instance of SQL Server has one or more processors that are assigned to be used with both the Affinity Mask and the Affinity I/O Mask options. Enabling a CPU with both these options can slow performance by forcing the processor to be overused.
SQL Server 64-Bit Affinity Mask Overlap	Checks whether the 64-bit instance of SQL Server has one or more processors that are assigned to be used with both the Affinity Mask and the Affinity I/O Mask options. Enabling a CPU with both these options can slow performance by forcing the processor to be overused.
SQL Server Affinity Mask	Checks whether the Affinity Mask option is set to 0. This is the default value, which dynamically controls CPU affinity. Using the default value is a recommended best practice.
SQL Server Blocked Process Threshold	Checks the Blocked Process Threshold option and ensures that it is set to 0 (disabled) or to a value higher than or equal to 5 seconds. Setting the Blocked Process Threshold option to a value from 1 through 4 can cause the deadlock monitor to run constantly, and this state is desirable only when you are troubleshooting.
SQL Server Default Trace	Determines whether the Default Trace option is disabled. When this option is enabled, default tracing provides information about configuration and DDL changes to the SQL Server Database Engine.
SQL Server Dynamic Locks	Checks whether the Locks option is set to 0. This is the default value, which dynamically controls locks. Using the default value is a recommended best practice. If the maximum number of locks is reached, batch jobs stop and SQL Server generates "out of locks" errors.

PREDEFINED POLICY NAME	DESCRIPTION
SQL Server I_O Affinity Mask for Non-Enterprise Servers	Checks whether the IO Affinity Mask option is set to 0 for editions of SQL Server other than Enterprise. With this value, SQL Server disk I/O is scheduled to any of the CPUs eligible to process SQL Server threads.
SQL Server Lightweight Pooling	Checks whether the Lightweight Pooling option is set to 0. This is the default value, which prevents SQL Server from using lightweight pooling. Using the default value is a recommended best practice.
SQL Server Login Mode	Checks the login security configuration to ensure Windows authentication is being used. Using Windows authentication is a recommended best practice because this mode uses the Kerberos security protocol, provides support for account lockout, and supports password expiration. For Windows Server 2008, Windows authentication also provides password policy enforcement in terms of complexity validation for strong passwords.
SQL Server Max Degree Of Parallelism	Checks whether the Max Degree Of Parallelism (MAXDOP) option is set to a value greater than 8. Setting this option to a value greater than 8 often causes unwanted resource consumption and performance degradation, so you usually want to reduce the value to 8 or less.
SQL Server Max Worker Threads For SQL Server 2005 And Above	Checks the Max Worker Threads option for potentially incorrect settings. Setting the Max Worker Threads option to a small value might prevent enough threads from servicing incoming client requests in a timely manner. Setting the option to a large value can waste address space because each active thread consumes 512 kilobytes (KB) on 32-bit servers and up to 4 megabytes (MB) on 64-bit servers. For instances of SQL Server 2005 and SQL Server 2012, you should set this option to 0, which allows SQL Server to determine the correct number of active worker threads automatically based on user requests.
SQL Server Network Packet Size	Determines whether the network packet size of any logged-in user is more than 8,060 bytes. As a best practice, the network packet size should not exceed 8,060 bytes. Otherwise, SQL Server performs different memory allocation operations, and this can cause an increase in the virtual address space that is not reserved for the buffer pool.

PREDEFINED POLICY NAME	DESCRIPTION
SQL Server Password Expiration	Checks whether password expiration is enabled for each SQL Server login. As a best practice, you should use ALTER LOGIN to enable password expiration for all SQL Server logins. Additionally, if SQL Server authentication is not required in your environment, you should enable only Windows authentication.
SQL Server Password Policy	Checks whether the Enforce Password policy is enabled for each SQL Server login. As a best practice, you should enable the Enforce Password policy for all the SQL Server logins by using ALTER LOGIN.
SQL Server System Tables Updatable	Checks whether system tables for SQL Server 2000 can be updated. As a best practice, you shouldn't allow updates to system tables.
Surface Area Configuration for Database Engine ...	A set of related policies for determining whether various editions of SQL Server are using default surface area settings. By disabling unneeded features, you can enhance security.
Symmetric Key Encryption For User Databases	Checks whether encryption keys that have a length of less than 128 bytes do not use the RC2 or RC4 encryption algorithm. As a best practice, you should use AES 128 bit or larger to create symmetric keys for data encryption. If AES is not supported by your operating system, you should use 3DES encryption.
Symmetric Key For *master* Database	Checks for user-created symmetric keys in the *master* database.
Symmetric Key For System Databases	Checks for user-created symmetric keys in the *model, msdb,* and *tempdb* databases. As a best practice, you should not create symmetric keys in the system databases.
Trustworthy Database	Checks whether the dbo role for a database is assigned to the sysadmin fixed server role and the database has its trustworthy bit set to ON. As a best practice, you should turn off the trustworthy bit or revoke sysadmin permissions from the dbo database role. Otherwise, a privileged database user can elevate privileges to the sysadmin role and then create and run unsafe assemblies that could compromise the system.

PREDEFINED POLICY NAME	DESCRIPTION
Windows Event Log Cluster Disk Resource Corruption Error	Checks the system event log for EventId 1066. This error can occur when a device is malfunctioning and also as a result of small computer system interface (SCSI) host adapter configuration issues.
Windows Event Log Device Driver Control Error	Checks the system event log for EventId 11. This error can be caused by a corrupt device driver, a hardware problem, faulty cabling, or connectivity issues.
Windows Event Log Device Not Ready Error	Checks the system event log for EventId 15. This error can be caused by SCSI host adapter configuration issues or related problems.
Windows Event Log Disk Defragmentation	Checks the system event log for EventId 55. This error occurs when the Disk Defragmenter tool cannot move a particular data element, and as a result Chkdsk.exe is scheduled to run.
Windows Event Log Failed I_O Request Error	Checks the system event log for EventId 50. This error is caused by a failed I/O request.
Windows Event Log I_O Delay Warning	Checks the event log for error message 833. This message indicates that SQL Server has issued a read or write request from disk and that the request has taken longer than 15 seconds to return. You can troubleshoot this error by examining the system event log for hardware-related error messages. Look also for hardware-specific logs.
Windows Event Log I_O Error During Hard Page Fault Error	Checks the system event log for EventId 51. This error is caused by an error during a hard page fault.
Windows Event Log Read Retry Error	Checks the event log for SQL Server error message 825. This message indicates that SQL Server was unable to read data from the disk on the first try. You need to check the disks, disk controllers, array cards, and disk drivers.
Windows Event Log Storage System I_O Timeout Error	Checks the system event log for EventId 9. This message indicates that an I/O time-out has occurred in the storage system.
Windows Event Log System Failure Error	Checks the system event log for EventId 6008. This event indicates an unexpected system shutdown.

REAL WORLD **Guest Permissions policy is meant for non-system databases only. Disabling guest access doesn't apply to *master, msdb,* and *tempb* system databases. Some SQL Server features require the guest user to be enabled in the *msdb* database. If guest access is disabled in *msdb,* you may receive error 916 when you try to connect to SQL Server using SQL Server Management Studio or other applications.**

Configuring Central Management Servers

By default, each instance of SQL Server is responsible for monitoring and enforcing its own policies. Although this configuration is useful in stand-alone deployments, you often want a more robust solution in the enterprise, and this is where central management servers are useful. Central management servers take over the responsibility of monitoring and enforcing policies from any instance of SQL Server registered as a subordinate server. From a central management server, you also can execute T-SQL statements on multiple instances of SQL Server simultaneously.

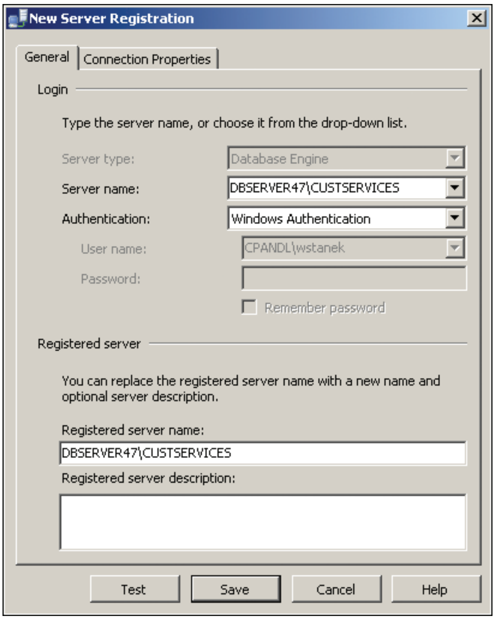
You can specify a SQL Server instance that you want to use as a central management server by registering the server in the Registered Servers view. Afterward, you can specify and register the subordinate servers that you will manage via the central management server. Although you must register subordinate servers individually, you can manage subordinate servers collectively by using subordinate server groups.

NOTE **Because SQL Server relies on Windows authentication to establish connections to registered servers, you must register all central management servers and subordinate servers by specifying Windows authentication during the registration process. Only members of the ServerGroupAdministratorRole role can manage the central management server. Membership in the ServerGroupReaderRole role is required to connect to a central management server.**

Registering Central Management Servers

A central management server cannot be a subordinate server or a member of a subordinate group that it maintains. You can register a central management server by following these steps:

1. In SQL Server Management Studio, use the Registered Servers view to work with central management servers. To use this view, or to display it if it is hidden, press Ctrl+Alt+G.
2. Under the Central Management Servers node, you'll see a list of previously registered central management servers. To register a new server, right-click the Central Management Servers node, and then select Register Central Management Server. This displays the New Server Registration dialog box, shown in Figure 3-1.


New Server Registration

General | Connection Properties

Login

Type the server name, or choose it from the drop-down list.

Server type: Database Engine
Server name: DBSERVER47\CUSTSERVICES
Authentication: Windows Authentication
User name: CPANDL\wstanek
Password:
Remember password

Registered server

You can replace the registered server name with a new name and optional server description.

Registered server name: DBSERVER47\CUSTSERVICES
Registered server description:

FIGURE 3-1 The New Server Registration dialog box.

3. In the Server Name box, type the fully qualified domain name (FQDN) or host name of the central management server, such as dbsvr23.cpandl.com or DBSvr23.
4. Choose Windows Authentication as the authentication type.
5. The registered server name is filled in for you on the basis of the server name you entered previously. Change the default name only if you want SQL Server Management Studio to use an alternate display name for the server.
6. To test the settings, click Test. If you successfully connect to the server, you will see a prompt confirming this. If the test fails, verify the information you provided, make changes as necessary, and then test the settings again.
7. Click Save.

Registering Subordinate Servers and Groups

After you register a central management server, you can register subordinate servers and create subordinate server groups. While individual subordinate servers don't have to be organized into groups, you can group servers according to their business unit, geographic location, or purpose for easier management.

You create a subordinate server group by completing the following steps:

1. In the Registered Servers view, expand the Central Management Servers node. You'll see a list of previously registered central management servers.

2. Right-click the central management server that will have management responsibility for the subordinate server group, and then select New Server Group.
3. In the New Server Group Properties dialog box, type a name and description for the new group in the boxes provided. Click OK.

You register a subordinate server by following these steps:

1. In the Registered Servers view, expand the Central Management Servers node, and then expand the node for the server that will have management responsibility for the subordinate server.
2. Right-click the server or one of its subordinate groups, and then select New Server Registration.
3. In the Server Name box, type the FQDN or host name of the subordinate server, such as DatabaseServer12.cpandl.com or DatabaseServer12.
4. Choose Windows Authentication as the authentication type.
5. The registered server name is filled in for you on the basis of the previously entered server name. Change the default name only if you want SQL Server Management Studio to use an alternative display name for the server.
6. To test the settings, click Test. If you successfully connect to the server, you will see a prompt confirming this. If the test fails, verify the information you provided, make changes as necessary, and then test the settings again.
7. Click Save.

Moving Subordinate Servers and Server Groups

Sometimes, you need to move a subordinate server or server group to a new location in the central management server hierarchy. You can do this by completing the following steps:

1. In the Registered Servers view, expand the Central Management Servers node and the related server and group nodes as necessary.
2. Right-click the subordinate server or server group that you want to move, point to Tasks, and then select Move To.
3. In the Move Server Registration dialog box, select the node into which you want to place the server or group.
4. Click OK.

The Move To process does not let you move a subordinate server or server group to a different central management server. To move a subordinate server or server group to a different central management server, you need to export the related registration settings and then import them to the new location. The export and import process works like this:

1. In the Registered Servers view, right-click the node with the settings to export, point to Tasks, and then select Export.

2. Click the options (...) button to the right of the Export File box.
3. Use the Save As dialog box to select a save location, type a name for the exported registered servers file, and then click Save.
4. Click OK to close the Export Registered Servers dialog box. If the file exists and you want to replace it with the current settings, click Yes when prompted.
5. Right-click the node where you want to import the settings, point to Tasks, and then select Import.
6. Click the options (...) button to the right of the Import File box.
7. Use the Open dialog box to navigate to the save location, select the exported registered servers file, and then click Open.
8. Click OK to close the Import Registered Servers dialog box. If identical subordinate groups and servers are already registered, you are prompted to replace the existing settings. Click Yes or Yes To All only if you want to overwrite existing settings.

Deleting Subordinate Servers and Server Groups

If you no longer use a server as a subordinate server or no longer want to use a server group, you can remove the entry for the server or server group. Right-click the server or server group, and then select Delete. When prompted to confirm, click Yes. When you delete a server group, SQL Server Management Studio removes the group and all the subordinate server registrations it contains.

Executing Statements Against Multiple Servers

You can query multiple servers at the same time by using central management servers. You can also execute T-SQL statements against local server groups in the Registered Servers view. Keep the following in mind:

- To query all subordinate servers for a central management server, right-click the central management server in the Registered Servers view and select New Query. In the Query Editor, type and execute your T-SQL statements.
- To query every server in a server group, right-click the server group in the Registered Servers view and select New Query. In the Query Editor, type and execute your T-SQL statements.

By default, the results pane combines the query results from all the servers. Because the connections to subordinate servers are executed using Windows authentication in the context of the currently logged-in user, the effective permissions might vary. If a connection cannot be established to one or more servers, those servers are ignored, and results for the other servers are displayed.

The combined results contain the server name but do not contain any login names. If you want, you can modify multiserver results by using the Options dialog box. Click Options on the Tools menu, expand Query Results and SQL Server, and then click Multiserver Results. On the Multiserver Results page, do one or more of the following, and then click OK:

- Configure the Add Login Names To The Results option. Use True to add login names to the results. Use False to remove login names from the results.
- Configure the Add Server Names To The Results option. Use True to add server names to the results. Use False to remove server names from the results.
- Configure the Merge Results option. Use True to merge results in a single results pane. Use False to display results in a separate pane for each server.

Managing Policies Throughout the Enterprise

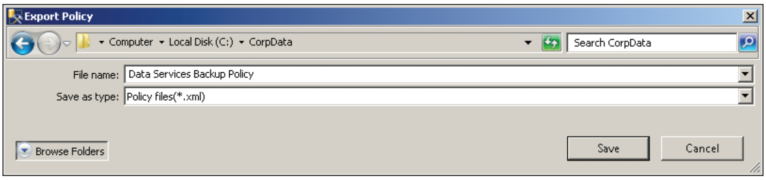
As discussed previously, you create and manage policies in SQL Server Management Studio. Implementing Policy-Based Management is a multistep process that involves selecting a facet that contains the properties you want to configure, defining a condition that specifies the permitted states of the facet, and defining a policy that contains the condition and sets an evaluation mode. The evaluation mode you set determines whether SQL Server uses automated policy monitoring, reporting, and compliance enforcement.

Importing and Exporting Policies

SQL Server 2012 includes predefined policies for the Database Engine, Analysis Services, and Reporting Services. You can import these predefined policies with their preconfigured conditions if you want to use them with a particular instance of the Database Engine, Analysis Services, or Reporting Services. If you create your own policies, you can export those policies with their conditions to XML files and then import the XML files to another instance of SQL Server.

You can export a policy by completing the following steps:

1. In SQL Server Management Studio, access the Management folder on the instance of the Database Engine you want to work with.
2. In Object Explorer, under the Management node, expand Policy Management and Policies. Right-click a policy, and then click Export Policy. This displays the Export Policy dialog box, shown in Figure 3-2.



File name: Data Services Backup Policy

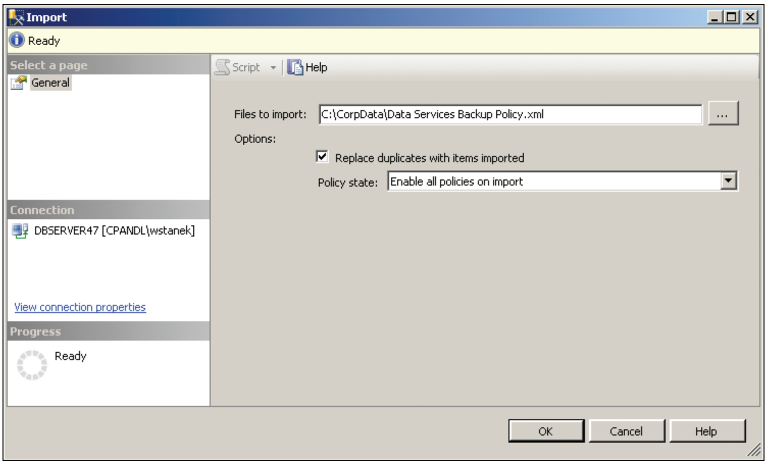
Save as type: Policy files(*.xml)

FIGURE 3-2 The Export Policy dialog box.

3. Use the options in the Export Policy dialog box to select a save location, and then type the name of the XML file.
4. Click Save. By default, SQL Server preserves the current state of the policy. This state will be set when the policy is imported. Note that any condition associated with the policy is exported as well.

You can import a policy by completing the following steps:

1. In SQL Server Management Studio, access the Management folder on the instance of the Database Engine you want to work with.
2. In Object Explorer, under the Management node, expand Policy Management, right-click Policies, and then click Import Policy.
3. In the Import dialog box, shown in Figure 3-3, click the options (...) button, and then use the Select Policy dialog box to locate the XML file that contains the policy. Select a policy by clicking it, and then click Open. You can select multiple files using Shift+click or Ctrl+click.



DBSERVER47 [CPANDL\wstanek]

Files to import: C:\CorpData\Data Services Backup Policy.xml

Options:

☑ Replace duplicates with items imported

Policy state: Enable all policies on import

FIGURE 3-3 The Import dialog box.

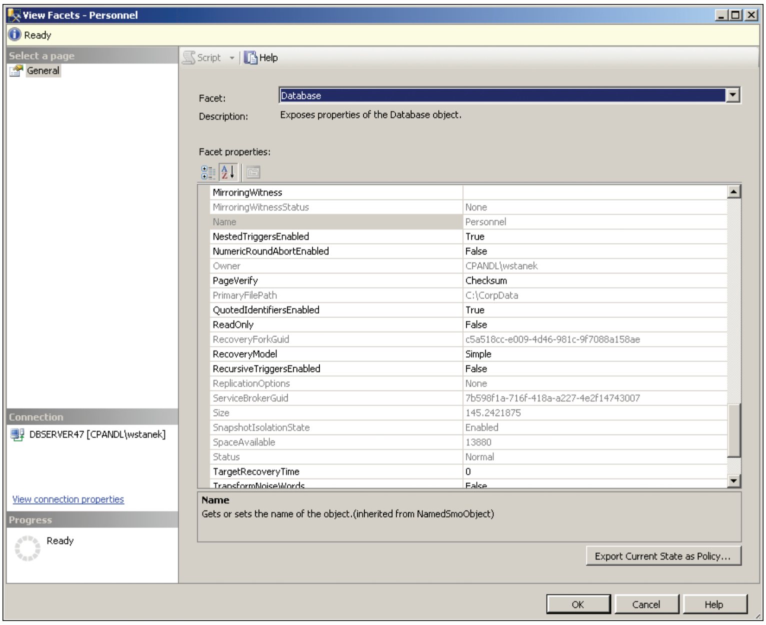
4. Remember that any condition associated with the policy is imported as well. If identically named policies (and conditions) already exist, the import process will fail with an error stating that the policies exist. To force SQL Server to overwrite existing policies, you must select the Replace Duplicates With Items Imported check box.
5. By default, SQL Server preserves the policy state on import. If a policy was enabled when it was exported, it will be enabled. If a policy was disabled when it was exported, it will be disabled. You can modify this behavior by explicitly setting the state. To enable the policies you are importing, select Enable All Policies On Import in the Policy State list. To disable the policies you are importing, select Disable All Policies On Import.
6. Click OK to begin the import process.

Configuring and Managing Policy Facets

Facets define management areas within the policy-based framework. Each management area has a set of related properties that you can configure by using a particular facet. You can view or modify the current state of any facet properties via the related object.

To view an object's current state and modify this state, follow these steps:

1. In Object Explorer, right-click a server instance, database, or database object, and then click Facets.
2. In the View Facets dialog box, shown in Figure 3-4, use the Facet list to select a facet related to the object. You then see a list of properties that shows the names and values of the facets.


View Facets - Personnel

Ready

Select a page
General

Connection
DBSERVER47 [CPANDL\wstanek]

View connection properties

Progress
Ready

Facet: Database

Description: Exposes properties of the Database object.

Facet properties:

MirroringWitness	
MirroringWitnessStatus	None
Name	Personnel
NestedTriggersEnabled	True
NumericRoundAbortEnabled	False
Owner	CPANDL\wstanek
PageVerify	Checksum
PrimaryFilePath	C:\CorpData
QuotedIdentifiersEnabled	True
ReadOnly	False
RecoveryForkGuid	c5a518cc-e009-4d46-981c-9f7088a158ae
RecoveryModel	Simple
RecursiveTriggersEnabled	False
ReplicationOptions	None
ServiceBrokerGuid	7b598f1a-716f-418a-a227-4e2f14743007
Size	145.2421875
SnapshotIsolationState	Enabled
SpaceAvailable	13880
Status	Normal
TargetRecoveryTime	0
TransformNoiseWords	False

Name
Gets or sets the name of the object.(inherited from NamedSmoObject)

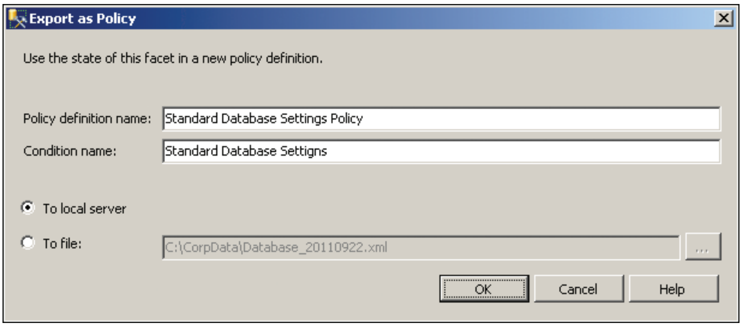
FIGURE 3-4 Modify property values as necessary.

3. Click in the box next to the property to select a property value. If a property appears dimmed, you cannot modify the property value.
4. Click OK.

Exporting an object's current state as policy allows you to use the current configuration of a server instance, database, or other database object to define policies that you want to implement throughout the enterprise. After you export an object's current state as policy, you can save the policy to the Policy Management\Policies node on the local server or to a policy file that you can import on another server.

Exporting an object's current state as policy creates a condition and a policy. To view an object's current state and export this state as policy, follow these steps:

1. In Object Explorer, right-click a server instance, database, or database object, and then click Facets.
2. In the View Facets dialog box, use the Facet list to select a facet related to the object. You then see a list of properties that shows the names and values of the facets.
3. Click Export Current State As Policy to display the Export As Policy dialog box, shown in Figure 3-5.



Use the state of this facet in a new policy definition.

Policy definition name: Standard Database Settings Policy

Condition name: Standard Database Settings

[✓ To local server]

To file: C:\CorpData\Database_20110922.xml

FIGURE 3-5 Export the property settings as a policy.

4. Type a name for the policy, and then type a name for the condition.
5. To save the policy to the Policy Management\Policies node on the local server, select To Local Server, and then click OK to close the Export As Policy dialog box. The related policy and condition are created under the appropriate nodes within Policy Management.
6. To save the policy to a file that you can import on another server, select To File. Click the options (...) button. Use the Export Policy dialog box to select a save location and name for the policy file, and then click Save. Click OK to close the Export As Policy dialog box. Later, you can import the policy using the technique discussed in the "Importing and Exporting Policies" section earlier in this chapter.

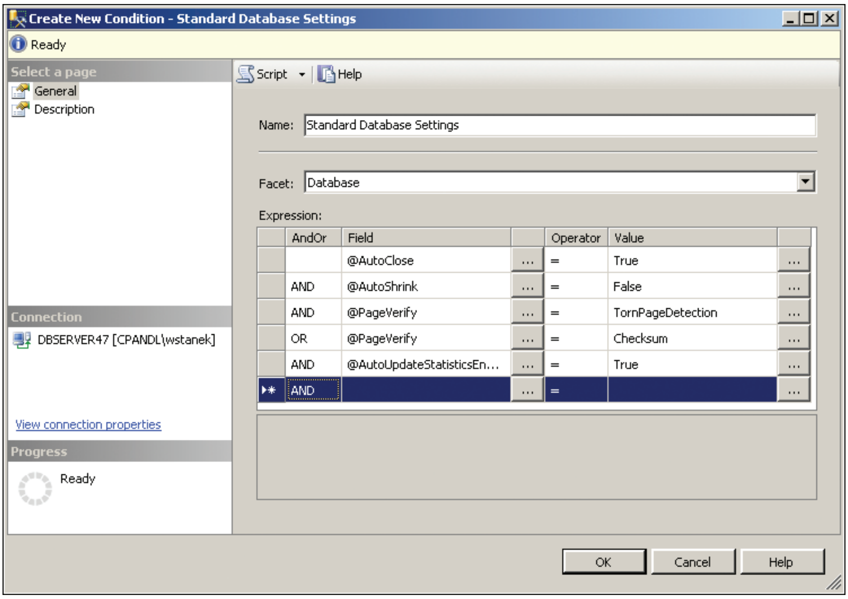
Creating and Managing Policy Conditions

Facets represent management areas within SQL Server. Most facets have multiple properties that you can manage or evaluate using conditions. Conditions define the permitted states for properties based on expressed values. Although you can use a single condition in multiple policies, you define conditions for each facet individually.

Defining Conditions Using Properties and Standard Expressions

When you are defining conditions, you join property evaluation expressions by using the And or Or clause to form a logical statement. For example, with the database facet, you might want to establish the condition shown in Figure 3-6. In this example, the evaluation expression specifies the following:

- AutoClose must be True,
- And AutoShrink must be False,
- And PageVerify must be set to either TornPageDetection or Checksum,
- And AutoUpdateStatisticsEnabled must be True,
- And Trustworthy must be True.


Create New Condition - Standard Database Settings

Ready

Name: Standard Database Settings

Facet: Database

Expression:

AndOr	Field	Operator	Value
	@AutoClose	=	True
AND	@AutoShrink	=	False
AND	@PageVerify	=	TornPageDetection
OR	@PageVerify	=	Checksum
AND	@AutoUpdateStatisticsEn...	=	True
AND		=	

Connection: DBSERVER47 [CPANDL\wstanek]

FIGURE 3-6 Define a condition by joining property expressions.

Although the allowed values depend on the property you are configuring, values generally can be numeric, a string, or a fixed list. With properties that are on (true) or off (false), you can set the operator to equals (=) or not equals (!=). When you set a property to equals, the property must equal the specified value. When you set a property to not equals, the property cannot equal the specified value, but it can be set to other permitted values.

With multivalued properties, other operators you can use are as follows:

- **>** Greater than; the property must be greater than the value specified.
- **>=** Greater than or equal to; the property must be greater than or equal to the value specified.

- **<** Less than; the property must be less than the value specified.
- **<=** Less than or equal to; the property must be less than or equal to the value specified.
- **Like** Pattern match, as with the LIKE clause in T-SQL; the property must match a specified pattern. Enclose the Like value in single quotation marks, such as '%computer%' or '[D-Z] arwin'.
- **Not Like** Pattern match, as with the NOT LIKE clause in T-SQL; the property must not match a specified pattern. Enclose the Not Like value in single quotation marks.
- **In** Query or list match, as with the IN clause for T-SQL; the property must match a value in the specified query or list. Enclose the In clause in parentheses, enclose individual values in single quotation marks, and separate values with commas, such as ('Hawaii', 'Idaho', 'Nebraska').
- **Not In** Query or list match, as with the NOT IN clause for T-SQL; the property must not match a value in the specified query or list.

Creating and Modifying Conditions

You can create a condition by completing the following steps:

1. In SQL Server Management Studio, access the Management folder on the instance of the Database Engine you want to work with.
2. In Object Explorer, under the Management node, expand Policy Management, expand Facets, right-click the facet that contains the properties that you want, and then click New Condition.
3. On the General page of the Create New Condition dialog box, type the name of the new condition, such as Standard Database Settings, in the Name box.
4. Confirm that the correct facet is shown in the Facet box, or select a different facet.
5. In the Expression area, construct condition expressions by selecting a facet property in the Field box together with its associated operator and value. When you add multiple expressions, the expressions can be joined by using And or Or.
6. To create complex expressions, press the Shift or Ctrl key, and then click two or more clauses to select a range. Right-click the selected area, and then click Group Clauses. Grouping clauses is like putting parentheses around an expression in a mathematical expression, which forces the clauses to operate as a single unit that is separate from the rest of the condition.
7. Optionally, on the Description page, type a description for the new condition.
8. Click OK to create the condition.

After creating a condition, you can view or modify its settings by completing these steps:

1. In SQL Server Management Studio, access the Management folder on the instance of the Database Engine you want to work with.
2. In Object Explorer, under the Management node, expand Policy Management, expand Conditions, right-click the condition that you want to view or modify, and then select Properties.
3. View the condition settings on the General page. Make changes as necessary, and then click OK.

Although you cannot delete a condition referenced in a policy, you can delete unreferenced conditions. You delete a condition by right-clicking it and then selecting Delete. When prompted to confirm, click OK.

Defining Complex Expressions

Although standard expressions are useful for evaluating conditions, complex expressions give you many more options, including the capability to replace object and schema names at run time. Complex expressions do this by extending the valid syntax for expressions to include a set of predefined functions that can be evaluated on either side of the condition operator. Thus, the standard expression syntax of

```
{property|constant} [operator] {property|constant}
```

becomes

```
{property|constant|function} [operator] {property|constant|function}
```

Available functions allow you to perform many complex tasks. You can add values, count values, or compute averages, and then evaluate the results. You can convert values to strings or concatenate strings and then evaluate the results. You can execute T-SQL queries or WMI Query Language (WQL) scripts and then evaluate the results.

To create a condition that uses complex expressions, complete the following steps:

1. Create a new condition or open an existing condition for editing using the techniques discussed previously.
2. While working with the condition, on the General page, click the options (…) button in the Expression area.
3. In the Advanced Edit dialog box, define your function in the Cell Value box and then click OK.

TIP **Available functions and properties are listed under Functions And Properties. Click a property or function to display detailed usage information. Double-click a property to insert it into the cell value area at the current cursor position. Double-click a function to insert its basic syntax into the cell value area at the current cursor position.**

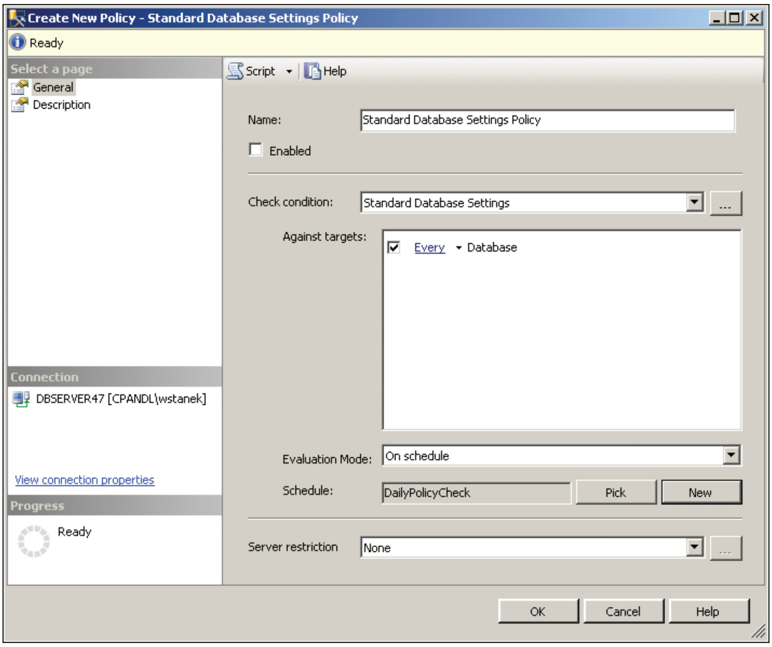
Creating and Managing Policies

You use policies to check and optionally enforce conditions. When you create a policy, you can use a condition that you created earlier, or you can create a new condition when you are creating the policy. Although you can use a particular condition in many policies, a policy can contain only a single condition.

When you create a policy, the policy normally is associated with the current instance of the Database Engine. If the current instance is a central management server, the policy can be applied to all subordinate servers. You also can directly create a policy by choosing New from the File menu and then saving the policy to a file. This enables you to create policies when you are not connected to the instance of the Database Engine that you want to work with.

You can create a policy and associate it with a particular instance of the Database Engine by completing the following steps:

1. In SQL Server Management Studio, access the Management folder on the instance of the Database Engine you want to work with.
2. In Object Explorer, under the Management node, expand Policy Management, right-click Policies, and then click New Policy. This displays the Create New Policy dialog box, shown in Figure 3-7.


Create New Policy - Standard Database Settings Policy

Ready

Name: Standard Database Settings Policy

Enabled: unchecked

Check condition: Standard Database Settings

Against targets: ✓ Every Database

Evaluation Mode: On schedule

Schedule: DailyPolicyCheck

Server restriction: None

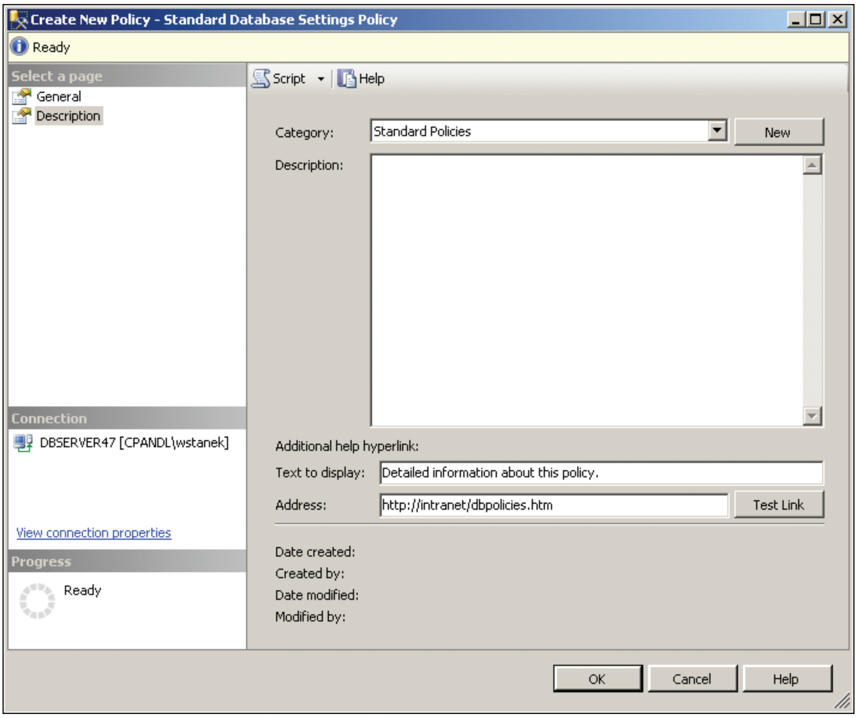
Connection: DBSERVER47 [CPANDL\wstanek]

Progress: Ready

FIGURE 3-7 Create the policy and specify the condition that applies.

3. In the Name box, type the name of the new policy, such as Standard Database Settings Policy.
4. Use the Check Condition list to select one of the existing conditions, or select New Condition. To edit a condition, select the condition, and then click the options (…) button. Make changes as necessary to the condition settings, and then click OK.
5. In the Against Targets box, select one or more target types for this policy. Some conditions and facets can be applied only to certain types of targets. The available target sets appear in the associated box. If no targets appear in this box, the check condition is scoped at the server level. To select a filtering condition for some types of targets, click the Every entry and then select an existing filter condition, or define a condition by selecting New Condition and then specifying the settings for the condition.
6. Use the Evaluation Mode list to select how this policy will behave. Different conditions can have different valid evaluation modes. Available modes can include On Demand, On Change: Prevent, On Change: Log Only, and On Schedule. If you specify a mode other than On Demand, you can enable the policy by selecting the Enabled check box. If you specify On Schedule as the mode, click Pick to select an existing run schedule, or click New to create a new schedule. For more information on creating schedules, see the "Configuring Job Schedules" section in Chapter 10.
7. To limit the policy to a subset of the target types, select a limiting condition in the Server Restriction list, or select New Condition to create a new condition.
8. By default, policies are assigned to the Default category. On the Description page, in the Category box, you can select a different default policy category. (See Figure 3-8.) Otherwise, to create a new category, click New, type a category name, and then click OK.
9. On the Description page, you can type an optional description of the policy. Click OK to create the policy.

REAL WORLD **To help administrators understand your policies, you can publish help documents on a website and then refer administrators to the help documentation by using a hyperlink. You can define a help hyperlink by using the Additional Help Hyperlink option on a policy's Description page. Enter the help text for the link in the Text To Display box, and then enter the hyperlink address in the Address box. You can provide a link to a webpage that starts with http:// or https://, or you can provide a mail link that starts with mailto://. After you type a hyperlink, click Test Link to check the validity of the hyperlink. When you are evaluating a server instance, database, or other object for policy compliance, the help text and link are displayed as part of the detailed results.**


Create New Policy - Standard Database Settings Policy

Ready

Category: Standard Policies

Description:

Additional help hyperlink:

Text to display: Detailed information about this policy.

Address: http://intranet/dbpolicies.htm

Date created:

Created by:

Date modified:

Modified by:

Connection: DBSERVER47 [CPANDL\wstanek]

Progress: Ready

FIGURE 3-8 Assign the policy to a category.

After creating a policy, you can view or modify its settings by completing these steps:

1. In SQL Server Management Studio, access the Management folder on the instance of the Database Engine you want to work with.
2. In Object Explorer, under the Management node, expand Policy Management, expand Policies, right-click the policy that you want to view or modify, and then select Properties.
3. View the policy settings on the General and Description pages. Make changes as necessary. Click OK.

You can manage policies you've created by using the following techniques:

- Delete a policy by right-clicking it and selecting Delete. When prompted to confirm, click OK.
- Disable a policy by right-clicking it and selecting Disable.
- Enable a policy by right-clicking it and selecting Enable.

Managing Policy Categories and Mandating Policies

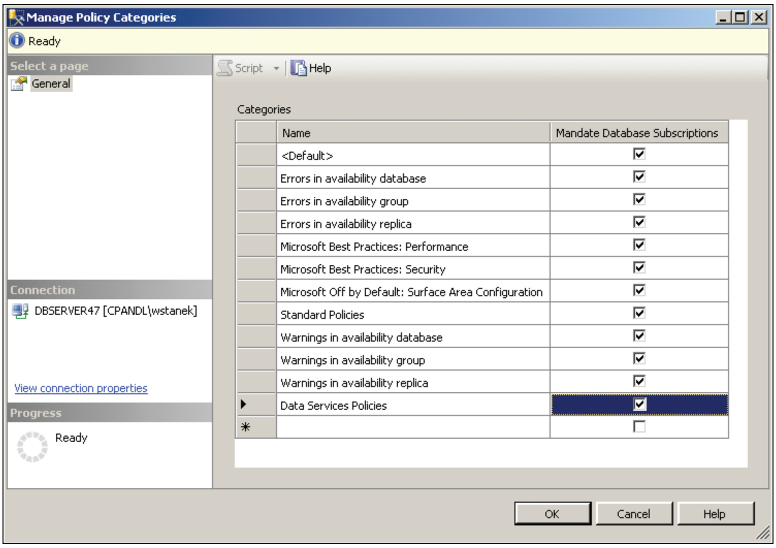
SQL Server 2012 uses policy categories to help you organize your policies. In a large organization, grouping policies into categories makes policy management easier. You can assign a policy to a category in several ways. By default, any policy you create belongs to the Default category. When you create a policy, you can assign the policy to an available category or to a new category as well. To move a policy to a different policy category, complete the following steps:

1. Right-click the policy that you want to view or modify, and then select Properties.
2. On the Description page, in the Category box, select a different default policy category. Otherwise, to create a new category, click New, type a category name, and then click OK.
3. Click OK to apply the changes.

In addition to helping you organize policies, policy categories help with policy application. Policies within categories are either mandated or not mandated. If a policy is mandated, it means that all databases on the instance of SQL Server must enforce the policy. If a policy is not mandated, it means that you must apply the policy manually as appropriate.

By default, any policy assigned to the Default category is a mandated policy. You can control whether a policy category and its related policies are mandated or not mandated by completing the following steps:

1. In SQL Server Management Studio, access the Management folder on the instance of the Database Engine you want to work with.
2. In Object Explorer, under the Management node, right-click the Policy Management node, and then select Manage Categories.
3. The available categories are listed by name. To create a new policy category, simply click in an empty text box in the Name column and type the category name.
4. In the Manage Policy Categories dialog box, shown in Figure 3-9, select or clear the Mandate Database Subscriptions check box for each category, as appropriate. Click OK.


Manage Policy Categories

Ready

Categories

	Name	Mandate Database Subscriptions
	<Default>	✓
	Errors in availability database	✓
	Errors in availability group	✓
	Errors in availability replica	✓
	Microsoft Best Practices: Performance	✓
	Microsoft Best Practices: Security	✓
	Microsoft Off by Default: Surface Area Configuration	✓
	Standard Policies	✓
	Warnings in availability database	✓
	Warnings in availability group	✓
	Warnings in availability replica	✓
▶	Data Services Policies	✓
*		☐

Connection

DBSERVER47 [CPANDL\wstanek]

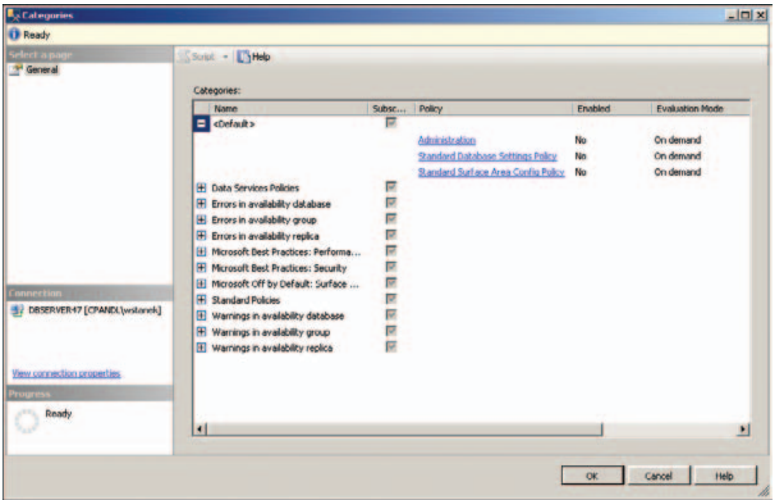
Progress

Ready

FIGURE 3-9 Specify whether policies within categories are mandated.

You can determine the policies that are applied to a database or other object by completing the following steps:

1. In Object Explorer, right-click a database or database object, point to Policies, and then click Categories.
2. In the Categories dialog box, expand the category entries to determine which policies are being applied. As shown in Figure 3-10, the following information is available:
   - **Name** Shows the name of the policy category.
   - **Subscribed** Indicates whether the selected object has subscribed to the policy category. If the related check box appears dimmed, the policy category is mandated and applies to all databases on the server.
   - **Policy** Shows the policies in the policy category, provided that you've expanded the related category node.
   - **Enabled** Indicates whether the policy is enabled or disabled.
   - **Evaluation Mode** Shows the evaluation mode of the policy.
   - **History** Click the View History link to open the Log File viewer and display the creation and change history for the policy.



Categories:

Name	Subsc...	Policy	Enabled	Evaluation Mode
<Default>	✓			
		Administration	No	On demand
		Standard Database Settings Policy	No	On demand
		Standard Surface Area Config Policy	No	On demand
Data Services Policies	✓			
Errors in availability database	✓			
Errors in availability group	✓			
Errors in availability replica	✓			
Microsoft Best Practices: Performa...	✓			
Microsoft Best Practices: Security	✓			
Microsoft Off by Default: Surface ...	✓			
Standard Policies	✓			
Warnings in availability database	✓			
Warnings in availability group	✓			
Warnings in availability replica	✓			

FIGURE 3-10 Determine which policies are being applied.

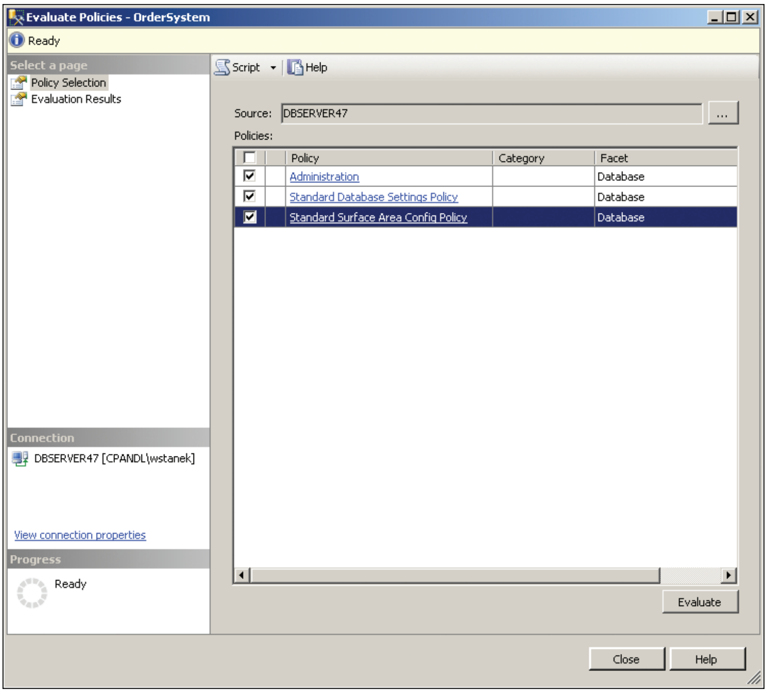
Evaluating Policies

By using automatically evaluated modes, you can check policy compliance when changes are made or on a regularly scheduled basis. You also can evaluate a policy manually to determine whether a server instance, database, or other object complies with the policy. If you later apply or enforce the policy, you can configure the selected database instance or database object to comply with the policy.

Because the connections to subordinate servers are executed using Windows authentication in the context of the currently logged-on user, the effective permissions might vary. If a connection cannot be established to one or more servers, those servers are ignored, and evaluation against the other servers continues independently.

You can determine whether a particular object complies with a policy by completing the following steps:

1. In Object Explorer, right-click a server instance, database, or database object, point to Policies, and then click Evaluate.
2. The Evaluate Policies dialog box shows only the policies that are appropriate for the object. (See Figure 3-11.) In the Evaluate Policies dialog box, select one or more policies, and then click Evaluate to run the policy in evaluation mode. (The evaluation mode is defined as part of the policy and cannot be changed in the Evaluate Policies dialog box.)


Evaluate Policies - OrderSystem

Ready

Select a page
- Policy Selection
- Evaluation Results

Script | Help

Source: DBSERVER47

Policies:

☐	Policy	Category	Facet
☑	Administration		Database
☑	Standard Database Settings Policy		Database
☑	Standard Surface Area Config Policy		Database

Connection

DBSERVER47 [CPANDL\wstanek]

View connection properties

Progress

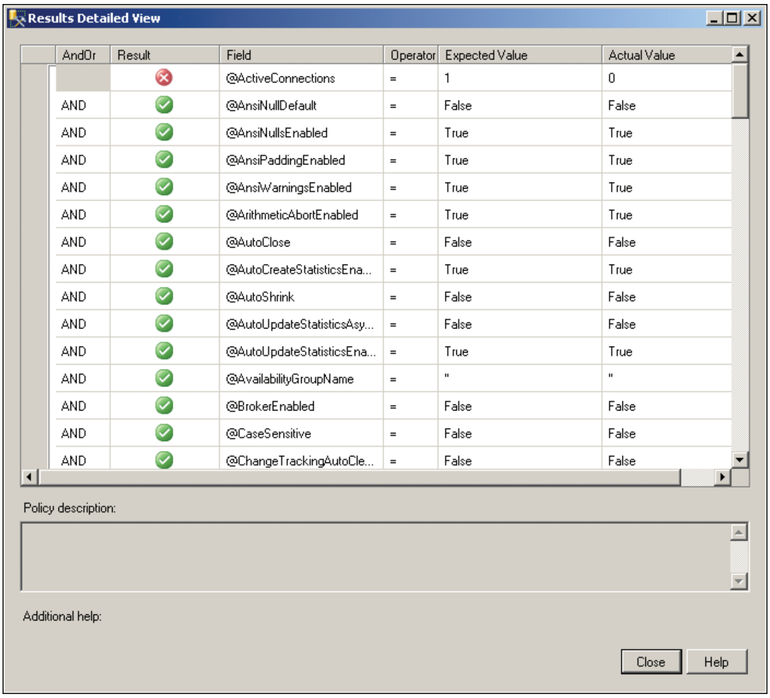
Ready

Evaluate

Close | Help

FIGURE 3-11 Evaluate an object against policies to determine compliance.

3. If there are compliance issues, you'll see a red warning icon. You can click the View link that appears in the Details column under Target Details to view the detailed compliance results. As shown in Figure 3-12, the Result column shows whether each property in the joined evaluation expression is in compliance or out of compliance. Expected and actual values are also shown. Note that help text is provided if it was previously defined. Click Close when you finish reviewing the detailed results.
4. In the Evaluate Policies dialog box, clicking Evaluate generates a compliance report for the target set but does not reconfigure SQL Server or enforce compliance. For targets that do not comply with the selected policies and have properties that can be reconfigured by Policy-Based Management, you can enforce policy compliance by selecting the policy or policies to apply on the Evaluation Results page and then clicking Apply.



Results Detailed View

AndOr	Result	Field	Operator	Expected Value	Actual Value
	✗	@ActiveConnections	=	1	0
AND	✓	@AnsiNullDefault	=	False	False
AND	✓	@AnsiNullsEnabled	=	True	True
AND	✓	@AnsiPaddingEnabled	=	True	True
AND	✓	@AnsiWarningsEnabled	=	True	True
AND	✓	@ArithmeticAbortEnabled	=	True	True
AND	✓	@AutoClose	=	False	False
AND	✓	@AutoCreateStatisticsEna...	=	True	True
AND	✓	@AutoShrink	=	False	False
AND	✓	@AutoUpdateStatisticsAsy...	=	False	False
AND	✓	@AutoUpdateStatisticsEna...	=	True	True
AND	✓	@AvailabilityGroupName	=	''	''
AND	✓	@BrokerEnabled	=	False	False
AND	✓	@CaseSensitive	=	False	False
AND	✓	@ChangeTrackingAutoCle...	=	False	False

Policy description:

Additional help:

FIGURE 3-12 Review compliance issues.

5. The first time you try to apply a policy, you'll see a Policy Evaluation Warning dialog box prompting you to confirm the action. Click Yes to proceed. If you don't want to see the warning in the future, select the Do Not Show This Message Again check box before clicking Yes.
6. After you apply a policy, you can review the detailed results by clicking the View link that appears in the Details column under Target Details. If the properties can be reconfigured using Policy-Based Management, the properties will be changed. Click Close when you finish reviewing the detailed results.
7. Optionally, you can export the results to a policy results file for later review. Click Export. Use the Export Results dialog box to select a save location for the results file, type a file name, and then click Save.

You can determine whether the targets of a policy are in compliance by completing the following steps:

1. In Object Explorer, expand Management, Policy Management, and Policies. Right-click a policy, and then click Evaluate.
2. Follow steps 2 through 7 in the previous procedure.

You can determine whether the targets of a policy are in compliance with a schedule by completing the following steps:

1. In Object Explorer, expand Management, Policy Management, and Policies. Right-click a policy, and then click Properties.
2. On the General page, specify On Schedule as the evaluation mode.
3. Click Pick to select an existing run schedule, or click New to create a schedule. Click OK twice to save your changes.

To view the history of compliance checks, right-click the policy and then select View History. In Log File Viewer, expand the available run dates to show additional details. Review the related details in the detailed view by clicking the link provided in the Details column.

Each property in the joined evaluation expression is listed according to whether it is in or out of compliance. If there are compliance issues, you'll see a red warning icon to show properties not in compliance or a green OK icon to show properties in compliance. Expected and actual values are also shown. Note that help text is provided if it was previously defined. Click Close when you finish reviewing the detailed results.

Troubleshooting Policies

In the *msdb* database, you'll find the following views for displaying policy information. These views are owned by the dbo schema:

- syspolicy_conditions
- syspolicy_policies
- syspolicy_policy_execution_history
- syspolicy_policy_execution_history_details
- syspolicy_policy_category_subscriptions
- syspolicy_policy_categories
- syspolicy_system_health_state
- syspolicy_target_sets

SQL Server records compliance issues in the Windows event logs. For scheduled policies, compliance issues are recorded in the SQL Server Agent log as well. To view the history information recorded in the SQL Server Agent logs, right-click the policy and then select View History. Review the related details in the detailed view by clicking the link provided in the Details column. If policies are not enabled or do not affect a target, the failure is not considered an error and is not logged. For more information, see the "Evaluating Policies" section earlier in this chapter.

Remember that compliance checks for scheduled policies occur only during scheduled run times and that on-demand policies run only when you manually execute them. If you have problems with policies set to On Change: Log or On Change: Prevent, be sure that the policies are enabled and that the target you want is not excluded by a filter. You also should ensure the target subscribes to the policy group that contains the policy. As discussed in the "Managing Policy Categories and Mandating Policies" section earlier in this chapter, you can determine the policies that are applied to a database or other object by right-clicking a database or database object, pointing to Policies, and then clicking Categories.

You can determine whether a policy was evaluated by right-clicking the policy and then selecting View History. The policy execution history in the msdb.dbo.syspolicy_policy_execution_history view also provides information about whether a policy was evaluated. You can also determine whether the policy executed for the specific target by checking the policy execution history for the specific target in the msdb.dbo.syspolicy_policy_execution_history_details view. You can determine the execution time for policies by querying the start_date and end_date columns in the msdb.dbo.syspolicy_policy_execution_history view.

For policies that use the On Change: Prevent mode, Service Broker handles the rollback of changes. You should ensure that Service Broker is running and configured properly. If it is, you can check the Service Broker Queue to be sure that it is monitoring for the correct events by using either of the following queries:

T-SQL

```
SELECT * FROM sys.server_event_notifications
WHERE name = N'syspolicy_event_notification';
GO
```

PowerShell

```
Set-Location SQLSERVER:\SQL\DbServer18\OrderSystem
Invoke-Sqlcmd -Query "SELECT * FROM sys.server_event_notifications
WHERE name = N'syspolicy_event_notification';"
```

NOTE **In the Windows PowerShell example, you define the working server context by explicitly setting the location. This allows you to invoke SQL commands in this location and is the same as using –ServerInstance "DbServer18\OrderSystem".**

Keep in mind that if the nested triggers server configuration option is disabled, On Change: Prevent mode will not work correctly. Policy-Based Management relies on DDL triggers to detect and roll back DDL operations that do not comply with policies that use this evaluation mode. Removing the Policy-Based Management DDL triggers or disabling nested triggers causes this evaluation mode to fail.

Because On Schedule policies rely on SQL Server Agent jobs, you should always check to be sure that SQL Server Agent is running and configured properly. You also should check to ensure that the related SQL Server Agent jobs are enabled and configured properly. Working with SQL Server Agent jobs is discussed in Chapter 10.

If you suspect there is a problem with policy-related jobs, you may be able to use sp_syspolicy_repair_policy_automation to repair the jobs. This stored procedure will attempt to repair triggers and jobs that are associated with On Schedule or On Change policies. Since policies are stored in the *msdb* database, you must run this stored procedure in the context of the *msdb* database, as shown in the following example:

```
EXEC msdb.dbo.sp_syspolicy_repair_policy_automation;
GO
```

Index