

Table of Contents

SQL Server Failover Cluster Installation

- Before Installing Failover Clustering

- Create a New SQL Server Failover Cluster (Setup)

- Add or Remove Nodes in a SQL Server Failover Cluster (Setup)

- Install Client Tools on a SQL Server Failover Cluster

- Remove a SQL Server Failover Cluster Instance (Setup)

- Rename a SQL Server Failover Cluster Instance

High Availability Solutions

- Automatic Page Repair (Availability Groups: Database Mirroring)

- Management of Logins and Jobs After Role Switching

- Troubleshoot Orphaned Users

Windows Server Failover Clustering (WSFC) with SQL Server

- WSFC Quorum Modes and Voting Configuration

 - View Cluster Quorum NodeWeight Settings

 - Configure Cluster Quorum NodeWeight Settings

- WSFC Disaster Recovery through Forced Quorum

 - Force a WSFC Cluster to Start Without a Quorum

- SQL Server Multi-Subnet Clustering

- Failover Cluster Troubleshooting

Always On Failover Cluster Instances

- Failover Policy for Failover Cluster Instances

 - Configure HealthCheckTimeout Property Settings

 - Configure FailureConditionLevel Property Settings

 - View and Read Failover Cluster Instance Diagnostics Log

- Failover Cluster Instance Administration and Maintenance

 - Add Dependencies to a SQL Server Resource

 - Recover from Failover Cluster Instance Failure

 - Change the IP Address of a Failover Cluster Instance

- Upgrade a SQL Server Failover Cluster Instance

Upgrade a SQL Server Failover Cluster Instance (Setup)

SQL Server Failover Cluster Installation

3/29/2017 • 3 min to read • [Edit Online](#)

To install a SQL Server failover cluster, you must create and configure a failover cluster instance by running SQL Server Setup.

Installing a Failover Cluster

To install a failover cluster, you must use a domain account with local administrator rights, permission to log on as a service, and to act as part of the operating system on all nodes in the failover cluster. To install a failover cluster by using the SQL Server Setup program, follow these steps:

1. To install, configure, and maintain a SQL Server failover cluster, use SQL Server Setup.
 - Identify the information you need to create your failover cluster instance (for example, cluster disk resource, IP addresses, and network name) and the nodes available for failover. For more information:
 - [Before Installing Failover Clustering](#)
 - [Security Considerations for a SQL Server Installation](#)
 - The configuration steps must take place before you run the SQL Server Setup program; use the Windows Cluster Administrator to carry them out. You must have one WSFC group for each failover cluster instance you want to configure.
 - You must ensure that your system meets minimum requirements. For more information on specific requirements for a SQL Server failover cluster, see [Before Installing Failover Clustering](#).
2. Add or remove nodes from a failover cluster configuration without affecting the other cluster nodes. For more information, see [Add or Remove Nodes in a SQL Server Failover Cluster \(Setup\)](#).
 - All nodes in a failover cluster must be of the same platform, either 32-bit or 64-bit, and must run the same operating system edition and version. Also, 64-bit SQL Server editions must be installed on 64-bit hardware running the 64-bit versions of Windows operating systems. There is no WOW64 support for failover clustering in this release.
3. Specify multiple IP addresses for each failover cluster instance. You can specify multiple IP addresses for each subnet. If the multiple IP addresses are on the same subnet, SQL Server Setup sets the dependency to AND. If you are clustering nodes across multiple subnets, SQL Server Setup sets the dependency to OR.

SQL Server Failover Cluster Installation options

Option 1: Integrated installation with Add Node

SQL Server integrated failover cluster installation consists of two steps:

1. Create and configure a single-node SQL Server failover cluster instance. At the completion of a successful configuration of the node, you have a fully functional failover cluster instance. At this time it does not have high-availability because there is only one node in the failover cluster.
2. On each node to be added to the SQL Server failover cluster, run Setup with Add Node functionality to add that node.

Option 2: Advanced/Enterprise installation

SQL Server Advanced/Enterprise failover cluster installation consists of two steps:

1. On each node that will be part of the SQL Server failover cluster, run Setup with Prepare Failover Cluster

functionality. This step prepares the nodes ready to be clustered, but there is no operational SQL Server instance at the end of this step.

2. After the nodes are prepared for clustering, run Setup on the node that owns the shared disk with the Complete Failover Cluster functionality. This step configures and completes the failover cluster instance. At the end of this step, you will have an operational SQL Server failover cluster instance.

NOTE

Either installation option allows for multi-node SQL Server failover cluster installation. Add Node can be used to add additional nodes for either option after a SQL Server failover cluster has been created.

IMPORTANT

The operating system drive letter for SQL Server install locations must match on all the nodes added to the SQL Server failover cluster.

IP Address Configuration During Setup

SQL Server Setup lets you set or change the IP resource dependency settings during the following actions:

- Integrated Install - [Create a New SQL Server Failover Cluster \(Setup\)](#)
- CompleteFailoverCluster (Advanced Install) - [Create a New SQL Server Failover Cluster \(Setup\)](#)
- Add Node - [Add or Remove Nodes in a SQL Server Failover Cluster \(Setup\)](#)
- Remove Node - [Add or Remove Nodes in a SQL Server Failover Cluster \(Setup\)](#)

Note IPV6 IP addresses are supported. If you configure both IPV4 and IPV6 there are treated like different subnets, and IPV6 is expected to come online first.

SQL Server Multi-Subnet Failover Cluster

You can set OR dependencies when the nodes on the cluster are on different subnets. However, each node in the SQL Server multi-subnet failover cluster must be a possible owner of at least one of IP address specified.

See Also

[Before Installing Failover Clustering](#)

[Create a New SQL Server Failover Cluster \(Setup\)](#)

[Install SQL Server 2016 from the Command Prompt](#)

[Upgrade a SQL Server Failover Cluster Instance](#)

Before Installing Failover Clustering

3/29/2017 • 16 min to read • [Edit Online](#)

Before you install a SQL Server failover cluster, you must select the hardware and the operating system on which SQL Server will run. You must also configure Windows Server Failover Clustering (WSFC), and review network, security, and considerations for other software that will run on your failover cluster.

If a Windows cluster has a local disk drive and the same drive letter is also used on one or more cluster nodes as a shared drive, you cannot install SQL Server on that drive.

You may also want to review the following topics to learn more about SQL Server failover clustering concepts, features and tasks.

TOPIC DESCRIPTION	TOPIC
Describes SQL Server failover clustering concepts, and provides links to associated content and tasks.	Always On Failover Cluster Instances (SQL Server)
Describes SQL Server failover policy concepts, and provides links to configuring the failover policy to suit your organizational requirements.	Failover Policy for Failover Cluster Instances
Describes how to maintain and your existing SQL Server failover cluster.	Failover Cluster Instance Administration and Maintenance
Explains how to install Analysis Services on a Windows Server Failover Cluster (WSFC).	How to Cluster SQL Server Analysis Services

Best Practices

- Review SQL Server 2016 [Release Notes](#)
- Install prerequisite software. Before running Setup to install or upgrade to SQL Server 2016, install the following prerequisites to reduce installation time. You can install prerequisite software on each failover cluster node and then restart nodes once before running Setup.
 - Windows PowerShell is no longer installed by SQL Server Setup. Windows PowerShell is a prerequisite for installing SQL Server 2016 Database Engine components and SQL Server Management Studio. If Windows PowerShell is not present on your computer, you can enable it by following the instructions on the [Windows Management Framework](#) page.
 - .NET Framework 3.5 SP1 is no longer installed by SQL Server Setup, but may be required while installing SQL Server on older Windows operating systems. For more information, see SQL Server 2016 [Release Notes](#).
 - **Microsoft Update package:** To avoid computer restart due to .NET Framework 4 installation during setup, SQL Server 2016 setup requires a Microsoft update to be installed on the computer. If you are installing SQL Server 2014 on Windows 7 SP1 or Windows Server 2008 SP2 this update is included. If you are installing on an older Windows operating system, download it from [Microsoft Update for .NET Framework 4.0 on Windows Vista and Windows Server 2008](#).
 - .NET Framework 4: Setup installs .NET Framework 4 on a clustered operating system. To reduce installation time, you may consider installing .NET Framework 4 before you run Setup.

- SQL Server Setup support files. You can install these files by running SqlSupport.msi located on your SQL Server 2016 installation media.
- Verify that antivirus software is not installed on your WSFC cluster. For more information, see the Microsoft Knowledge Base article, [Antivirus software may cause problems with cluster services](#).
- When naming a cluster group for your failover cluster installation, you must not use any of the following characters in the cluster group name:
 - Less than operator (<)
 - Greater than operator (>)
 - Double quote (")
 - Single quote (')
 - Ampersand (&)

Also verify that existing cluster group names do not contain unsupported characters.

- Ensure that all cluster nodes are configured identically, including COM+, disk drive letters, and users in the administrators group.
- Verify that you have cleared the system logs in all nodes and viewed the system logs again. Ensure that the logs are free of any error messages before continuing.
- Before you install or update a SQL Server failover cluster, disable all applications and services that might use SQL Server components during installation, but leave the disk resources online.
- SQL Server Setup automatically sets dependencies between the SQL Server cluster group and the disks that will be in the failover cluster. Do not set dependencies for disks before Setup.
 - During SQL Server Failover Cluster installation, computer object (Active Directory computer accounts) for the SQL Server Network Resource Name is created. In a Windows Server 2008 cluster, the cluster name account (computer account of the cluster itself) needs to have permissions to create computer objects. For more information, see [Configuring Accounts in Active Directory](#).
 - If you are using SMB File share as a storage option, the SQL Server Setup account must have SeSecurityPrivilege on the file server. To do this, using the Local Security Policy console on the file server, add the SQL Server setup account to **Manage auditing and security log** rights.

Verify Your Hardware Solution

- If the cluster solution includes geographically dispersed cluster nodes, additional items like network latency and shared disk support must be verified.
 - For more information about Windows Server 2008 and Windows Server 2008 R2, see [Validating Hardware for a failover cluster](#) and [Support Policy for Windows Failover Clusters](#).
- Verify that the disk where SQL Server will be installed is not compressed or encrypted. If you attempt to install SQL Server to a compressed drive or an encrypted drive, SQL Server Setup fails.
- SAN configurations are also supported on Windows Server 2008 and Windows Server 2008 R2 Advanced Server and Datacenter Server editions. The Windows Catalog and Hardware Compatibility List category "Cluster/Multi-cluster Device" lists the set of SAN-capable storage devices that have been tested and are supported as SAN storage units with multiple WSFC clusters attached. Run cluster validation after finding the certified components.
- SMB File Share is also supported for installing data files. For more information, see [Storage Types for Data](#)

WARNING

If you are using Windows File Server as a SMB File Share storage, the SQL Server Setup account must have SeSecurityPrivilege on the file server. To do this, using the Local Security Policy console on the file server, add the SQL Server setup account to **Manage auditing and security log** rights.

If you are using SMB file share storage other than Windows File server, please consult the storage vendor for an equivalent setting on the file server side.

- SQL Server supports mount points.

A mounted volume, or mount point, allows you to use a single drive letter to refer to many disks or volumes. If you have a drive letter D: that refers to a regular disk or volume, you can connect or "mount" additional disks or volumes as directories under drive letter D: without the additional disks or volumes requiring drive letters of their own.

Additional mount point considerations for SQL Server failover clustering:

- SQL Server Setup requires that the base drive of a mounted drive has an associated drive letter. For failover cluster installations, this base drive must be a clustered drive. Volume GUIDs are not supported in this release.
- The base drive, the one with the drive letter, cannot be shared among failover cluster instances. This is a normal restriction for failover clusters, but is not a restriction on stand-alone, multi-instance servers.
- The clustered installations of SQL Server are limited to the number of available drive letters. Assuming that you use only one drive letter for the operating system, and all other drive letters are available as normal cluster drives or cluster drives hosting mount points, you are limited to a maximum of 25 instances of SQL Server per failover cluster.

TIP

The 25 instance limit can be overcome by using SMB file share option. If you use SMB file share as the storage option, you can install up to 50 SQL Server failover cluster instances.

- Formatting a drive after mounting additional drives is not supported.
- SQL Server failover cluster installation supports Local Disk only for installing the tempdb files. Ensure that the path specified for the tempdb data and log files is valid on all the cluster nodes. During failover, if the tempdb directories are not available on the failover target node, the SQL Server resource will fail to come online. For more information, see [Storage Types for Data Files](#) and [Database Engine Configuration - Data Directories](#).
- If you deploy a SQL Server failover cluster on iSCSI technology components, we recommend that you use appropriate caution. For more information, see [Support for SQL Server on iSCSI technology components](#).
- For more information, see [SQL Server support policy for Microsoft Clustering](#).
- For more information about proper quorum drive configuration, see [Quorum Drive Configuration Information](#).
- To install a SQL Server failover cluster when the SQL Server source installation files and the cluster exist on different domains, copy the installation files to the current domain available to the SQL Server failover cluster.

Review Security Considerations

- To use encryption, install the server certificate with the fully qualified DNS name of the WSFC cluster on all nodes in the SQL Server failover cluster. For example, if you have a two-node cluster, with nodes named "Test1.DomainName.com" and "Test2.DomainName.com" and a SQL Server failover cluster instance named "Virtsql", you must get a certificate for "Virtsql.DomainName.com" and install the certificate on the test1 and test2 nodes. Then you can select the **Force protocol encryption** check box on the SQL Server Configuration Manager to configure your failover cluster for encryption.

IMPORTANT

Do not select the **Force protocol encryption** check box until you have certificates installed on all participating nodes in your failover cluster instance.

- For SQL Server installations in side-by-side configurations with previous versions, SQL Server services must use accounts found only in the global domains group. Additionally, accounts used by SQL Server services must not appear in the local Administrators group. Failure to comply with this guideline will result in unexpected security behavior.
- To create a failover cluster, you must be a local administrator with permissions to log on as a service, and to act as part of the operating system on all nodes of the failover cluster instance.
- On Windows Server 2008, service SIDs are generated automatically for use with SQL Server 2016 services. For SQL Server 2016 failover cluster instances upgraded from previous versions of SQL Server, existing domain groups and ACL configurations will be preserved.
- Domain groups must be within the same domain as the machine accounts. For example, if the machine where SQL Server will be installed is in the SQLSVR domain which is a child of MYDOMAIN, you must specify a group in the SQLSVR domain. The SQLSVR domain may contain user accounts from MYDOMAIN.
- SQL Server failover clustering cannot be installed where cluster nodes are domain controllers.
- Review content in [Security Considerations for a SQL Server Installation](#).
- To enable Kerberos authentication with SQL Server, see [How to use Kerberos authentication in SQL Server](#) in the Microsoft Knowledge Base.

Review Network, Port, and Firewall Considerations

- Verify that you have disabled NetBIOS for all private network cards before beginning SQL Server Setup.
- The network name and IP address of your SQL Server should not be used for any other purpose, such as file sharing. If you want to create a file share resource, use a different, unique network name and IP address for the resource.

IMPORTANT

We recommend that you do not use file shares on data drives, because they can affect SQL Server behavior and performance.

- Even though SQL Server supports both Named Pipes and TCP/IP Sockets over TCP/IP within a cluster, we recommend that you use TCP/IP Sockets in a clustered configuration.
- Note that ISA server is not supported on Windows Clustering and consequently is also not supported on SQL Server failover clusters.

- The Remote Registry service must be up and running.
- Remote Administration must be enabled.
- For the SQL Server port, use SQL Server Configuration Manager to check the SQL Server network configuration for the TCP/IP protocol for the instance you want to unblock. You must enable the TCP port for IPALL if you want to connect to SQL Server using TCP after installation. By default, SQL Browser listens on UDP port 1434.
- Failover cluster Setup operations include a rule that checks network binding order. Although binding orders might seem correct, you might have disabled or "ghosted" NIC configurations on the system. "Ghosted" NIC configurations can affect the binding order and cause the binding order rule to issue a warning. To avoid this situation, use the following steps to identify and remove disabled network adapters:
 1. At a command prompt, type: set devmgr_Show_Nonpersistent_Devices=1.
 2. Type and run: start Devmgmt.msc.
 3. Expand the list of network adapters. Only the physical adapters should be in the list. If you have a disabled network adapter, Setup will report a failure for the network binding order rule. Control Panel/Network Connections will also show that adapter was disabled. Confirm that Network Settings in Control Panel shows the same list of enabled physical adapters that devmgmt.msc shows.
 4. Remove disabled network adapters before you run SQL Server Setup.
 5. After Setup finishes, return to Network Connections in Control Panel and disable any network adapters that are not currently in use.

Verify Your Operating System

Make sure that your operating system is installed properly and is designed to support failover clustering. The following table is a list of SQL Server editions and the operating systems that support them.

SQL SERVER EDITION	WINDOWS SERVER 2008 ENTERPRISE	WINDOWS SERVER 2008 DATACENTER SERVER	WINDOWS SERVER 2008 R2 ENTERPRISE	WINDOWS SERVER 2008 R2 DATACENTER SERVER
SQL Server 2014 Enterprise (64-bit) x64*	Yes	Yes	Yes**	Yes**
SQL Server 2014 Enterprise (32-bit)	Yes	Yes		
SQL Server 2016 –bit) Developer (64	Yes	Yes	Yes**	Yes**
SQL Server 2016 Developer (32-bit)	Yes	Yes		
SQL Server 2016 Standard (64-bit)	Yes	Yes	Yes	Yes
SQL Server 2016 Standard (32-bit)	Yes	Yes		

* SQL Server clusters are not supported in WOW mode. That includes upgrades from previous versions of SQL

Server failover clusters that were originally installed in WOW. For those the only upgrade option is to install the new version side by side and migrate.

**Supported for SQL Server multi-subnet failover clustering.

Additional Considerations for Multi-Subnet Configurations

The sections below describe the requirements to keep in mind when installing a SQL Server multi-subnet failover cluster. A multi-subnet configuration involves clustering across multiple subnets, therefore involves using multiple IP addresses and changes to IP address resource dependencies.

SQL Server Edition and Operating System Considerations

- For information about the editions of SQL Server that support a SQL Server multi-subnet failover cluster, see [Features Supported by the Editions of SQL Server 2016](#).
- To create a SQL Server multi-subnet failover cluster, you must first create the Windows Server 2008 R2 multi-site failover cluster on multiple subnets.
- SQL Server failover cluster depends on the Windows Server failover cluster to make sure that the IP dependency conditions are valid if there is a failover.
- Windows Server 2008 R2 requires that all the cluster servers must be in the same Active Directory domain. Therefore, SQL Server multi-subnet failover cluster requires that all the cluster nodes be in the same Active Directory domain even if they are in different subnets.

IP Address and IP Address Resource Dependencies

1. The IP Address resource dependency is set to OR in a multi-subnet configuration. For more information, see [Create a New SQL Server Failover Cluster \(Setup\)](#)
2. Mixed AND-OR IP address dependencies are not supported. For example, <IP1> AND <IP2> OR <IP3> is not supported.
3. More than one IP address per subnet is not supported.

If you decide to use more than one IP address configured for the same subnet, you may experience client connection failures during SQL Server startup.

Related Content

For more information about Windows Server 2008 R2 multi-site failover, see [Windows Server 2008 R2 Failover Clustering Site](#) and [Design for a Clustered Service or Application in a Multi-Site Failover Cluster](#).

Configure Windows Server Failover Cluster

- Microsoft Cluster Service (WSFC) must be configured on at least one node of your server cluster. You must also run SQL Server Enterprise, SQL Server Business Intelligence, or SQL Server Standard in conjunction with WSFC. SQL Server Enterprise support failover clusters with up to 16 nodes. SQL Server Business Intelligence and SQL Server Standard supports two-node failover clusters.
- The resource DLL for the SQL Server service exports two functions used by WSFC Cluster Manager to check for availability of the SQL Server resource. For more information, see [Failover Policy for Failover Cluster Instances](#).
- WSFC must be able to verify that the failover clustered instance is running by using the IsAlive check. This requires connecting to the server by using a trusted connection. By default, the account that runs the cluster service is not configured as an administrator on nodes in the cluster, and the BUILTIN\Administrators group does not have permission to log into SQL Server. These settings change only if you change permissions on the cluster nodes.

- Configure Domain Name Service (DNS) or Windows Internet Name Service (WINS). A DNS server or WINS server must be running in the environment where your SQL Server failover cluster will be installed. SQL Server Setup requires dynamic domain name service registration of the SQL Server IP interface virtual reference. DNS server configuration should allow cluster nodes to dynamically register an online IP address map to Network Name. If the dynamic registration cannot be completed, Setup fails and the installation is rolled back. For more information, see [this knowledgebase article](#)

Install Microsoft Distributed Transaction Coordinator

Before installing SQL Server on a failover cluster, determine whether the Microsoft Distributed Transaction Coordinator (MSDTC) cluster resource must be created. If you are installing only the Database Engine, the MSDTC cluster resource is not required. If you are installing the Database Engine and SSIS, Workstation Components, or if you will use distributed transactions, you must install MSDTC. Note that MSDTC is not required for Analysis Services-only instances.

On Windows Server 2008 and Windows Server 2008 R2, you can install multiple instances of MSDTC on a single failover cluster. The first instance of MSDTC that is installed will be the cluster default instance of MSDTC. SQL Server will take advantage of an instance of MSDTC installed to the SQL Server local cluster resource group by automatically using the instance of MSDTC. However, individual applications can be mapped to any instance of MSDTC on the cluster.

The following rules are applied for an instance of MSDTC to be chosen by SQL Server:

- Use MSDTC installed to the local group, else
- Use the mapped instance of MSDTC, else
- Use the cluster's default instance of MSDTC, else
- Use the local machine's installed instance of MSDTC

IMPORTANT

If the MSDTC instance that is installed to the local cluster group of SQL Server fails, SQL Server does not automatically attempt to use the default cluster instance or the local machine instance of MSDTC. You would need to completely remove the failed instance of MSDTC from the SQL Server group to use another instance of MSDTC. Likewise, if you create a mapping for SQL Server and the mapped instance of MSDTC fails, your distributed transactions will also fail. If you want SQL Server to use a different instance of MSDTC, you must either add an instance of MSDTC to the local cluster group of the SQL Server or delete the mapping.

Configure Microsoft Distributed Transaction Coordinator

After you install the operating system and configure your cluster, you must configure MSDTC to work in a cluster by using the Cluster Administrator. Failure to cluster MSDTC will not block SQL Server Setup, but SQL Server application functionality may be affected if MSDTC is not properly configured.

See Also

[Hardware and Software Requirements for Installing SQL Server 2016](#)

[Check Parameters for the System Configuration Checker](#)

[Failover Cluster Instance Administration and Maintenance](#)

Create a New SQL Server Failover Cluster (Setup)

3/29/2017 • 28 min to read • [Edit Online](#)

To install or upgrade a SQL Server failover cluster, you must run the Setup program on each node of the failover cluster. To add a node to an existing SQL Server failover cluster, you must run SQL Server Setup on the node that is to be added to the SQL Server failover cluster instance. Do not run Setup on the active node to manage the other nodes.

Depending on how the nodes are clustered, the SQL Server failover cluster is configured in the following ways:

- Nodes on the same subnet or the same set of subnets - The IP address resource dependency is set to AND for these types of configurations.
- Nodes on different subnets - The IP address resource dependency is set to OR and this configuration is called a SQL Server multi-subnet failover cluster configuration. For more information, see [SQL Server Multi-Subnet Clustering \(SQL Server\)](#).

The following options are available for SQL Server failover cluster installation:

Option1: Integration Installation with Add Node

SQL Server integrated failover cluster installation consists of the following steps:

- Create and configure a single-node SQL Server failover cluster instance. When you configure the node successfully, you have a fully functional failover cluster instance. At this point, it does not have high availability because there is only one node in the failover cluster.
- On each node to be added to the SQL Server failover cluster, run Setup with Add Node functionality to add that node.
 - If the node you are adding has additional or different subnets, Setup allows you to specify additional IP addresses. If the node you are adding is on a different subnet, you also need to confirm the IP address resource dependency change to OR. For more information on the different possible scenarios during Add Node operations, see [Add or Remove Nodes in a SQL Server Failover Cluster \(Setup\)](#).

Option 2: Advanced/Enterprise Installation

SQL Server Advanced/Enterprise failover cluster installation consists of the following steps:

- On each node that is a possible owner of the new SQL Server failover cluster, follow the Prepare Failover Cluster setup steps that are listed in the [Prepare section](#). After you run the Prepare Failover Cluster on one node, Setup creates the Configuration.ini file that lists all the settings that you specified. On the additional nodes to be prepared, instead of following these steps, you can supply the autogenerated Configuration.ini file from first node as an input to the Setup command line. For more information, see [Install SQL Server 2016 Using a Configuration File](#). This step prepares the nodes ready to be clustered, but there is no operational instance of SQL Server at the end of this step.
- After the nodes are prepared for clustering, run Setup on one of the prepared nodes. This step configures and finishes the failover cluster instance. At the end of this step, you will have an operational SQL Server failover cluster instance and all the nodes that were prepared previously for that instance will be the possible owners of the newly-created SQL Server failover cluster.

If you are clustering nodes on multiple subnets, Setup will detect the union of all the subnets across all nodes that have the SQL Server prepared failover cluster instance. You will be able to specify multiple IP addresses for the subnets. Each prepared node must be the possible owner of at least one IP address.

If each of the IP addresses specified for the subnets are supported on all the prepared nodes, the dependency is set to AND.

If each of the IP addresses specified for the subnets are not supported on all the prepared nodes, the dependency is set to OR. For more information, see [SQL Server Multi-Subnet Clustering \(SQL Server\)](#).

NOTE

Complete Failover Cluster requires that the underlying Windows failover cluster exists. If the Windows failover cluster does not exist, Setup gives an error and exits.

For more information about how to add nodes to or remove nodes from an existing failover cluster instance, see [Add or Remove Nodes in a SQL Server Failover Cluster \(Setup\)](#).

For more information about remote installation, see [Supported Version and Edition Upgrades](#).

For more information about installing Analysis Services in a Windows failover cluster, see [How to Cluster SQL Server Analysis Services](#).

Prerequisites

Before you begin, review the following SQL Server Books Online topics:

- [Planning a SQL Server Installation](#)
- [Before Installing Failover Clustering](#)
- [Security Considerations for a SQL Server Installation](#)
- [SQL Server Multi-Subnet Clustering \(SQL Server\)](#)

NOTE

Take note of the location of the shared drive in the Cluster Administrator before you run SQL Server Setup. You must have this information to create a new failover cluster.

To install a new SQL Server failover cluster using Integrated Install with Add Node.

1. Insert the SQL Server installation media, and from the root folder, double-click Setup.exe. To install from a network share, browse to the root folder on the share, and then double-click Setup.exe. For more information about how to install prerequisites, see [Before Installing Failover Clustering](#).
2. The Installation Wizard starts the SQL Server Installation Center. To create a new cluster installation of SQL Server, click **New SQL Server failover cluster installation** on the installation page.
3. The System Configuration Checker runs a discovery operation on your computer. To continue, Click **OK**. You can view the details on the screen by clicking **Show Details**, or as an HTML report by clicking **View detailed report**.
4. To continue, click **Next**.
5. On the Setup Support Files page, click **Install** to install the Setup support files.
6. The System Configuration Checker verifies the system state of your computer before Setup continues. After the check is complete, click **Next** to continue. You can view the details on the screen by clicking **Show Details**, or as an HTML report by clicking **View detailed report**.
7. On the Product key page, indicate whether you are installing a free edition of SQL Server, or whether you

have a PID key for a production version of the product. For more information, see [Editions and Components of SQL Server 2016](#).

8. On the License Terms page, read the license agreement, and then select the check box to accept the license terms and conditions. To help improve SQL Server, you can also enable the feature usage option and send reports to Microsoft. Click **Next** to continue. To end Setup, click **Cancel**.
9. On the Feature Selection page, select the components for your installation. A description for each component group appears in the right pane after you select the feature name. You can select any combination of check boxes, but only Database Engine, Analysis Services in tabular mode, and Analysis Services in multidimensional mode support failover clustering. Other selected components will run as a stand-alone feature without failover capability on the current node that you are running Setup on. For more information on Analysis Services modes, see [Determine the Server Mode of an Analysis Services Instance](#).

The prerequisites for the selected features are displayed on the right-hand pane. SQL Server Setup will install the prerequisite that are not already installed during the installation step described later in this procedure.

You can specify a custom directory for shared components by using the field at the bottom of this page. To change the installation path for shared components, either update the path in the field provided at the bottom of the dialog box, or click the ellipsis button to browse to an installation directory. The default installation path is C:\Program Files\ Microsoft SQL Server \.

SQL Server also supports installing system databases (Master, Model, MSDB, and TempDB), and Database Engine user databases on a Server Message Block (SMB) file share. For more information on installing SQL Server with SMB file share as a storage, see [Install SQL Server with SMB Fileshare as a Storage Option](#).

The path specified for the shared components must be an absolute path. The folder must not be compressed or encrypted. Mapped drives are not supported. If you are installing SQL Server on a 64-bit operating system, you will see the following options:

- a. Shared feature directory
- b. Shared feature directory (x86)

The path specified for each of the above options must be different.

NOTE

When you select the Database Engine Services feature, both replication and full-text search are selected automatically. Data Quality Services (DQS) is also selected when you select the Database Engine Services feature. Unselecting any of these subfeatures also unselects the Database Engine Services feature.

10. SQL Server setup runs one more set of rules that are based on the features you selected to validate your configuration.
11. On the Instance Configuration page, specify whether to install a default or a named instance. For more information, see [Instance Configuration](#).

SQL Server Network Name — Specify a network name for the new SQL Server failover cluster. This is the name that is used to identify your failover cluster on the network.

NOTE

This is known as the virtual SQL Server name in earlier versions of SQL Server failover clusters.

Instance ID — By default, the instance name is used as the Instance ID. This is used to identify installation directories and registry keys for your instance of SQL Server. This is the case for default instances and named instances. For a default instance, the instance name and instance ID would be MSSQLSERVER. To use a nondefault instance ID, select the **Instance ID** box and provide a value.

NOTE

Typical stand-alone instances of SQL Server, whether default or named instances, do not use a nondefault value for the **Instance ID** box.

Instance root directory — By default, the instance root directory is C:\Program Files\ Microsoft SQL Server\. To specify a nondefault root directory, use the field provided, or click the ellipsis button to locate an installation folder.

Detected SQL Server instances and features on this computer - The grid shows instances of SQL Server that are on the computer where Setup is running. If a default instance is already installed on the computer, you must install a named instance of SQL Server. Click **Next** to continue.

12. Use the Cluster Resource Group page to specify the cluster resource group name where SQL Server virtual server resources will be located. To specify the SQL Server cluster resource group name, you have two options:
 - Use the drop-down box to specify an existing group to use.
 - Type the name of a new group to create. Be aware that the name "Available storage" is not a valid group name.
13. On the Cluster Disk Selection page, select the shared cluster disk resource for your SQL Server failover cluster. The cluster disk is where the SQL Server data will be put. More than one disk can be specified. The Available shared disks grid displays a list of available disks, whether each is qualified as a shared disk, and a description of each disk resource. Click **Next** to continue.

NOTE

The first drive is used as the default drive for all databases, but can be changed on the Database Engine or Analysis Services configuration pages.

On the Cluster Disk Selection page, you can optionally skip selecting any shared disk if you want to use SMB fileshare as a data folder.

14. On the Cluster Network Configuration page, specify the network resources for your failover cluster instance:
 - **Network Settings** — Specify the IP type and IP address for your failover cluster instance.

Click **Next** to continue.
15. Use this page to specify Cluster Security Policy.
 - Windows Server 2008 and later versions - Service SIDs (server security IDs) are the recommended and default setting. There is no option for changing this to security groups. For information about service SIDs functionality on Windows Server 2008, see [Configure Windows Service Accounts and Permissions](#). This has been tested in standalone and cluster setup on Windows Server 2008 R2.
 - On Windows Server 2003, specify domain groups for SQL Server services. All resource permissions are controlled by domain-level groups that include SQL Server service accounts as group members.

Click **Next** to continue.

16. Work flow for the rest of this topic depends on the features that you have specified for your installation. You might not see all the pages, depending on your selections (Database Engine, Analysis Services, and Reporting Services).

17. On the Server Configuration — Service Accounts page, specify login accounts for SQL Server services. The actual services that are configured on this page depend on the features that you selected to install.

You can assign the same login account to all SQL Server services, or you can configure each service account individually. The startup type is set to manual for all cluster-aware services, including full-text search and SQL Server Agent, and cannot be changed during installation. Microsoft recommends that you configure service accounts individually to provide least privileges for each service, where SQL Server services are granted the minimum permissions they have to have complete their tasks. For more information, see [Server Configuration - Service Accounts](#) and [Configure Windows Service Accounts and Permissions](#).

To specify the same logon account for all service accounts in this instance of SQL Server, provide credentials in the fields at the bottom of the page.

Security Note Do not use a blank password. Use a strong password.

When you are finished specifying login information for SQL Server services, click **Next**.

18. Use the **Server Configuration - Collation** tab to specify nondefault collations for the Database Engine and Analysis Services.

19. Use the Database Engine Configuration — Account Provisioning page to specify the following:

- Security Mode - select Windows Authentication or Mixed Mode Authentication for your instance of SQL Server. If you select Mixed Mode Authentication, you must provide a strong password for the built-in SQL Server system administrator account.

After a device establishes a successful connection to SQL Server, the security mechanism is the same for both Windows Authentication and Mixed Mode.

- SQL Server Administrators - You must specify at least one system administrator for the instance of SQL Server. To add the account under which SQL Server Setup is running, click **Add Current User**. To add or remove accounts from the list of system administrators, click **Add** or **Remove**, and then edit the list of users, groups, or computers that will have administrator privileges for the instance of SQL Server.

When you are finished editing the list, Click **OK**. Verify the list of administrators in the configuration dialog box. When the list is complete, click **Next**.

20. Use the Database Engine Configuration - Data Directories page to specify nondefault installation directories. To install to default directories, click **Next**.

IMPORTANT

If you specify nondefault installation directories, make sure that the installation folders are unique to this instance of SQL Server. None of the directories in this dialog box should be shared with directories from other instances of SQL Server. The data directories should be located on the shared cluster disk for the failover cluster.

NOTE

To specify a Server Message Block (SMB) file server as the Data Directory, set **default data root directory** to the fileshare of the format \\Servername\ShareName\...

21. Use the Database Engine Configuration - FILESTREAM page to enable FILESTREAM for your instance of SQL Server. Click **Next** to continue.
22. Use the Analysis Services Configuration — Account Provisioning page to specify users or accounts that will have administrator permissions for Analysis Services. You must specify at least one system administrator for Analysis Services. To add the account under which SQL Server Setup is running, click **Add Current User**. To add or remove accounts from the list of system administrators, click **Add** or **Remove**, and then edit the list of users, groups, or computers that will have administrator privileges for Analysis Services.

When you are finished editing the list, Click **OK**. Verify the list of administrators in the configuration dialog box. When the list is complete, click **Next**.

23. Use the Analysis Services Configuration — Data Directories page to specify nondefault installation directories. To install to default directories, click **Next**.

IMPORTANT

If you specify nondefault installation directories, make sure that the installation folders are unique to this instance of SQL Server. None of the directories in this dialog box should be shared with directories from other instances of SQL Server. The data directories should be located on the shared cluster disk for the failover cluster.

24. Use the Reporting Services Configuration page to specify the kind of Reporting Services installation to create. For failover cluster installation, the option is set to Unconfigured Reporting Services installation. You must configure Reporting Services services after you complete the installation.
25. The System Configuration Checker runs one more set of rules to validate your configuration with the SQL Server features that you have specified.
26. The Ready to Install page displays a tree view of installation options that were specified during Setup. To continue, click **Install**. Setup will first install the required prerequisites for the selected features followed by the feature installation.
27. During installation, the Installation Progress page provides status so that you can monitor installation progress as Setup continues.
28. After installation, the **Complete** page provides a link to the summary log file for the installation and other important notes. To complete the SQL Server installation process, click **Close**.
29. If you are instructed to restart the computer, do so now. It is important to read the message from the Installation Wizard when you have finished with Setup. For information about Setup log files, see [View and Read SQL Server Setup Log Files](#).
30. To add nodes to the single-node failover you just created, run Setup on each additional node and follow the steps for AddNode operation. For more information, see [Add or Remove Nodes in a SQL Server Failover Cluster \(Setup\)](#).

NOTE

If you are adding more than one node, you can use the configuration file to deploy the installations. For more information, see [Install SQL Server 2016 Using a Configuration File](#).

The SQL Server edition you are installing must match across all the nodes in a SQL Server failover cluster. When you add a new node to an existing SQL Server failover cluster, make sure that you specify that the edition matches the edition of the existing failover cluster.

Prepare

Advanced/Enterprise Failover Cluster Install Step 1: Prepare

1. Insert the SQL Server installation media, and from the root folder, double-click Setup.exe. To install from a network share, browse to the root folder on the share, and then double-click Setup.exe. For more information about how to install prerequisites, see [Before Installing Failover Clustering](#). You may be asked to install the prerequisites, if they are not previously installed.
2. Windows Installer 4.5 is required, and may be installed by the Installation Wizard. If you are prompted to restart your computer, restart and then start SQL Server Setup again.
3. After the prerequisites are installed, the Installation Wizard starts the SQL Server Installation Center. To prepare the node for clustering, move to the **Advanced** page and then click **Advanced cluster preparation**.
4. The System Configuration Checker runs a discovery operation on your computer. To continue, Click **OK**. You can view the details on the screen by clicking **Show Details**, or as an HTML report by clicking **View detailed report**.
5. On the Setup Support Files page click **Install** to install the Setup support files.
6. The System Configuration Checker verifies the system state of your computer before Setup continues. After the check is complete, click **Next** to continue. You can view the details on the screen by clicking **Show Details**, or as an HTML report by clicking **View detailed report**.
7. On the Language Selection page, you can specify the language for your instance of SQL Server if you are installing on a localized operating system and the installation media includes language packs for both English and the language corresponding to the operating system. For more information about cross-language support and installation considerations, see [Local Language Versions in SQL Server](#).

To continue, click **Next**.

8. On the Product key page, click to indicate whether you are installing a free edition of SQL Server, or whether you have a PID key for a production version of the product. For more information, see [Editions and Components of SQL Server 2016](#).

NOTE

You must specify the same product key on all the nodes that you are preparing for the same failover cluster.

9. On the License Terms page, read the license agreement, and then select the check box to accept the license terms and conditions. To help improve SQL Server, you can also enable the feature usage option and send reports to Microsoft. Click **Next** to continue. To end Setup, click **Cancel**.
10. On the Feature Selection page, select the components for your installation. A description for each component group appears in the right pane after you select the feature name. You can select any combination of check boxes, but only Database Engine, Analysis Services in tabular mode, and Analysis

Services in multidimensional mode support failover clustering. Other selected components will run as a stand-alone feature without failover capability on the current node that you are running Setup on. For more information on Analysis Services modes, see [Determine the Server Mode of an Analysis Services Instance](#).

The prerequisites for the selected features are displayed on the right-hand pane. SQL Server Setup will install the prerequisite that are not already installed during the installation step described later in this procedure.

You can specify a custom directory for shared components by using the field at the bottom of this page. To change the installation path for shared components, either update the path in the field provided at the bottom of the dialog box, or click the ellipsis button to browse to an installation directory. The default installation path is C:\Program Files\ Microsoft SQL Server\.

NOTE

When you select the Database Engine Services feature, both replication and full-text search are selected automatically. Unselecting any of these subfeatures also unselects the Database Engine Services feature.

11. On the Instance Configuration page, specify whether to install a default or a named instance.

Instance ID — By default, the instance name is used as the Instance ID. This is used to identify installation directories and registry keys for your instance of SQL Server. This is the case for default instances and named instances. For a default instance, the instance name and instance ID would be MSSQLSERVER. To use a nondefault instance ID, select the **Instance ID** text box and provide a value.

NOTE

Typical stand-alone instances of SQL Server, whether default or named instances, do not use a nondefault value for the **Instance ID** text box.

IMPORTANT

Use the same InstanceID for all the nodes that are prepared for the failover cluster

Instance root directory — By default, the instance root directory is C:\Program Files\ Microsoft SQL Server\. To specify a nondefault root directory, use the field provided, or click the ellipsis button to locate an installation folder.

Installed instances - The grid shows instances of SQL Server that are on the computer where Setup is running. If a default instance is already installed on the computer, you must install a named instance of SQL Server. Click **Next** to continue.

12. The Disk Space Requirements page calculates the required disk space for the features that you specify, and compares requirements to the available disk space on the computer where Setup is running.
13. Use this page to specify Cluster Security Policy.
 - Windows Server 2008 and later versions - Service SIDs (server security IDs) are the recommended and default setting. There is no option for changing this to security groups. For information about service SIDs functionality on Windows Server 2008, see [Configure Windows Service Accounts and Permissions](#). This has been tested in standalone and cluster setup on Windows Server 2008 R2.
 - On Windows Server 2003, specify domain groups for SQL Server services. All resource permissions are controlled by domain-level groups that include SQL Server service accounts as group members.

Click **Next** to continue.

14. Work flow for the rest of this topic depends on the features that you have specified for your installation. You might not see all the pages, depending on your selections.
15. On the Server Configuration — Service Accounts page, specify login accounts for SQL Server services. The actual services that are configured on this page depend on the features that you selected to install.

You can assign the same login account to all SQL Server services, or you can configure each service account individually. The startup type is set to manual for all cluster-aware services, including full-text search and SQL Server Agent, and cannot be changed during installation. Microsoft recommends that you configure service accounts individually to provide least privileges for each service, where SQL Server services are granted the minimum permissions they have to have complete their tasks. For more information, see [Configure Windows Service Accounts and Permissions](#).

To specify the same logon account for all service accounts in this instance of SQL Server, provide credentials in the fields at the bottom of the page.

Security Note Do not use a blank password. Use a strong password.

When you are finished specifying login information for SQL Server services, click **Next**.

16. Use the **Server Configuration - Collation** tab to specify nondefault collations for the Database Engine and Analysis Services.
17. Use the **Server Configuration - Filestream** to enable FILESTREAM for your instance of SQL Server. Click **Next** to continue.
18. Use the Reporting Services Configuration page to specify the kind of Reporting Services installation to create. For failover cluster installation, the option is set to Unconfigured Reporting Services installation. You must configure Reporting Services services after you complete the installation.
19. On the Error Reporting page, specify the information that you want to send to Microsoft that will help improve SQL Server. By default, options for error reporting is enabled.
20. The System Configuration Checker runs one more set of rules to validate your configuration with the SQL Server features that you have specified.
21. The Ready to Install page displays a tree view of installation options that were specified during Setup. To continue, click **Install**. Setup will first install the required prerequisites for the selected features followed by the feature installation.

During installation, the Installation Progress page provides status so that you can monitor installation progress as Setup continues. After installation, the **Complete** page provides a link to the summary log file for the installation and other important notes.

22. To complete the SQL Server installation process, click **Close**.
23. If you are instructed to restart the computer, do so now. It is important to read the message from the Installation Wizard when you have finished with Setup. For information about Setup log files, see [View and Read SQL Server Setup Log Files](#).
24. Repeat the previous steps to prepare the other nodes for the failover cluster. You can also use the autogenerated configuration file to run prepare on the other nodes. For more information, see [Install SQL Server 2016 Using a Configuration File](#).

Complete

1. After preparing all the nodes as described in the [prepare step](#), run Setup on one of the prepared nodes, preferably the one that owns the shared disk. On the **Advanced** page of the SQL Server Installation Center, click **Advanced cluster completion**.
2. The System Configuration Checker runs a discovery operation on your computer. To continue, Click **OK**. You can view the details on the screen by clicking **Show Details**, or as an HTML report by clicking **View detailed report**.
3. On the Setup Support Files page, click **Install** to install the Setup support files.
4. The System Configuration Checker verifies the system state of your computer before Setup continues. After the check is complete, click **Next** to continue. You can view the details on the screen by clicking **Show Details**, or as an HTML report by clicking **View detailed report**.
5. On the Language Selection page, you can specify the language for your instance of SQL Server if you are installing on a localized operating system and the installation media includes language packs for both English and the language corresponding to the operating system. For more information about cross-language support and installation considerations, see [Local Language Versions in SQL Server](#).

To continue, click **Next**.

6. Use the Cluster node configuration page to select the instance name prepared for clustering, and specify the network name for the new SQL Server failover cluster. This is the name that is used to identify your failover cluster on the network.

NOTE

This is known as the virtual SQL Server name in earlier versions of SQL Server failover clusters.

7. SQL Server setup runs one more set of rules that are based on the features you selected to validate your configuration.
8. Use the Cluster Resource Group page to specify the cluster resource group name where SQL Server virtual server resources will be located. To specify the SQL Server cluster resource group name. You have two options:
 - Use the list to specify an existing group to use.
 - Type the name of a new group to create. Be aware that the name "Available storage" is not a valid group name.
9. On the Cluster Disk Selection page, select the shared cluster disk resource for your SQL Server failover cluster. The cluster disk is where the SQL Server data will be put. More than one disk can be specified. The Available shared disks grid displays a list of available disks, whether each is qualified as a shared disk, and a description of each disk resource. Click **Next** to continue.

NOTE

The first drive is used as the default drive for all databases, but can be changed on the Database Engine or Analysis Services configuration pages.

10. On the Cluster Network Configuration page, specify the network resources for your failover cluster instance:
 - **Network Settings** — Specify the IP type and IP address for all the nodes and subnets for your failover cluster instance. You can specify multiple IP addresses for a multi-subnet failover cluster, but only one IP address per subnet is supported. Every prepared node should be an owner of at

least one IP address. If you have multiple subnets in your SQL Server failover cluster, you will be prompted to set the IP address resource dependency to OR.

Click **Next** to continue.

11. Work flow for the rest of this topic depends on the features that you have specified for your installation. You might not see all the pages, depending on your selections.

12. Use the Database Engine Configuration — Account Provisioning page to specify the following:

- Security Mode - select Windows Authentication or Mixed Mode Authentication for your instance of SQL Server. If you select Mixed Mode Authentication, you must provide a strong password for the built-in SQL Server system administrator account.

After a device establishes a successful connection to SQL Server, the security mechanism is the same for both Windows Authentication and Mixed Mode.

- SQL Server Administrators - You must specify at least one system administrator for the instance of SQL Server. To add the account under which SQL Server Setup is running, click **Add Current User**. To add or remove accounts from the list of system administrators, click **Add** or **Remove**, and then edit the list of users, groups, or computers that will have administrator privileges for the instance of SQL Server.

When you are finished editing the list, Click **OK**. Verify the list of administrators in the configuration dialog box. When the list is complete, click **Next**.

13. Use the Database Engine Configuration - Data Directories page to specify nondefault installation directories. To install to default directories, click **Next**.

IMPORTANT

If you specify nondefault installation directories, make sure that the installation folders are unique to this instance of SQL Server. None of the directories in this dialog box should be shared with directories from other instances of SQL Server. The data directories should be located on the shared cluster disk for the failover cluster.

14. Use the Analysis Services Configuration — Account Provisioning page to specify users or accounts that will have administrator permissions for Analysis Services. You must specify at least one system administrator for Analysis Services. To add the account under which SQL Server Setup is running, click **Add Current User**. To add or remove accounts from the list of system administrators, click **Add** or **Remove**, and then edit the list of users, groups, or computers that will have administrator privileges for Analysis Services.

When you are finished editing the list, Click **OK**. Verify the list of administrators in the configuration dialog box. When the list is complete, click **Next**.

15. Use the Analysis Services Configuration — Data Directories page to specify nondefault installation directories. To install to default directories, click **Next**.

IMPORTANT

If you specify nondefault installation directories, make sure that the installation folders are unique to this instance of SQL Server. None of the directories in this dialog box should be shared with directories from other instances of SQL Server. The data directories should be located on the shared cluster disk for the failover cluster.

16. The System Configuration Checker runs one more set of rules to validate your configuration with the SQL Server features that you have specified.

17. The Ready to Install page displays a tree view of installation options that were specified during Setup. To

continue, click **Install**. Setup will first install the required prerequisites for the selected features followed by the feature installation.

18. During installation, the Installation Progress page provides status so that you can monitor installation progress as Setup continues.
19. After installation, the **Complete** page provides a link to the summary log file for the installation and other important notes. To complete the SQL Server installation process, click **Close**. With this step, all the prepared nodes for the same failover cluster are now part of the completed SQL Server failover cluster.

Next Steps

Configure your new SQL Server installation — To reduce the attackable surface area of a system, SQL Server selectively installs and enables key services and features. For more information, see [Surface Area Configuration](#).

For more information about log file locations, see [View and Read SQL Server Setup Log Files](#).

See Also

[Install SQL Server 2016 from the Command Prompt](#)

Add or Remove Nodes in a SQL Server Failover Cluster (Setup)

3/29/2017 • 5 min to read • [Edit Online](#)

Use this procedure to manage nodes to an existing SQL Server failover cluster instance.

To update or remove a SQL Server failover cluster, you must be a local administrator with permission to log in as a service on all nodes of the failover cluster. For local installations, you must run Setup as an administrator. If you install SQL Server from a remote share, you must use a domain account that has read and execute permissions on the remote share.

To add a node to an existing SQL Server failover cluster, you must run SQL Server Setup on the node that is to be added to the SQL Server failover cluster instance. Do not run Setup on the active node.

To remove a node from an existing SQL Server failover cluster, you must run SQL Server Setup on the node that is to be removed from the SQL Server failover cluster instance.

To view procedural steps to add or remove nodes, select one of the following operations:

- [Add a node to an existing SQL Server failover cluster](#)
- [Remove a node from an existing SQL Server failover cluster](#)

IMPORTANT

The operating system drive letter for SQL Server install locations must match on all the nodes added to the SQL Server failover cluster.

Add Node

To add a node to an existing SQL Server failover cluster

1. Insert the SQL Server installation media, and from the root folder, double-click Setup.exe. To install from a network share, navigate to the root folder on the share, and then double-click Setup.exe.
2. The Installation Wizard will launch the SQL Server Installation Center. To add a node to an existing failover cluster instance, click **Installation** in the left-hand pane. Then, select **Add node to a SQL Server failover cluster**.
3. The System Configuration Checker will run a discovery operation on your computer. To continue, Click **OK**.
4. On the Language Selection page, you can specify the language for your instance of SQL Server if you are installing on a localized operating system and the installation media includes language packs for both English and the language corresponding to the operating system. For more information about cross-language support and installation considerations, see [Local Language Versions in SQL Server](#).

To continue, click **Next**.

5. On the Product key page, specify the PID key for a production version of the product. Note that the product key you enter for this installation must be for the same SQL Server edition as that which is installed on the active node.
6. On the License Terms page, read the license agreement, and then select the check box to accept the

licensing terms and conditions. To help improve SQL Server, you can also enable the feature usage option and send reports to Microsoft. To continue, click **Next**. To end Setup, click **Cancel**.

7. The System Configuration Checker will verify the system state of your computer before Setup continues. After the check is complete, click **Next** to continue.
8. On the Cluster Node Configuration page, use the drop-down box to specify the name of the SQL Server failover cluster instance that will be modified during this Setup operation.
9. On the Server Configuration — Service Accounts page, specify login accounts for SQL Server services. The actual services that are configured on this page depend on the features you selected to install. For failover cluster installations, account name and startup type information will be pre-populated on this page based on settings provided for the active node. You must provide passwords for each account. For more information, see [Server Configuration - Service Accounts](#) and [Configure Windows Service Accounts and Permissions](#).

Security Note Do not use a blank password. Use a strong password.

When you are finished specifying login information for SQL Server services, click **Next**.

10. On the Reporting page, specify the information you would like to send to Microsoft to improve SQL Server. By default, option for error reporting is enabled.
11. The System Configuration Checker will run one more set of rules to validate your computer configuration with the SQL Server features you have specified.
12. The Ready to Add Node page displays a tree view of installation options that were specified during Setup.
13. Add Node Progress page provides status so you can monitor installation progress as Setup proceeds.
14. After installation, the Complete page provides a link to the summary log file for the installation and other important notes. To complete the SQL Server installation process, click **Close**.
15. If you are instructed to restart the computer, do so now. It is important to read the message from the Installation Wizard when you are done with Setup. For more information about Setup log files, see [View and Read SQL Server Setup Log Files](#).

Remove Node

To remove a node from an existing SQL Server failover cluster

1. Insert the SQL Server installation media. From the root folder, double-click setup.exe. To install from a network share, navigate to the root folder on the share, and then double-click Setup.exe.
2. The Installation Wizard launches the SQL Server Installation Center. To remove a node to an existing failover cluster instance, click **Maintenance** in the left-hand pane, and then select **Remove node from a SQL Server failover cluster**.
3. The System Configuration Checker will run a discovery operation on your computer. To continue, Click **OK**.
4. After you click install on the Setup Support Files page, the System Configuration Checker verifies the system state of your computer before Setup continues. After the check is complete, click **Next** to continue.
5. On the Cluster Node Configuration page, use the drop-down box to specify the name of the SQL Server failover cluster instance to be modified during this Setup operation. The node to be removed is listed in the **Name of this node** field.
6. The Ready to Remove Node page displays a tree view of options that were specified during Setup. To continue, click **Remove**.

7. During the remove operation, the Remove Node Progress page provides status.
8. The Complete page provides a link to the summary log file for the remove node operation and other important notes. To complete the SQL Server remove node, click **Close**. For more information about Setup log files, see [View and Read SQL Server Setup Log Files](#).

See Also

[View and Read SQL Server Setup Log Files](#)

Install Client Tools on a SQL Server Failover Cluster

3/29/2017 • 2 min to read • [Edit Online](#)

Client tools such as SQL Server Management Studio are shared features common across all instances on the same machine. They are backward compatible, with supported SQL Server versions that can be installed side by side. Only one version of the client tool exists on a node at a time.

If the SQL Server client tools are installed during setup on first node of the SQL Server cluster, they are automatically added to any nodes that may be added later to the instance of SQL Server using Add Node.

IMPORTANT

SQL Server Books Online is not automatically added to the additional nodes added to the SQL Server cluster using Add Node. SQL Server Books Online can be installed manually on the nodes that you wish to have a local copy of SQL Server Books Online.

If you do not install the SQL Server client tools during the initial installation of the SQL Server cluster, you can install it later as described in the procedures below.

Installation procedures

Installing SQL Server Client Tools Using the Setup User Interface

1. Insert the SQL Server installation media. From the root installation folder, double click Setup.exe. To install from the network share, locate the root folder on the share, and then double-click Setup.exe.
2. On the **Installation** page, click **New SQL Server stand-alone installation or add Features to an existing installation**. Do not click **New SQL Server failover cluster installation**.
3. The system configuration checker verifies the system state of your computer before Setup will continue.
4. On the **Installation Type** page, click **Perform a new installation of SQL Server 2016**.
5. On the **Feature Selection** page, select the tools that you want to install and follow through the rest of the steps of the Setup process.

Installing SQL Server client tools at the command prompt

1. To install SQL Server client tools and SQL Server Books Online, run the following command:
Setup.exe/q/Action=Install /Features=Tools
2. To install only the basic SQL Server Management tools run the following command:
Setup.exe/q/Action=Install Features=SSMS. This will install Management Studio support for SQL Server Database Engine, SQL Server Express, sqlcmd utility, and the SQL Server Powershell provider.
3. To install the complete SQL Server Management tools, run the following command:
Setup.exe/q/Action=Install /Features=ADV_SSMS. For more information about parameter values for the features, see [Install SQL Server 2016 from the Command Prompt](#).

Uninstalling SQL Server Client Tools

They appear in Add or Remove programs in Control Panel as **Microsoft SQL Server 2016**, and can be removed from there. When you use Remove Node to uninstall an instance of SQL Server from the failover cluster, the client components are not uninstalled at the same time.

See Also

[View and Read SQL Server Setup Log Files](#)

Remove a SQL Server Failover Cluster Instance (Setup)

3/29/2017 • 1 min to read • [Edit Online](#)

Use this procedure to uninstall a SQL Server failover clustered instance.

IMPORTANT

To update or remove a SQL Server failover cluster, you must be a local administrator with permission to login as a service on all nodes of the failover cluster.

Before you begin

Consider the following important points before you uninstall a SQL Server failover cluster:

- If SQL Server Native Client is uninstalled by accident, SQL Server resources will fail to start. To reinstall SQL Server Native Client, run the SQL Server Setup program to install SQL Server prerequisites.
- If you uninstall a failover cluster that has more than one SQL IP cluster resource, you must remove the additional SQL IP resources using cluster administrator.

For information about command prompt syntax, see [Install SQL Server 2016 from the Command Prompt](#).

To uninstall a SQL Server failover cluster

1. To uninstall a SQL Server failover cluster, use the Remove Node functionality provided by SQL Server Setup to remove each node individually. For more information, see [Add or Remove Nodes in a SQL Server Failover Cluster \(Setup\)](#).

See Also

[View and Read SQL Server Setup Log Files](#)

Rename a SQL Server Failover Cluster Instance

3/29/2017 • 3 min to read • [Edit Online](#)

When a SQL Server instance is part of a failover cluster, the process of renaming the virtual server differs from that of renaming a stand-alone instance. For more information, see [Rename a Computer that Hosts a Stand-Alone Instance of SQL Server](#).

The name of the virtual server is always the same as the name of the SQL Network Name (the SQL Virtual Server Network Name). Although you can change the name of the virtual server, you cannot change the instance name. For example, you can change a virtual server named VS1\instance1 to some other name, such as SQL35\instance1, but the instance portion of the name, instance1, will remain unchanged.

Before you begin the renaming process, review the items below.

- SQL Server does not support renaming servers involved in replication, except in the case of using log shipping with replication. The secondary server in log shipping can be renamed if the primary server is permanently lost. For more information, see [Log Shipping and Replication \(SQL Server\)](#).
- When renaming a virtual server that is configured to use database mirroring, you must turn off database mirroring before the renaming operation, and then re-establish database mirroring with the new virtual server name. Metadata for database mirroring will not be updated automatically to reflect the new virtual server name.

To rename a virtual server

1. Using Cluster Administrator, change the SQL Network Name to the new name.
2. Take the network name resource offline. This takes the SQL Server resource and other dependent resources offline as well.
3. Bring the SQL Server resource back online.

Verify the Renaming Operation

After a virtual server has been renamed, any connections that used the old name must now connect using the new name.

To verify that the renaming operation has completed, select information from either **@@servername** or **sys.servers**. The **@@servername** function will return the new virtual server name, and the **sys.servers** table will show the new virtual server name. To verify that the failover process is working correctly with the new name, the user should also try to fail the SQL Server resource over to the other nodes.

For connections from any node in the cluster, the new name can be used almost immediately. However, for connections using the new name from a client computer, the new name cannot be used to connect to the server until the new name is visible to that client computer. The length of time required for the new name to be propagated across a network can be a few seconds, or as long as 3 to 5 minutes, depending on the network configuration; additional time may be required before the old virtual server name is no longer visible on the network.

To minimize network propagation delay of a virtual server renaming operation, use the following steps:

To minimize network propagation delay

1. Issue the following commands from a command prompt on the server node:

```
ipconfig /flushdns  
ipconfig /registerdns  
nbtstat -RR
```

Additional considerations after the Renaming Operation

After we rename the network name of failover cluster we need to verify and perform the following instructions to enable all scenarios in SQL Server Agent and Analysis Services.

SQL Server Agent Service: Verify and perform the below additional actions for SQL Server Agent Service:

- Fix the registry settings if SQL Agent is configured for event forwarding. For more information, see [Designate an Events Forwarding Server \(SQL Server Management Studio\)](#).
- Fix the master server (MSX) and target servers (TSX) instance names when machines / cluster network name is renamed. For more information, see the following topics:
 - [Defect Multiple Target Servers from a Master Server](#)
 - [Create a Multiserver Environment](#)
- Reconfigure the Log Shipping so that updated server name is used to backup and restore logs. For more information, see the following topics:
 - [Configure Log Shipping \(SQL Server\)](#)
 - [Remove Log Shipping \(SQL Server\)](#)
- Update the Jobsteps that depend on server name. For more information, see [Manage Job Steps](#).

See Also

[Rename a Computer that Hosts a Stand-Alone Instance of SQL Server](#)

High Availability Solutions (SQL Server)

3/29/2017 • 2 min to read • [Edit Online](#)

This topic introduces several SQL Server high-availability solutions that improve the availability of servers or databases. A high-availability solution masks the effects of a hardware or software failure and maintains the availability of applications so that the perceived downtime for users is minimized.

Note! Want to know which SQL Server editions support a given high availability solution? See the "High Availability (Always On)" section of [Features Supported by the Editions of SQL Server 2016](#).

Overview of SQL Server High-Availability Solutions

SQL Server provides several options for creating high availability for a server or database. The high-availability options include the following:

- Always On Failover Cluster Instances

As part of the SQL Server Always On offering, Always On Failover Cluster Instances leverages Windows Server Failover Clustering (WSFC) functionality to provide local high availability through redundancy at the server-instance level—a *failover cluster instance* (FCI). An FCI is a single instance of SQL Server that is installed across Windows Server Failover Clustering (WSFC) nodes and, possibly, across multiple subnets. On the network, an FCI appears to be an instance of SQL Server running on a single computer, but the FCI provides failover from one WSFC node to another if the current node becomes unavailable.

For more information, see [Always On Failover Cluster Instances \(SQL Server\)](#).

- Always On availability groups

Always On availability groups is an enterprise-level high-availability and disaster recovery solution introduced in SQL Server 2012 to enable you to maximize availability for one or more user databases. Always On availability groups requires that the SQL Server instances reside on Windows Server Failover Clustering (WSFC) nodes. For more information, see [Always On Availability Groups \(SQL Server\)](#).

Note! An FCI can leverage Always On availability groups to provide remote disaster recovery at the database level. For more information, see [Failover Clustering and Always On Availability Groups \(SQL Server\)](#).

- Database mirroring. **Note!** This feature will be removed in a future version of Microsoft SQL Server. Avoid using this feature in new development work, and plan to modify applications that currently use this feature. We recommend that you use Always On availability groups instead.

Database mirroring is a solution to increase database availability by supporting almost instantaneous failover. Database mirroring can be used to maintain a single standby database, or *mirror database*, for a corresponding production database that is referred to as the *principal database*. For more information, see [Database Mirroring \(SQL Server\)](#).

- Log shipping

Like Always On availability groups and database mirroring, log shipping operates at the database level. You can use log shipping to maintain one or more warm standby databases (referred to as *secondary databases*) for a single production database that is referred to as the *primary database*. For more information about log shipping, see [About Log Shipping \(SQL Server\)](#).

Recommended Solutions for Using SQL Server to Protect Data

Our recommendation for providing data protection for your SQL Server environment:

- For data protection through a third-party shared disk solution (a SAN), we recommend that you use Always On Failover Cluster Instances.
- For data protection through SQL Server, we recommend that you use Always On availability groups.

We recommend using log shipping if you are running an edition of SQL Server that does not support Always On availability groups. For information about which editions of SQL Server support Always On availability groups, see the "High Availability (Always On)" section of [Features Supported by the Editions of SQL Server 2016](#).

See Also

[Windows Server Failover Clustering \(WSFC\) with SQL Server](#)

[Database Mirroring: Interoperability and Coexistence \(SQL Server\)](#)

[Deprecated Database Engine Features in SQL Server 2016](#)

Automatic Page Repair (Availability Groups: Database Mirroring)

3/31/2017 • 5 min to read • [Edit Online](#)

Automatic page repair is supported by database mirroring and by Always On availability groups. After certain types of errors corrupt a page, making it unreadable, a database mirroring partner (principal or mirror) or an availability replica (primary or secondary) attempts to automatically recover the page. The partner/replica that cannot read the page requests a fresh copy of the page from its partner or from another replica. If this request succeeds, the unreadable page is replaced by the readable copy, and this usually resolves the error.

Generally speaking, database mirroring and Always On availability groups handle I/O errors in equivalent ways. The few differences are explicitly called out here.

NOTE

Automatic page repair differs from DBCC repair. All of the data is preserved by an automatic page repair. In contrast, correcting errors by using the DBCC REPAIR_ALLOW_DATA_LOSS option might require that some pages, and therefore data, be deleted.

- [Error types that cause an automatic page-repair attempt](#)
- [Page types that cannot be automatically repaired](#)
- [Handling i/o errors on the principal/primary database](#)
- [Handling i/o errors on the mirror/secondary database](#)
- [Developer best practice](#)
- [How to: view automatic page-repair attempts](#)

Error Types That Cause an Automatic Page-Repair Attempt

Database mirroring automatic page repair tries to repair only pages in a data file on which an operation has failed for one of the errors listed in the following table.

ERROR NUMBER	DESCRIPTION	INSTANCES THAT CAUSE AUTOMATIC PAGE-REPAIR ATTEMPT
823	Action is taken only if the operating system performed a cyclic redundancy check (CRC) that failed on the data.	ERROR_CRC. The operating-system value for this error is 23.
824	Logical errors.	Logical data errors, such as torn write or bad page checksum.
829	A page has been marked as restore pending.	All.

To view recent 823 CRC errors and 824 errors, see the [suspect_pages](#) table in the [msdb](#) database.

Page Types That Cannot Be Automatically Repaired

Automatic page repair cannot repair the following control page types:

- File header page (page ID 0).
- Page 9 (the database boot page).
- Allocation pages: Global Allocation Map (GAM) pages, Shared Global Allocation Map (SGAM) pages, and Page Free Space (PFS) pages.

Handling I/O Errors on the Principal/Primary Database

On the principal/primary database, automatic page repair is tried only when the database is in the SYNCHRONIZED state and the principal/primary is still sending log records for the database to the mirror/secondary. The basic sequence of actions in an automatic page-repair attempt are as follows:

1. When a read error occurs on a data page in the principal/primary database, the principal/primary inserts a row in the [suspect_pages](#) table with the appropriate error status. For database mirroring, the principal then requests a copy of the page from the mirror. For Always On availability groups, the primary broadcasts the request to all the secondaries and gets the page from the first to respond. The request specifies the page ID and the LSN that is currently at the end of the flushed log. The page is marked as *restore pending*. This makes it inaccessible during the automatic page-repair attempt. Attempts to access this page during the repair attempt will fail with error 829 (restore pending).
2. After receiving the page request, the mirror/secondary waits until it has redone the log up to the LSN specified in the request. Then, the mirror/secondary tries to access the page in its copy of the database. If the page can be accessed, the mirror/secondary sends the copy of the page to the principal/primary. Otherwise, the mirror/secondary returns an error to the principal/primary, and the automatic page-repair attempt fails.
3. The principal/primary processes the response that contains the fresh copy of the page.
4. After the automatic page-repair attempt fixes a suspect page, the page is marked in the **suspect_pages** table as restored (**event_type** = 5).
5. If the page I/O error caused any [deferred transactions](#), after you repair the page, the principal/primary tries to resolve those transactions.

Handling I/O Errors on the Mirror/Secondary Database

I/O errors on data pages that occur on the mirror/secondary database are handled in generally the same way by database mirroring and by Always On availability groups.

1. With database mirroring, if the mirror encounters one or more page I/O errors when it redoes a log record, the mirroring session enters the SUSPENDED state. With Always On availability groups, if a secondary replica encounters one or more page I/O errors when it redoes a log record, the secondary database enters the SUSPENDED state. At that point, the mirror/secondary inserts a row in the **suspect_pages** table with the appropriate error status. The mirror/secondary then requests a copy of the page from the principal/primary.
2. The principal/primary tries to access the page in its copy of the database. If the page can be accessed, the principal/primary sends the copy of page to the mirror/secondary.
3. If the mirror/secondary receives copies of every page it has requested, the mirror/secondary tries to resume the mirroring session. If an automatic page-repair attempt fixes a suspect page, the page is marked in the **suspect_pages** table as restored (**event_type** = 4).

If a mirror/secondary does not receive a page that it requested from the principal/primary, the automatic page-repair attempt fails. With database mirroring, the mirroring session remains suspended. With Always

On availability groups, the secondary database remains suspended. If the mirroring session or secondary database is resumed manually, the corrupted pages will be hit again during the synchronization phase.

Developer Best Practice

An automatic page repair is an asynchronous process that runs in the background. Therefore, a database operation that requests an unreadable page fails and returns the error code for whatever condition caused the failure. When developing an application for a mirrored database or an availability database, you should intercept exceptions for failed operations. If the SQL Server error code is 823, 824, or 829, you should retry the operation later.

How To: View Automatic Page-Repair Attempts

The following dynamic management views return rows for the latest automatic page-repair attempts on a given availability database or mirrored database, with a maximum of 100 rows per database.

- **Always On Availability Groups:**

[sys.dm_hadr_auto_page_repair \(Transact-SQL\)](#)

Returns a row for every automatic page-repair attempt on any availability database on an availability replica that is hosted for any availability group by the server instance.

- **Database mirroring:**

[sys.dm_db_mirroring_auto_page_repair \(Transact-SQL\)](#)

Returns a row for every automatic page-repair attempt on any mirrored database on the server instance.

See Also

[Manage the suspect_pages Table \(SQL Server\)](#)

[Overview of Always On Availability Groups \(SQL Server\)](#)

[Database Mirroring \(SQL Server\)](#)

Management of Logins and Jobs After Role Switching (SQL Server)

3/29/2017 • 2 min to read • [Edit Online](#)

When deploying a high-availability or disaster-recovery solution for a SQL Server database, it is important to reproduce relevant information that is stored for the database in the **master** or **msdb** databases. Typically, the relevant information includes the jobs of the primary/principal database and the logins of users or processes that need to connect to the database. You should duplicate this information on any instance of SQL Server that hosts a secondary/mirror database. If possible after roles are switched, it is best to programmatically reproduce the information on the new primary/principal database.

Logins

On every server instances that hosts a copy of the database, you should reproduce the logins that have permission to access the principal database. When the primary/principal role switches, only users whose logins exist on the new primary/principal server instance can access the new primary/principal database. Users whose logins are not defined on the new primary/principal server instance are orphaned and cannot access the database.

If a user is orphaned, create the login on the new primary/principal server instance and run [sp_change_users_login](#). For more information, see [Troubleshoot Orphaned Users \(SQL Server\)](#).

Logins Of Applications That Use SQL Server Authentication or a Local Windows Login

If an application uses SQL Server Authentication or a local Windows login, mismatched SIDs can prevent the application's login from resolving on a remote instance of SQL Server. The mismatched SIDs cause the login to become an orphaned user on the remote server instance. This issue can occur when an application connects to a mirrored or log shipping database after a failover or to a replication subscriber database that was initialized from a backup.

To prevent this issue, we recommend that you take preventative measures when you set up such an application to use a database that is hosted by a remote instance of SQL Server. Prevention involves transferring the logins and the passwords from the local instance of SQL Server to the remote instance of SQL Server. For more information about how to prevent this issue, see KB article 918992 —[How to transfer logins and passwords between instances of SQL Server](#)).

NOTE

This problem affects Windows local accounts on different computers. However, this problem does not occur for domain accounts because the SID is the same on each of the computers.

For more information, see [Orphaned Users with Database Mirroring and Log Shipping](#) (a Database Engine blog).

Jobs

Jobs, such as backup jobs, require special consideration. Typically, after a role switch, the database owner or system administrator must re-create the jobs for the new primary/principal database.

When the former primary/principal server instance is available, you should delete the original jobs on that instance of SQL Server. Note that jobs on the current mirror database fail because it is in the RESTORING state, making it unavailable.

NOTE

Different instances of SQL Server might be configured differently, with different drive letters or such. The jobs for each partner must allow for any such differences.





See Also

[Manage Metadata When Making a Database Available on Another Server Instance \(SQL Server\)](#)

[Troubleshoot Orphaned Users \(SQL Server\)](#)

Troubleshoot Orphaned Users (SQL Server)

3/29/2017 • 4 min to read • [Edit Online](#)

THIS TOPIC APPLIES TO:  SQL Server (starting with 2008)  Azure SQL Database  Azure SQL Data Warehouse  Parallel Data Warehouse

Orphaned users in SQL Server occur when a database user is based on a login in the **master** database, but the login no longer exists in **master**. This can occur when the login is deleted, or when the database is moved to another server where the login does not exist. This topic describes how to find orphaned users, and remap them to logins.

NOTE

Reduce the possibility of orphaned users by using contained database users for databases that might be moved. For more information, see [Contained Database Users - Making Your Database Portable](#).

Background

To connect to a database on an instance of SQL Server using a security principal (database user identity) based on a login, the principal must have a valid login in the **master** database. This login is used in the authentication process that verifies the principal's identity and determines if the principal is allowed to connect to the instance of SQL Server. The SQL Server logins on a server instance are visible in the **sys.server_principals** catalog view and the **sys.sql_logins** compatibility view.

SQL Server logins access individual databases as "database user" that is mapped to the SQL Server login. There are three exceptions to this rule:

- Contained database users

Contained database users authenticate at the user-database level and are not associated with logins. This is recommended because the databases are more portable and contained database users cannot become orphaned. However they must be recreated for each database. This might be impractical in an environment with many databases.

- The **guest** account.

When enabled in the database, this account permits SQL Server logins that are not mapped to a database user to enter the database as the **guest** user. The **guest** account is disabled by default.

- Microsoft Windows group memberships.

A SQL Server login created from a Windows user can enter a database if the Windows user is a member of a Windows group that is also a user in the database.

Information about the mapping of a SQL Server login to a database user is stored within the database. It includes the name of the database user and the SID of the corresponding SQL Server login. The permissions of this database user are applied for authorization in the database.

A database user (based on a login) for which the corresponding SQL Server login is undefined or is incorrectly defined on a server instance cannot log in to the instance. Such a user is said to be an *orphaned user* of the database on that server instance. Orphaning can happen if the database user is mapped to a login SID that is not present in the **master** instance. A database user can become orphaned after a database is restored or attached to a different instance of SQL Server where the login was never created. A database

user can also become orphaned if the corresponding SQL Server login is dropped. Even if the login is recreated, it will have a different SID, so the database user will still be orphaned.

To Detect Orphaned Users

For SQL Server and PDW

To detect orphaned users in SQL Server based on missing SQL Server authentication logins, execute the following statement in the user database:

```
SELECT dp.type_desc, dp.SID, dp.name AS user_name
FROM sys.database_principals AS dp
LEFT JOIN sys.server_principals AS sp
    ON dp.SID = sp.SID
WHERE sp.SID IS NULL
    AND authentication_type_desc = 'INSTANCE';
```

The output lists the SQL Server authentication users and corresponding security identifiers (SID) in the current database that are not linked to any SQL Server login.

For SQL Database and SQL Data Warehouse

The `sys.server_principals` table is not available in SQL Database or SQL Data Warehouse. Identify orphaned users in those environments with the following steps:

1. Connect to the `master` database and select the SID's for the logins with the following query:

```
SELECT sid
FROM sys.sql_logins
WHERE type = 'S';
```

2. Connect to the user database and review the SID's of the users in the `sys.database_principals` table, by using the following query:

```
SELECT name, sid, principal_id
FROM sys.database_principals
WHERE type = 'S'
    AND name NOT IN ('guest', 'INFORMATION_SCHEMA', 'sys')
    AND authentication_type_desc = 'INSTANCE';
```

3. Compare the two lists to determine if there are user SID's in the user database `sys.database_principals` table which are not matched by login SID's in the master database `sql_logins` table.

To Resolve an Orphaned User

In the master database, use the [CREATE LOGIN](#) statement with the SID option to recreate a missing login, providing the `SID` of the database user obtained in the previous section:

```
CREATE LOGIN <login_name>
WITH PASSWORD = '<use_a_strong_password_here>',
SID = <SID>;
```

To map an orphaned user to a login which already exists in **master**, execute the [ALTER USER](#) statement in the user database, specifying the login name.


```
ALTER USER <user_name> WITH Login = <login_name>;
```

When you recreate a missing login, the user can access the database using the password provided. Then the user can alter the password of the login account by using the ALTER LOGIN statement.

```
ALTER LOGIN <login_name> WITH PASSWORD = '<enterStrongPasswordHere>';
```

IMPORTANT

Any login can change its own password. Only logins with the `ALTER ANY LOGIN` permission can change the password of another user's login. However, only members of the **sysadmin** role can modify passwords of **sysadmin** role members.

The deprecated procedure [sp_change_users_login](#) also works with orphaned users. `sp_change_users_login` cannot be used with SQL Database.

See Also

[CREATE LOGIN \(Transact-SQL\)](#)

[ALTER USER \(Transact-SQL\)](#)

[CREATE USER \(Transact-SQL\)](#)

[sys.database_principals \(Transact-SQL\)](#)

[sys.server_principals \(Transact-SQL\)](#)

[sp_change_users_login \(Transact-SQL\)](#)

[sp_addlogin \(Transact-SQL\)](#)

[sp_grantlogin \(Transact-SQL\)](#)

[sp_password \(Transact-SQL\)](#)

[sys.sysusers \(Transact-SQL\)](#)

[sys.sql_logins sys.syslogins \(Transact-SQL\)](#)

Windows Server Failover Clustering (WSFC) with SQL Server

3/29/2017 • 11 min to read • [Edit Online](#)

A *Windows Server Failover Clustering* (WSFC) cluster is a group of independent servers that work together to increase the availability of applications and services. SQL Server 2016 takes advantage of WSFC services and capabilities to support Always On availability groups and SQL Server Failover Cluster Instances.

Terms and Definitions

WSFC cluster

A Windows Server Failover Clustering (WSFC) cluster is a group of independent servers that work together to increase the availability of applications and services.

Failover cluster instance

An instance of a Windows service that manages an IP address resource, a network name resource, and additional resources that are required to run one or more applications or services. Clients can use the network name to access the resources in the group, similar to using a computer name to access the services on a physical server. However, because a failover cluster instance is a group, it can be failed over to another node without affecting the underlying name or address.

Node

A Microsoft Windows Server system that is an active or inactive member of a server cluster.

Cluster resource

A physical or logical entity that can be owned by a node, brought online and taken offline, moved between nodes, and managed as a cluster object. A cluster resource can be owned by only a single node at any point in time.

Resource group

A collection of cluster resources managed as a single cluster object. Typically a resource group contains all of the cluster resources that are required to run a specific application or service. Failover and failback always act on resource groups.

Resource dependency

A resource on which another resource depends. If resource A depends on resource B, then B is a dependency of A.

Network name resource

A logical server name that is managed as a cluster resource. A network name resource must be used with an IP address resource.

Preferred owner

A node on which a resource group prefers to run. Each resource group is associated with a list of preferred owners sorted in order of preference. During automatic failover, the resource group is moved to the next preferred node in the preferred owner list.

Possible owner

A secondary node on which a resource can run. Each resource group is associated with a list of possible owners. Resource groups can fail over only to nodes that are listed as possible owners.

Quorum mode

The quorum configuration in a failover cluster that determines the number of node failures that the cluster can sustain.

Forced quorum

The process to start the cluster even though only a minority of the elements that are required for quorum are in communication.

Overview of Windows Server Failover Clustering

Windows Server Failover Clustering provides infrastructure features that support the high-availability and disaster recovery scenarios of hosted server applications such as Microsoft SQL Server and Microsoft Exchange. If a cluster node or service fails, the services that were hosted on that node can be automatically or manually transferred to another available node in a process known as *failover*.

The nodes in the WSFC cluster work together to collectively provide these types of capabilities:

- **Distributed metadata and notifications.** WSFC service and hosted application metadata is maintained on each node in the cluster. This metadata includes WSFC configuration and status in addition to hosted application settings. Changes to a node's metadata or status are automatically propagated to the other nodes in the cluster.
- **Resource management.** Individual nodes in the cluster may provide physical resources such as direct-attached storage, network interfaces, and access to shared disk storage. Hosted applications register themselves as a cluster resource, and may configure startup and health dependencies upon other resources.
- **Health monitoring.** Inter-node and primary node health detection is accomplished through a combination of heartbeat-style network communications and resource monitoring. The overall health of the cluster is determined by the votes of a quorum of nodes in the cluster.
- **Failover coordination.** Each resource is configured to be hosted on a primary node, and each can be automatically or manually transferred to one or more secondary nodes. A health-based failover policy controls automatic transfer of resource ownership between nodes. Nodes and hosted applications are notified when failover occurs so that they may react appropriately.

For more information, see: [Failover Clustering Overview - Windows Server](#)

SQL Server Always On Technologies and WSFC

SQL Server 2016 *Always On* is a high availability and disaster recovery solution that takes advantage of WSFC. Always On provides an integrated, flexible solution that increases application availability, provides better returns on hardware investments, and simplifies high availability deployment and management.

Both Always On availability groups and Always On Failover Cluster Instances use WSFC as a platform technology, registering components as WSFC cluster resources. Related resources are combined into a *resource group*, which can be made dependent upon other WSFC cluster resources. The WSFC cluster service can then sense and signal the need to restart the SQL Server instance or automatically fail it over to a different server node in the WSFC cluster.

IMPORTANT!! To take full advantage of SQL Server Always On technologies, you should apply several WSFC-related prerequisites.

For more information, see: [Prerequisites, Restrictions, and Recommendations for Always On Availability Groups \(SQL Server\)](#)

Instance-level High Availability with Always On Failover Cluster Instances

An Always On *Failover Cluster Instance* (FCI) is a SQL Server instance that is installed across nodes in a WSFC cluster. This type of instance depends on resources for storage and virtual network name. The storage can use Fibre Channel, iSCSI, FCoE, or SAS for shared disk storage, or use locally attached storage with [Storage Spaces Direct \(S2D\)](#). The virtual network name resource depends on one or more virtual IP addresses, each in a different

subnet. The SQL Server service and the SQL Server Agent service are also resources, and both are dependent upon the storage and virtual network name resources.

In the event of a failover, the WSFC service transfers ownership of instance's resources to a designated failover node. The SQL Server instance is then re-started on the failover node, and databases are recovered as usual. At any given moment, only a single node in the cluster can host the FCI and underlying resources.

NOTE: An Always On Failover Cluster Instance requires symmetrical shared disk storage such as a storage area network (SAN) or SMB file share. The shared disk storage volumes must be available to all potential failover nodes in the WSFC cluster.

For more information, see: [Always On Failover Cluster Instances \(SQL Server\)](#)

Database-level High Availability with Always On availability groups

An *availability group* is a set of user databases that fail over together. An availability group consists of a primary *availability replica* and one to four secondary replicas that are maintained through SQL Server log-based data movement for data protection without the need for shared storage. Each replica is hosted by an instance of SQL Server on a different node of the WSFC cluster. The availability group and a corresponding virtual network name are registered as resources in the WSFC cluster.

An *availability group listener* on the primary replica's node responds to incoming client requests to connect to the virtual network name, and based on attributes in the connection string, it redirects each request to the appropriate SQL Server instance.

In the event of a failover, instead of transferring ownership of shared physical resources to another node, WSFC is leveraged to reconfigure a secondary replica on another SQL Server instance to become the availability group's primary replica. The availability group's virtual network name resource is then transferred to that instance.

At any given moment, only a single SQL Server instance may host the primary replica of an availability group's databases, all associated secondary replicas must each reside on a separate instance, and each instance must reside on separate physical nodes.

NOTE: Always On availability groups do not require deployment of a Failover Cluster Instance or use of symmetric shared storage (SAN or SMB).

A Failover Cluster Instance (FCI) may be used together with an availability group to enhance the availability of an availability replica. However, to prevent potential race conditions in the WSFC cluster, automatic failover of the availability group is not supported to or from an availability replica that is hosted on a FCI.

For more information, see: [Overview of Always On Availability Groups \(SQL Server\)](#)

WSFC Health Monitoring and Failover

High availability for an Always On solution is accomplished through proactive health monitoring of physical and logical WSFC cluster resources, together with automatic failover onto and re-configuration of redundant hardware. A system administrator can also initiate a *manual failover* of an availability group or SQL Server instance from one node to another.

Failover Policies for Nodes, Failover Cluster Instances, and Availability Groups

A *failover policy* is configured at the WSFC cluster node, the SQL Server Failover Cluster Instance (FCI), and the availability group levels. These policies, based on the severity, duration, and frequency of unhealthy cluster resource status and node responsiveness, can trigger a service restart or an *automatic failover* of cluster resources from one node to another, or can trigger the move of an availability group primary replica from one SQL Server instance to another.

Failover of an availability group replica does not affect the underlying SQL Server instance. Failover of a FCI moves the hosted availability group replicas with the instance.

For more information, see: [Failover Policy for Failover Cluster Instances](#)

WSFC Resource Health Detection

Each resource in a WSFC cluster node can report its status and health, periodically or on-demand. A variety of circumstances may indicate resource failure; e.g. power failure, disk or memory errors, network communication errors, or non-responsive services.

WSFC cluster resources such as networks, storage, or services can be made dependent upon one another. The cumulative health of a resource is determined by successively rolling up its health with the health of each of its resource dependencies.

WSFC Inter-node Health Detection and Quorum Voting

Each node in a WSFC cluster participates in periodic heartbeat communication to share the node's health status with the other nodes. Unresponsive nodes are considered to be in a failed state.

A *quorum* node set is a majority of the voting nodes and witnesses in the WSFC cluster. The overall health and status of a WSFC cluster is determined by a periodic *quorum vote*. The presence of a quorum means that the cluster is healthy and able to provide node-level fault tolerance.

A *quorum mode* is configured at the WSFC cluster level that dictates the methodology used for quorum voting and when to perform an automatic failover or take the cluster offline.

TIP!! It is best practice to always have an odd number of quorum votes in a WSFC cluster. For the purposes of quorum voting, SQL Server does not have to be installed on all nodes in the cluster. An additional server can act as a quorum member, or the WSFC quorum model can be configured to use a remote file share as a tie-breaker.

For more information, see: [WSFC Quorum Modes and Voting Configuration \(SQL Server\)](#)

Disaster Recovery Through Forced Quorum

Depending upon operational practices and WSFC cluster configuration, you can incur both automatic and manual failovers, and still maintain a robust, fault-tolerant SQL Server Always On solution. However, if a quorum of the eligible voting nodes in the WSFC cluster cannot communicate with one another, or if the WSFC cluster otherwise fails health validation, then the WSFC cluster may go offline.

If the WSFC cluster goes offline because of an unplanned disaster, or due to a persistent hardware or communications failure, then manual administrative intervention is required to *force a quorum* and bring the surviving cluster nodes back online in a non-fault-tolerant configuration.

Afterwards, a series of steps must also be taken to reconfigure the WSFC cluster, recover the affected database replicas, and to re-establish a new quorum.

For more information, see: [WSFC Disaster Recovery through Forced Quorum \(SQL Server\)](#)

Relationship of SQL Server AlwaysOn Components to WSFC

Several layers of relationships exist between SQL Server Always On and WSFC features and components.

Always On availability groups are hosted on SQL Server instances.

A client request that specifies a logical availability group listener network name to connect to a primary or secondary database is redirected to the appropriate instance network name of the underlying SQL Server instance or SQL Server Failover Cluster Instance (FCI).

SQL Server instances are actively hosted on a single node.

If present, a stand-alone SQL Server Instance always resides on a single Node with a static instance network name. If present, a SQL Server FCI is active on one of two or more possible failover nodes with a single virtual Instance Network Name.

Nodes are members of a WSFC cluster.

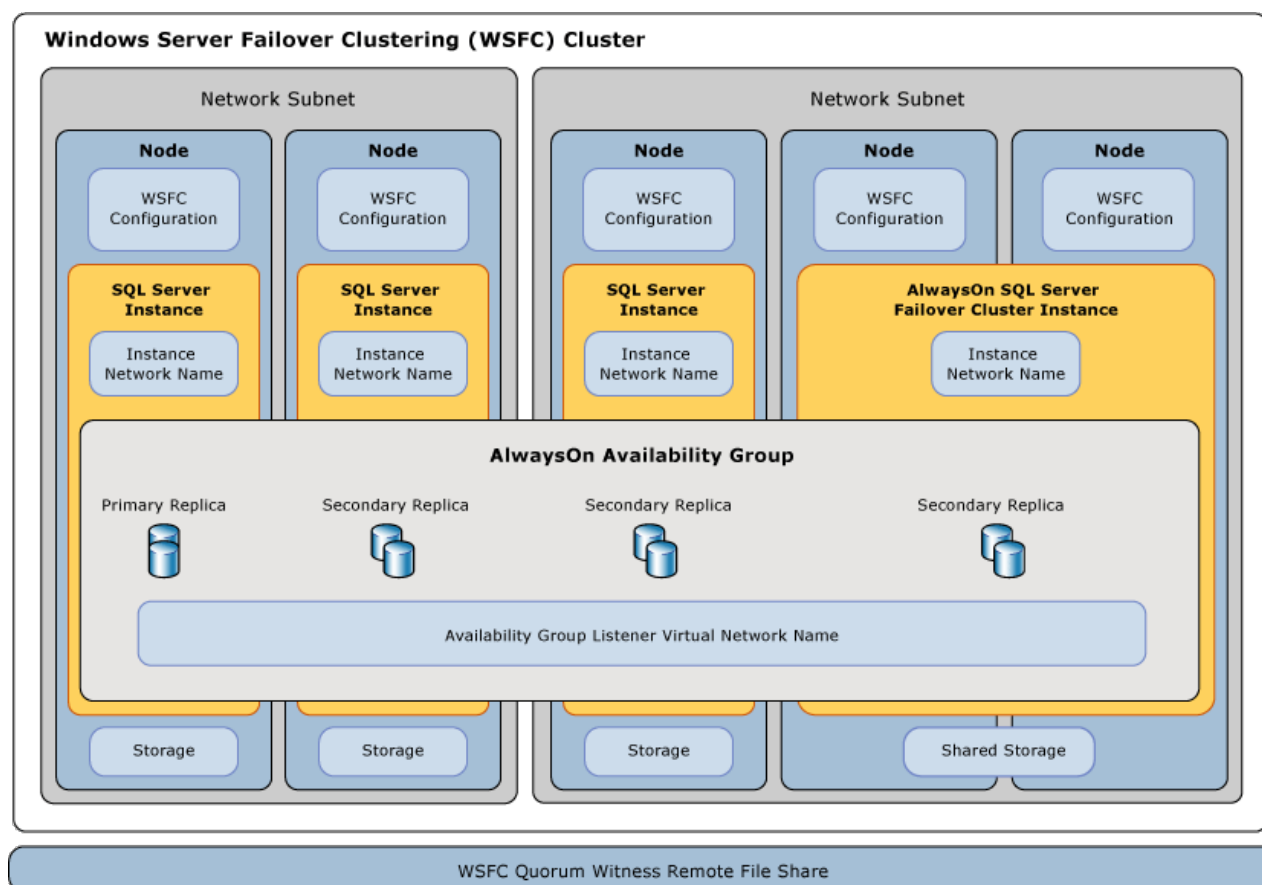
WSFC configuration metadata and status for all nodes is stored on each node. Each server may provide asymmetric storage or shared storage (SAN) volumes for user or system databases. Each server has at least one physical network interface on one or more IP subnets.

The WSFC service monitors health and manages configuration for a group of servers.

The Windows Server Failover Cluster (WSFC) service propagates changes to WSFC Configuration metadata and status to all nodes in the cluster. Partial metadata and status may be stored on a WSFC quorum-witness remote file share. Two or more active nodes or witnesses constitute a quorum to vote on the health of the WSFC cluster.

Always On availability groups registry keys are subkeys of the WSFC cluster.

If you delete and re-create a WSFC cluster, you must disable and re-enable the Always On availability groups feature on each server instance that was enabled for Always On availability groups on the original WSFC cluster. For more information, see [Enable and Disable Always On Availability Groups \(SQL Server\)](#).



Related Tasks

- [View Cluster Quorum NodeWeight Settings](#)
- [Configure Cluster Quorum NodeWeight Settings](#)
- [Force a WSFC Cluster to Start Without a Quorum](#)

Related Content

- [Windows Server Technologies: Failover Clusters](#)
- [Storage Spaces Direct \(S2D\) Overview](#)

- [Failover Clusters in Windows Server 2008 R2](#)
- [View Events and Logs for a Failover Cluster](#)
- [Get-ClusterLog Failover Cluster Cmdlet](#)

See Also

[Always On Failover Cluster Instances \(SQL Server\)](#)

[Overview of Always On Availability Groups \(SQL Server\)](#)

[WSFC Quorum Modes and Voting Configuration \(SQL Server\)](#)

[Failover Policy for Failover Cluster Instances](#)

[WSFC Disaster Recovery through Forced Quorum \(SQL Server\)](#)

[SQL Server 2016 Supports Windows Server 2016 Storage Spaces Direct](#)

WSFC Quorum Modes and Voting Configuration (SQL Server)

3/29/2017 • 7 min to read • [Edit Online](#)

Both SQL Server Always On availability groups and Always On Failover Cluster Instances (FCI) take advantage of Windows Server Failover Clustering (WSFC) as a platform technology. WSFC uses a quorum-based approach to monitoring overall cluster health and maximize node-level fault tolerance. A fundamental understanding of WSFC quorum modes and node voting configuration is very important to designing, operating, and troubleshooting your Always On high availability and disaster recovery solution.

In this topic:

- [Cluster Health Detection by Quorum](#)
- [Quorum Modes](#)
- [Voting and Non-Voting Nodes](#)
- [Recommended Adjustments to Quorum Voting](#)
- [Related Tasks](#)
- [Related Content](#)

Cluster Health Detection by Quorum

Each node in a WSFC cluster participates in periodic heartbeat communication to share the node's health status with the other nodes. Unresponsive nodes are considered to be in a failed state.

A *quorum* node set is a majority of the voting nodes and witnesses in the WSFC cluster. The overall health and status of a WSFC cluster is determined by a periodic *quorum vote*. The presence of a quorum means that the cluster is healthy and able to provide node-level fault tolerance.

The absence of a quorum indicates that the cluster is not healthy. Overall WSFC cluster health must be maintained in order to ensure that healthy secondary nodes are available for primary nodes to fail over to. If the quorum vote fails, the WSFC cluster will be set offline as a precautionary measure. This will also cause all SQL Server instances registered with the cluster to be stopped.

IMPORTANT

If a WSFC cluster is set offline because of quorum failure, manual intervention is required to bring it back online.

For more information, see: [WSFC Disaster Recovery through Forced Quorum \(SQL Server\)](#).

Quorum Modes

A *quorum mode* is configured at the WSFC cluster level that dictates the methodology used for quorum voting. The Failover Cluster Manager utility will recommend a quorum mode based on the number of nodes in the cluster.

The following quorum modes can be used to determine what constitutes a quorum of votes:

- **Node Majority.** More than one-half of the voting nodes in the cluster must vote affirmatively for the

cluster to be healthy.

- **Node and File Share Majority.** Similar to Node Majority quorum mode, except that a remote file share is also configured as a voting witness, and connectivity from any node to that share is also counted as an affirmative vote. More than one-half of the possible votes must be affirmative for the cluster to be healthy.

As a best practice, the witness file share should not reside on any node in the cluster, and it should be visible to all nodes in the cluster.

- **Node and Disk Majority.** Similar to Node Majority quorum mode, except that a shared disk cluster resource is also designated as a voting witness, and connectivity from any node to that shared disk is also counted as an affirmative vote. More than one-half of the possible votes must be affirmative for the cluster to be healthy.
- **Disk Only.** A shared disk cluster resource is designated as a witness, and connectivity by any node to that shared disk is counted as an affirmative vote.

TIP

When using an asymmetric storage configuration for Always On availability groups, you should generally use the Node Majority quorum mode when you have an odd number of voting nodes, or the Node and File Share Majority quorum mode when you have an even number of voting nodes.

Voting and Non-Voting Nodes

By default, each node in the WSFC cluster is included as a member of the cluster quorum; each node has a single vote in determining the overall cluster health, and each node will continuously attempt to establish a quorum. The quorum discussion to this point has carefully qualified the set of WSFC cluster nodes that vote on cluster health as *voting nodes*.

No individual node in a WSFC cluster can definitively determine that the cluster as a whole is healthy or unhealthy. At any given moment, from the perspective of each node, some of the other nodes may appear to be offline, or appear to be in the process of failover, or appear unresponsive due to a network communication failure. A key function of the quorum vote is to determine whether the apparent state of each of node in the WSFC cluster is indeed that actual state of those nodes.

For all of the quorum models except 'Disk Only', the effectiveness of a quorum vote depends on reliable communications between all of the voting nodes in the cluster. Network communications between nodes on the same physical subnet should be considered reliable; the quorum vote should be trusted.

However, if a node on another subnet is seen as non-responsive in a quorum vote, but it is actually online and otherwise healthy, that is most likely due to a network communications failure between subnets. Depending upon the cluster topology, quorum mode, and failover policy configuration, that network communications failure may effectively create more than one set (or subset) of voting nodes.

When more than one subset of voting nodes is able to establish a quorum on its own, that is known as a *split-brain scenario*. In such a scenario, the nodes in the separate quorums may behave differently, and in conflict with one another.

NOTE

The split-brain scenario is only possible when a system administrator manually performs a forced quorum operation, or in very rare circumstances, a forced failover; explicitly subdividing the quorum node set.

In order to simplify your quorum configuration and increase up-time, you may want to adjust each node's

NodeWeight setting so that the node's vote is not counted towards the quorum.

IMPORTANT

In order to use *NodeWeight* settings, the following hotfix must be applied to all servers in the WSFC cluster:

[KB2494036](#): A hotfix is available to let you configure a cluster node that does not have quorum votes in Windows Server 2008 and in Windows Server 2008 R2

Recommended Adjustments to Quorum Voting

When enabling or disabling a given WSFC node's vote, follow these guidelines:

- **No vote by default.** Assume that each node should not vote without explicit justification.
- **Include all primary replicas.** Each WSFC node that hosts an availability group primary replica or is the preferred owner of an FCI should have a vote.
- **Include possible automatic failover owners.** Each node that could host a primary replica, as the result of an automatic availability group failover or FCI failover, should have a vote. If there is only one availability group in the WSFC cluster and availability replicas are hosted only by standalone instances, this rule includes only the secondary replica that is the automatic failover target.
- **Exclude secondary site nodes.** In general, do not give votes to WSFC nodes that reside at a secondary disaster recovery site. You do not want nodes in the secondary site to contribute to a decision to take the cluster offline when there is nothing wrong with the primary site.
- **Odd number of votes.** If necessary, add a witness file share, a witness node, or a witness disk to the cluster and adjust the quorum mode to prevent possible ties in the quorum vote.
- **Re-assess vote assignments post-failover.** You do not want to fail over into a cluster configuration that does not support a healthy quorum.

IMPORTANT

When validating WSFC quorum vote configuration, the Always On Availability Group Wizard shows a warning if any of the following conditions are true:

- The cluster node that hosts the primary replica does not have a vote
 - A secondary replica is configured for automatic failover and its cluster node does not have a vote.
 - [KB2494036](#) is not installed on all cluster nodes that host availability replicas. This patch is required to add or remove votes for cluster nodes in multi-site deployments. However, in single-site deployments, it is usually not required and you may safely ignore the warning.

TIP

SQL Server exposes several system dynamic management views (DMVs) that can help you manage settings related WSFC cluster configuration and node quorum voting.

For more information, see: [sys.dm_hadr_cluster](#), [sys.dm_hadr_cluster_members](#), [sys.dm_os_cluster_nodes](#), [sys.dm_hadr_cluster_networks](#)

Related Tasks

- [View Cluster Quorum NodeWeight Settings](#)

- [Configure Cluster Quorum NodeWeight Settings](#)

Related Content

- [Microsoft SQL Server Always On Solutions Guide for High Availability and Disaster Recovery](#)
- [Quorum vote configuration check in Always On Availability Group Wizards](#)
- [Windows Server Technologies: Failover Clusters](#)
- [Failover Cluster Step-by-Step Guide: Configuring the Quorum in a Failover Cluster](#)

See Also

[WSFC Disaster Recovery through Forced Quorum \(SQL Server\)](#)

[Windows Server Failover Clustering \(WSFC\) with SQL Server](#)

View Cluster Quorum NodeWeight Settings

3/29/2017 • 2 min to read • [Edit Online](#)

This topic describes how to view NodeWeight settings for each member node in a Windows Server Failover Clustering (WSFC) cluster. NodeWeight settings are used during quorum voting to support disaster recovery and multi-subnet scenarios for Always On availability groups and SQL Server Failover Cluster Instances.

- **Before you start:** [Prerequisites](#), [Security](#)
- **To view quorum NodeWeight settings using:** [Using Transact-SQL](#), [Using Powershell](#), [Using Cluster.exe](#)

Before You Start

Prerequisites

This feature is supported only in Windows Server 2008 or later versions.

IMPORTANT

In order to use NodeWeight settings, the following hotfix must be applied to all servers in the WSFC cluster:

[KB2494036](#): A hotfix is available to let you configure a cluster node that does not have quorum votes in Windows Server 2008 and in Windows Server 2008 R2

TIP

If this hotfix is not installed, the examples in this topic will return empty or NULL values for NodeWeight.

Security

The user must be a domain account that is member of the local Administrators group on each node of the WSFC cluster.

Using Transact-SQL

To view NodeWeight settings

1. Connect to any SQL Server instance in the cluster.
2. Query the [sys].[dm_hadr_cluster_members] view.

Example (Transact-SQL)

The following example queries a system view to return values for all of the nodes in that instance's cluster.

```
SELECT member_name, member_state_desc, number_of_quorum_votes
FROM sys.dm_hadr_cluster_members;
```

Using Powershell

To view NodeWeight settings

1. Start an elevated Windows PowerShell via **Run as Administrator**.
2. Import the `FailoverClusters` module to enable cluster commandlets.

3. Use the `Get-ClusterNode` object to return a collection of cluster node objects.
4. Output the cluster node properties in a readable format.

Example (Powershell)

The following example output some of the node properties for the cluster called "Cluster001".

```
Import-Module FailoverClusters

$cluster = "Cluster001"
$nodes = Get-ClusterNode -Cluster $cluster

$nodes | Format-Table -property NodeName, State, NodeWeight
```

Using Cluster.exe

NOTE

The cluster.exe utility is deprecated in the Windows Server 2008 R2 release. Please use PowerShell with Failover Clustering for future development. The cluster.exe utility will be removed in the next release of Windows Server. For more information, see [Mapping Cluster.exe Commands to Windows PowerShell Cmdlets for Failover Clusters](#).

To view **NodeWeight** settings

1. Start an elevated Command Prompt via **Run as Administrator**.
2. Use **cluster.exe** to return node status and NodeWeight values

Example (Cluster.exe)

The following example outputs some of the node properties for the cluster called "Cluster001".

```
cluster.exe Cluster001 node /status /properties
```

See Also

[WSFC Quorum Modes and Voting Configuration \(SQL Server\)](#)

[Configure Cluster Quorum NodeWeight Settings](#)

[sys.dm_hadr_cluster_members \(Transact-SQL\)](#)

[Failover Cluster Cmdlets in Windows PowerShell Listed by Task Focus](#)

Configure Cluster Quorum NodeWeight Settings

3/29/2017 • 2 min to read • [Edit Online](#)

This topic describes how to configure NodeWeight settings for a member node in a Windows Server Failover Clustering (WSFC) cluster. NodeWeight settings are used during quorum voting to support disaster recovery and multi-subnet scenarios for Always On availability groups and SQL Server Failover Cluster Instances.

- **Before you start:** [Prerequisites](#), [Security](#)
- **To view quorum NodeWeight settings using:** [Using Powershell](#), [Using Cluster.exe](#)
- [Related Content](#)

Before You Start

Prerequisites

This feature is supported only in Windows Server 2008 or later versions.

IMPORTANT

In order to use NodeWeight settings, the following hotfix must be applied to all servers in the WSFC cluster:

[KB2494036](#): A hotfix is available to let you configure a cluster node that does not have quorum votes in Windows Server 2008 and in Windows Server 2008 R2

TIP

If this hotfix is not installed, the examples in this topic will return empty or NULL values for NodeWeight.

Security

The user must be a domain account that is member of the local Administrators group on each node of the WSFC cluster.

Using Powershell

To configure NodeWeight settings

1. Start an elevated Windows PowerShell via **Run as Administrator**.
2. Import the `FailoverClusters` module to enable cluster commandlets.
3. Use the `Get-ClusterNode` object to set the `NodeWeight` property for each node in the cluster.
4. Output the cluster node properties in a readable format.

Example (Powershell)

The following example changes the NodeWeight setting to remove the quorum vote for the "Always OnSrv1" node, and then outputs the settings for all nodes in the cluster.

```
Import-Module FailoverClusters

$node = "Always OnSrv1"
(Get-ClusterNode $node).NodeWeight = 0

$cluster = (Get-ClusterNode $node).Cluster
$nodes = Get-ClusterNode -Cluster $cluster

$nodes | Format-Table -property NodeName, State, NodeWeight
```

Using Cluster.exe

NOTE

The cluster.exe utility is deprecated in the Windows Server 2008 R2 release. Please use PowerShell with Failover Clustering for future development. The cluster.exe utility will be removed in the next release of Windows Server. For more information, see [Mapping Cluster.exe Commands to Windows PowerShell Cmdlets for Failover Clusters](#).

To configure NodeWeight settings

1. Start an elevated Command Prompt via **Run as Administrator**.
2. Use **cluster.exe** to set `NodeWeight` values.

Example (Cluster.exe)

The following example changes the NodeWeight value to remove the quorum vote of the "Always OnSrv1" node in the "Cluster001" cluster.

```
cluster.exe Cluster001 node Always OnSrv1 /prop NodeWeight=0
```

Related Content

- [View Events and Logs for a Failover Cluster](#)
- [Get-ClusterLog Failover Cluster Cmdlet](#)

See Also

[WSFC Quorum Modes and Voting Configuration \(SQL Server\)](#)

[View Cluster Quorum NodeWeight Settings](#)

[Failover Cluster Cmdlets in Windows PowerShell Listed by Task Focus](#)

WSFC Disaster Recovery through Forced Quorum (SQL Server)

3/29/2017 • 5 min to read • [Edit Online](#)

Quorum failure is usually caused by a systemic disaster, or a persistent communications failure, or a misconfiguration involving several nodes in the WSFC cluster. Manual intervention is required to recovery from a quorum failure.

- **Before you start:** [Prerequisites](#), [Security](#)
- **WSFC Disaster Recovery through the Forced Quorum Procedure** [WSFC Disaster Recovery through the Forced Quorum Procedure](#)
- [Related Tasks](#)
- [Related Content](#)

Before You Start

Prerequisites

The Forced Quorum Procedure assumes that a healthy quorum existed before the quorum failure.

WARNING

The user should be well-informed on the concepts and interactions of Windows Server Failover Clustering, WSFC Quorum Models, SQL Server, and the environment's specific deployment configuration.

For more information, see: [Windows Server Failover Clustering \(WSFC\) with SQL Server](#), [WSFC Quorum Modes and Voting Configuration \(SQL Server\)](#)

Security

The user must be a domain account that is member of the local Administrators group on each node of the WSFC cluster.

WSFC Disaster Recovery through the Forced Quorum Procedure

Remember that quorum failure will cause all clustered services, SQL Server instances, and Always On availability groups, in the WSFC cluster to be set offline, because the cluster, as configured, cannot ensure node-level fault tolerance. A quorum failure means that healthy voting nodes in the WSFC cluster no longer satisfy the quorum model. Some nodes may have failed completely, and some may have just shut down the WSFC service and are otherwise healthy, except for the loss of the ability to communicate with a quorum.

To bring the WSFC cluster back online, you must correct the root cause of the quorum failure under the existing configuration, recover the affected databases as needed, and you may want to reconfigure the remaining nodes in the WSFC cluster to reflect the surviving cluster topology.

You can use the *forced quorum* procedure on a WSFC cluster node to override the safety controls that took the cluster offline. This effectively tells the cluster to suspend the quorum voting checks, and lets you bring the WSFC cluster resources and SQL Server back online on any of the nodes in the cluster.

This type of disaster recovery process should include the following steps:

To Recover from Quorum Failure:

1. **Determine the scope of the failure.** Identify which availability groups or SQL Server instances are non-responsive, which cluster nodes are online and available for post-disaster use, and examine the Windows event logs and the SQL Server system logs. Where practical, you should preserve forensic data and system logs for later analysis.

TIP

On a responsive instance of SQL Server 2016, you can obtain information about the health of availability groups that possess an availability replica on the local server instance by querying the [sys.dm_hadr_availability_group_states](#) dynamic management view (DMV).

2. **Start the WSFC cluster by using forced quorum on a single node.** Identify a node with a minimal number of component failures, other than that the WSFC cluster service was shut down. Verify that this node can communicate with a majority of the other nodes.

On this node, manually force the cluster to come online using the forced quorum procedure. To minimize potential data loss, select a node that was last hosting an availability group primary replica.

For more information, see: [Force a WSFC Cluster to Start Without a Quorum](#)

NOTE

The forced quorum setting has a cluster-wide affect to block quorum checks until the logical WSFC cluster achieves a majority of votes and automatically transitions to a regular quorum mode of operation.

3. **Start the WSFC service normally on each otherwise healthy node, one at a time.** You do not have to specify the forced quorum option when you start the cluster service on the other nodes.

As the WSFC service on each node comes back online, it negotiates with the other healthy nodes to synchronize the new cluster configuration state. Remember to do this one node at a time to prevent potential race conditions in resolving the last known state of the cluster.

WARNING

Ensure that each node that you start can communicate with the other newly online nodes. Consider disabling the WSFC service on the other nodes. Otherwise, you run the risk of creating more than one quorum node set; that is a split-brain scenario. If your findings in step 1 were accurate, this should not occur.

4. **Apply new quorum mode and node vote configuration.** If forcing quorum successfully restarted all the nodes in the cluster and the root cause of the quorum failure has been corrected, changes to the original quorum mode and node vote configuration are unnecessary.

Otherwise, you should evaluate the newly recovered cluster node and availability replica topology, and change the quorum mode and vote assignments for each node as appropriate. Un-recovered nodes should be set offline or have their node votes set to zero.

TIP

At this point, the nodes and SQL Server instances in the cluster may appear to be restored back to regular operation. However, a healthy quorum may still not exist. Using the Failover Cluster Manager, or the Always On Dashboard within SQL Server Management Studio, or the appropriate DMVs, verify that a quorum has been restored.

5. **Recover availability group database replicas as needed.** Non-availability group databases should recover and come back online on their own as part of the regular SQL Server startup process.

You can minimize potential data loss and recovery time for the availability group replicas by bringing them back online in this sequence: primary replica, synchronous secondary replicas, asynchronous secondary replicas.

6. **Repair or replace failed components and re-validate cluster.** Now that you have recovered from the initial disaster and quorum failure, you should repair or replace the failed nodes and adjust related WSFC and Always On configurations accordingly. This can include dropping availability group replicas, evicting nodes from the cluster, or flattening and re-installing software on a node.

You must repair or remove all failed availability replicas. SQL Server will not truncate the transaction log past the last known point of the farthest behind availability replica. If a failed replica is not repaired or removed from the availability group, the transaction logs will grow and you will run the risk of running out of transaction log space on the other replicas.

NOTE

If you run the WSFC Validate a Configuration Wizard when an availability group listener exists on the WSFC cluster, the wizard generates the following incorrect warning message:

"The RegisterAllProviderIP property for network name 'Name:' is set to 1 For the current cluster configuration this value should be set to 0."

Please ignore this message.

7. **Repeat step 4 as needed.** The goal is to re-establish the appropriate level of fault tolerance and high availability for healthy operations.
8. **Conduct RPO/RTO analysis.** You should analyze SQL Server system logs, database timestamps, and Windows event logs to determine root cause of the failure, and to document actual recovery point and recovery time experiences.

Related Tasks

- [Force a WSFC Cluster to Start Without a Quorum](#)
- [Perform a Forced Manual Failover of an Availability Group \(SQL Server\)](#)
- [View Cluster Quorum NodeWeight Settings](#)
- [Configure Cluster Quorum NodeWeight Settings](#)
- [Use the AlwaysOn Dashboard \(SQL Server Management Studio\)](#)

Related Content

- [View Events and Logs for a Failover Cluster](#)
- [Get-ClusterLog Failover Cluster Cmdlet](#)

See Also

[Windows Server Failover Clustering \(WSFC\) with SQL Server](#)

Force a WSFC Cluster to Start Without a Quorum

3/29/2017 • 3 min to read • [Edit Online](#)

This topic describes how to force a Windows Server Failover Clustering (WSFC) cluster node to start without a quorum. This may be required in disaster recovery and multi-subnet scenarios to recover data and fully re-establish high-availability for Always On availability groups and SQL Server Failover Cluster Instances.

- **Before you start:** [Recommendations](#), [Security](#)
- **To force a cluster to start without a quorum using:** [Using Failover Cluster Manager](#), [Using Powershell](#), [Using Net.exe](#)
- **Follow up:** [Follow Up: After Forcing Cluster to Start without a Quorum](#)

Before You Start

Recommendations

Except where explicitly directed, the procedures in this topic should work if you execute them from any node in the WSFC cluster. However, you may obtain better results, and avoid networking issues, by executing these steps from the node that you intend to force to start without a quorum.

Security

The user must be a domain account that is member of the local Administrators group on each node of the WSFC cluster.

Using Failover Cluster Manager

To force a cluster to start without a quorum

1. Open a Failover Cluster Manager and connect to the desired cluster node to force online.
2. In the **Actions** pane, click **Force Cluster Start**, and then click **Yes – Force my cluster to start**.
3. In the left pane, in the **Failover Cluster Manager** tree, click the cluster name.
4. In the summary pane, confirm that the current **Quorum Configuration** value is: **Warning: Cluster is running in ForceQuorum state**.

Using Powershell

To force a cluster to start without a quorum

1. Start an elevated Windows PowerShell via **Run as Administrator**.
2. Import the `FailoverClusters` module to enable cluster commandlets.
3. Use `Stop-ClusterNode` to make sure that the cluster service is stopped.
4. Use `Start-ClusterNode` with `-FixQuorum` to force the cluster service to start.
5. Use `Get-ClusterNode` with `-Property NodeWeight = 1` to set the value that guarantees that the node is a voting member of the quorum.
6. Output the cluster node properties in a readable format.

Example (Powershell)

The following example forces the Always OnSrv02 node cluster service to start without a quorum, sets the

`NodeWeight = 1` , and then enumerates cluster node status from the newly forced node.

```
Import-Module FailoverClusters

$node = "Always OnSrv02"
Stop-ClusterNode -Name $node
Start-ClusterNode -Name $node -FixQuorum

(Get-ClusterNode $node).NodeWeight = 1

$nodes = Get-ClusterNode -Cluster $node
$nodes | Format-Table -property NodeName, State, NodeWeight
```

Using Net.exe

To force a cluster to start without a quorum

1. Use Remote Desktop to connect to the desired cluster node to force online.
2. Start an elevated Command Prompt via **Run as Administrator**.
3. Use **net.exe** to make sure that the local cluster service is stopped.
4. Use **net.exe** with `/forcequorum` to force the local cluster service to start.

Example (Net.exe)

The following example forces a node cluster service to start without a quorum, sets the `NodeWeight = 1` , and then enumerates cluster node status from the newly forced node.

```
net.exe stop clussvc
net.exe start clussvc /forcequorum
```

Follow Up: After Forcing Cluster to Start without a Quorum

- You must re-evaluate and reconfigure `NodeWeight` values to correctly construct a new quorum before bringing other nodes back online. Otherwise, the cluster may go back offline again.

For more information, see [WSFC Quorum Modes and Voting Configuration \(SQL Server\)](#).

- The procedures in this topic are only one step in bringing the WSFC cluster back online if an un-planned quorum failure were to occur. You may also want to take additional steps to prevent other WSFC cluster nodes from interfering with the new quorum configuration.
- Other SQL Server features such as Always On availability groups, database mirroring, and log shipping may also require subsequent actions to recover data and to fully re-establish high-availability.

For more information:

[Perform a Forced Manual Failover of an Availability Group \(SQL Server\)](#)

[Force Service in a Database Mirroring Session \(Transact-SQL\)](#)

[Fail Over to a Log Shipping Secondary \(SQL Server\)](#)

Related Content

- [View Events and Logs for a Failover Cluster](#)
- [Get-ClusterLog Failover Cluster Cmdlet](#)

See Also

[WSFC Disaster Recovery through Forced Quorum \(SQL Server\)](#)

[Configure Cluster Quorum NodeWeight Settings](#)

[Failover Cluster Cmdlets in Windows PowerShell Listed by Task Focus](#)

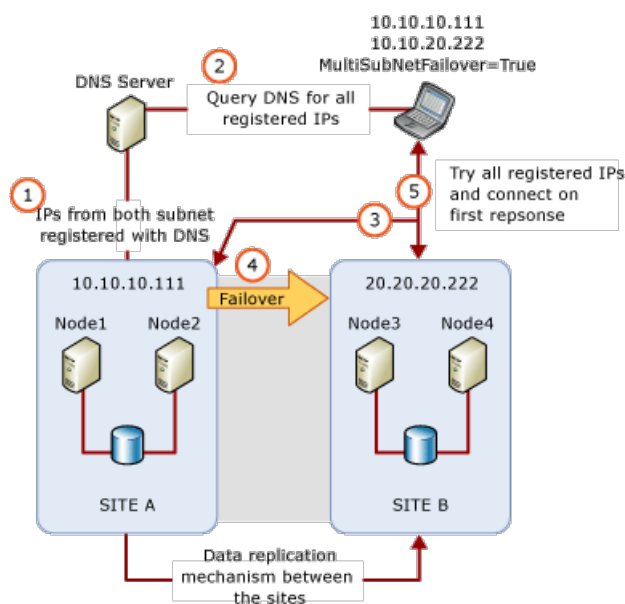
SQL Server Multi-Subnet Clustering (SQL Server)

3/29/2017 • 5 min to read • [Edit Online](#)

A SQL Server multi-subnet failover cluster is a configuration where each failover cluster node is connected to a different subnet or different set of subnets. These subnets can be in the same location or in geographically dispersed sites. Clustering across geographically dispersed sites is sometimes referred to as stretch clusters. As there is no shared storage that all the nodes can access, data should be replicated between the data storage on the multiple subnets. With data replication, there is more than one copy of the data available. Therefore, a multi-subnet failover cluster provides a disaster recovery solution in addition to high availability.

SQL Server Multi-Subnet Failover Cluster (Two-Nodes, Two-Subnets)

The following illustration represents a two node, two subnet failover cluster instance (FCI) in SQL Server 2016.



Multi-Subnet Failover Cluster Instance Configurations

The following are some examples of SQL Server FCIs that use multiple subnets:

- SQL Server FCI SQLCLUST1 includes Node1 and Node2. Node1 is connected to Subnet1. Node2 is connected to Subnet2. SQL Server Setup sees this configuration as a multi-subnet cluster and sets the IP address resource dependency to **OR**.
- SQL Server FCI SQLCLUST1 includes Node1, Node2, and Node3. Node1 and Node2 are connected to Subnet1. Node 3 is connected to Subnet2. SQL Server Setup sees this configuration as a multi-subnet cluster and sets the IP address resource dependency to **OR**. Because Node1 and Node2 are on the same subnet, this configuration provides additional local high availability.
- SQL Server FCI SQLCLUST1 includes Node1 and Node2. Node1 is on Subnet1. Node2 is on Subnet1 and Subnet2. SQL Server Setup sees this configuration as a multi-subnet cluster and sets the IP address resource dependency to **OR**.
- SQL Server FCI SQLCLUST1 includes Node1 and Node2. Node1 is connected to Subnet1 and Subnet2. Node2 is also connected to Subnet1 and Subnet2. The IP address resource dependency is set to **AND** by SQL Server Setup.

NOTE: This configuration is not considered as a multi-subnet failover cluster configuration because the clustered nodes are on the same set of subnets.

IP Address Resource Considerations

In a multi-subnet failover cluster configuration, the IP addresses are not owned by all the nodes in the failover cluster, and may not be all online during SQL Server startup. Beginning in SQL Server 2012, you can set the IP address resource dependency to **OR**. This enables SQL Server to be online when there is at least one valid IP address that it can bind to.

NOTE: In the SQL Server versions earlier than SQL Server 2012, a stretch V-LAN technology was used in multi-site cluster configurations to expose a single IP address for failover across sites. With the new capability of SQL Server to cluster nodes across different subnets, you can now configure SQL Server failover clusters across multiple sites without implementing the stretch V-LAN technology.

IP Address Resource OR Dependency Considerations

You may want to consider the following failover behavior if you set the IP address resource dependency is set to **OR**:

- When there is a failure of one of the IP addresses on the node that currently owns the SQL Server cluster resource group, a failover is not triggered automatically until all the IP addresses valid on that node fail.
- When a failover occurs, SQL Server will come online if it can bind to at least one IP address that is valid on the current node. The IP addresses that did not bind to SQL Server at startup will be listed in the error log.

When a SQL Server FCI is installed side-by-side with a standalone instance of the SQL Server Database Engine, take care to avoid TCP port number conflicts on the IP addresses. Conflicts usually occur when two instances of the Database Engine are both configured to use the default TCP port (1433). To avoid conflicts, configure one instance to use a non-default fixed port. Configuring a fixed port is usually easiest on the standalone instance. Configuring the Database Engine to use different ports will prevent an unexpected IP Address/TCP port conflict that blocks an instance startup when a SQL Server FCI fails to the standby node.

Client Recovery Latency During Failover

A multi-subnet FCI by default enables the RegisterAllProvidersIP cluster resource for its network name. In a multi-subnet configuration, both the online and offline IP addresses of the network name will be registered at the DNS server. The client application then retrieves all registered IP addresses from the DNS server and attempts to connect to the addresses either in order or in parallel. This means that client recovery time in multi-subnet failovers no longer depend on DNS update latencies. By default, the client tries the IP addresses in order. When the client uses the new optional **MultiSubnetFailover=True** parameter in its connection string, it will instead try the IP addresses simultaneously and connects to the first server that responds. This can help minimize the client recovery latency when failovers occur. For more information, see [Always On Client Connectivity \(SQL Server\)](#) and [Create or Configure an Availability Group Listener \(SQL Server\)](#).

With legacy client libraries or third party data providers, you cannot use the **MultiSubnetFailover** parameter in your connection string. To help ensure that your client application works optimally with multi-subnet FCI in SQL Server 2016, try to adjust the connection timeout in the client connection string by 21 seconds for each additional IP address. This ensures that the client's reconnection attempt does not timeout before it is able to cycle through all IP addresses in your multi-subnet FCI.

The default client connection time-out period for SQL Server Management Studio and **sqlcmd** is 15 seconds.

Related Content

CONTENT DESCRIPTION	TOPIC
Installing a SQL Server Failover Cluster	Create a New SQL Server Failover Cluster (Setup)
In-place upgrade of your existing SQL Server Failover Cluster	Upgrade a SQL Server Failover Cluster Instance (Setup)
Maintaining your existing SQL Server Failover Cluster	Add or Remove Nodes in a SQL Server Failover Cluster (Setup)
Use the Failover Cluster Management snap-in to view WSFC events and logs	View Events and Logs for a Failover Cluster
Use Windows PowerShell to create a log file for all nodes (or a specific a node) in a WSFC failover cluster	Get-ClusterLog Failover Cluster Cmdlet

Failover Cluster Troubleshooting

3/29/2017 • 9 min to read • [Edit Online](#)

This topic provides information about the following issues:

- Basic troubleshooting steps.
- Recovering from a failover cluster failure.
- Resolving the most common failover clustering problems.
- Using extended stored procedures and COM objects.

Basic Troubleshooting Steps

The first diagnostic step is to run a fresh cluster validation check. For details on validation, see [Failover Cluster Step-by-Step Guide: Validating Hardware for a Failover Cluster](#). This can be completed without any interruption of service as it does not affect any online cluster resources. Validation can be run at any time once the Failover Clustering feature has been installed, including before the cluster has been deployed, during cluster creation and while the cluster is running. In fact, additional tests are executed once the cluster is in use, which check that best practices are being followed for highly-available workloads. Across these dozens of tests, only a few of them will impact running cluster workloads and these are all within the storage category, so skipping this entire category is an easy way to avoid disruptive tests.

Failover Clustering comes with a built-in safeguard to prevent accidental downtime when running the storage tests during validation. If the cluster has any online groups when validation is initiated, and the storage tests remain selected, it will prompt the user for confirmation whether they want to run all the tests (and cause downtime), or to skip testing the disks of any online groups to avoid downtime. If the entire storage category was excluded from being tested, then this prompt is not displayed. This will enable cluster validation with no downtime.

How to revalidate your cluster

1. In the Failover Cluster snap-in, in the console tree, make sure **Failover Cluster Management** is selected and then, under **Management**, click **Validate a Configuration**.
2. Follow the instructions in the wizard to specify the servers and the tests, and run the tests. The **Summary** page appears after the tests run.
3. While still on the **Summary** page, click **View Report** to view the test results.

To view the results of the tests after you close the wizard, see

%SystemRoot%\Cluster\Reports\Validation Report date and time.html where **%SystemRoot%** is the folder in which the operating system is installed (for example, **C:\Windows**).

4. To view help topics that will help you interpret the results, click **More about cluster validation tests**.

To view help topics about cluster validation after you close the wizard, in the Failover Cluster snap-in, click **Help**, click **Help Topics**, click the **Contents** tab, expand the contents for the failover cluster help, and click **Validating a Failover Cluster Configuration**. After the validation wizard has completed, the **Summary Report** will display the results. All tests must pass with either a green check mark or in some cases a yellow triangle (warning). When looking for problem areas (red Xs or yellow question marks), in the part of the report that summarizes the test results, click an individual test to review the details. Any red X issues will need to be resolved prior to troubleshooting SQL Server issues.

Install Updates

Installing updates is an important part of avoiding problems with your system. Useful links:

- [Recommended hotfixes and updates for Windows Server 2012 R2-based failover clusters](#)
- [Recommended hotfixes and updates for Windows Server 2012-based failover clusters](#)
- [Recommended hotfixes and updates for Windows Server 2008 R2-based failover clusters](#)
- [Recommended hotfixes and updates for Windows Server 2008-based failover clusters](#)

Recovering from Failover Cluster Failure

Usually, failover cluster failure is to the result of one of two causes:

- Hardware failure in one node of a two-node cluster. This hardware failure could be caused by a failure in the SCSI card or in the operating system.

To recover from this failure, remove the failed node from the failover cluster using the SQL Server Setup program, address the hardware failure with the computer offline, bring the machine back up, and then add the repaired node back to the failover cluster instance.

For more information, see [Create a New SQL Server Failover Cluster \(Setup\)](#) and [Recover from Failover Cluster Instance Failure](#).

- Operating system failure. In this case, the node is offline, but is not irretrievably broken.

To recover from an operating system failure, recover the node and test failover. If the SQL Server instance does not fail over properly, you must use the SQL Server Setup program to remove SQL Server from the failover cluster, make necessary repairs, bring the computer back up, and then add the repaired node back to the failover cluster instance.

Recovering from operating system failure this way can take time. If the operating system failure can be recovered easily, avoid using this technique.

For more information, see [Create a New SQL Server Failover Cluster \(Setup\)](#) and [How to: Recover from Failover Cluster Failure in Scenario 2](#).

Resolving Common Problems

The following list describes common usage issues and explains how to resolve them.

Problem: Incorrect use of command-prompt syntax to install SQL Server

Issue 1: It is difficult to diagnose Setup issues when using the `/qn` switch from the command prompt, as the `/qn` switch suppresses all Setup dialog boxes and error messages. If the `/qn` switch is specified, all Setup messages, including error messages, are written to Setup log files. For more information about log files, see [View and Read SQL Server Setup Log Files](#).

Resolution 1: Use the `/qb` switch instead of the `/qn` switch. If you use the `/qb` switch, the basic UI in each step will be displayed, including error messages.

Problem: SQL Server cannot log on to the network after it migrates to another node

Issue 1: SQL Server service accounts are unable to contact a domain controller.

Resolution 1: Check your event logs for signs of networking issues such as adapter failures or DNS problems. Verify that you can ping your domain controller.

Issue 2: SQL Server service account passwords are not identical on all cluster nodes, or the node does not restart a SQL Server service that has migrated from a failed node.

Resolution 2: Change the SQL Server service account passwords using SQL Server Configuration Manager. If you do not, and you change the SQL Server service account passwords on one node, you must also change the passwords on all other nodes. SQL Server Configuration Manager does this automatically.

Problem: SQL Server cannot access the cluster disks

Issue 1: Firmware or drivers are not updated on all nodes.

Resolution 1: Verify that all nodes are using correct firmware versions and same driver versions.

Issue 2: A node cannot recover cluster disks that have migrated from a failed node on a shared cluster disk with a different drive letter.

Resolution 2: Disk drive letters for the cluster disks must be the same on both servers. If they are not, review your original installation of the operating system and Microsoft Cluster Service (MSCS).

Problem: Failure of a SQL Server service causes failover

Resolution: To prevent the failure of specific services from causing the SQL Server group to fail over, configure those services using Cluster Administrator in Windows, as follows:

- Clear the **Affect the Group** check box on the **Advanced** tab of the **Full Text Properties** dialog box. However, if SQL Server causes a failover, the full-text search service restarts.

Problem: SQL Server does not start automatically

Resolution: Use Cluster Administrator in MSCS to automatically start a failover cluster. The SQL Server service should be set to start manually; the Cluster Administrator should be configured in MSCS to start the SQL Server service. For more information, see [Managing Services](#).

Problem: The Network Name is offline and you cannot connect to SQL Server using TCP/IP

Issue 1: DNS is failing with cluster resource set to require DNS.

Resolution 1: Correct the DNS problems.

Issue 2: A duplicate name is on the network.

Resolution 2: Use NBTSTAT to find the duplicate name and then correct the issue.

Issue 3: SQL Server is not connecting using Named Pipes.

Resolution 3: To connect using Named Pipes, create an alias using the SQL Server Configuration Manager to connect to the appropriate computer. For example, if you have a cluster with two nodes (**Node A** and **Node B**), and a failover cluster instance (**Virtsql**) with a default instance, you can connect to the server that has the Network Name resource offline using the following steps:

1. Determine on which node the group containing the instance of SQL Server is running by using the Cluster Administrator. For this example, it is **Node A**.
2. Start the SQL Server service on that computer using **net start**. For more information about using **net start**, see [Starting SQL Server Manually](#).
3. Start the SQL Server SQL Server Configuration Manager on **Node A**. View the pipe name on which the server is listening. It should be similar to \\.\\$\VIRTSQL\pipe\sql\query.
4. On the client computer, start the SQL Server Configuration Manager.
5. Create an alias SQLTEST1 to connect through Named Pipes to this pipe name. To do this, enter **Node A** as the server name and edit the pipe name to be \\.\pipe\\$\VIRTSQL\sql\query.
6. Connect to this instance using the alias SQLTEST1 as the server name.

Problem: SQL Server Setup fails on a cluster with error 11001

Issue: An orphan registry key in [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\MSSQL.X\Cluster]

Resolution: Make sure the MSSQL.X registry hive is not currently in use, and then delete the cluster key.

Problem: Cluster Setup Error: "The installer has insufficient privileges to access this directory: <drive>\Microsoft SQL Server. The installation cannot continue. Log on as an administrator or contact your system administrator"

Issue: This error is caused by a SCSI shared drive that is not partitioned properly.

Resolution: Re-create a single partition on the shared disk using the following steps:

1. Delete the disk resource from the cluster.
2. Delete all partitions on the disk.
3. Verify in the disk properties that the disk is a basic disk.
4. Create one partition on the shared disk, format the disk, and assign a drive letter to the disk.
5. Add the disk to the cluster using Cluster Administrator (cluadmin).
6. Run SQL Server Setup.

Problem: Applications fail to enlist SQL Server resources in a distributed transaction

Issue: Because the Microsoft Distributed Transaction Coordinator (MS DTC) is not completely configured in Windows, applications may fail to enlist SQL Server resources in a distributed transaction. This problem can affect linked servers, distributed queries, and remote stored procedures that use distributed transactions. For more information about how to configure MS DTC, see [Before Installing Failover Clustering](#).

Resolution: To prevent such problems, you must fully enable MS DTC services on the servers where SQL Server is installed and MS DTC is configured.

To fully enable MS DTC, use the following steps:

1. In Control Panel, open **Administrative Tools**, and then open **Computer Management**.
2. In the left pane of Computer Management, expand **Services and Applications**, and then click **Services**.
3. In the right pane of Computer Management, right-click **Distributed Transaction Coordinator**, and select **Properties**.
4. In the **Distributed Transaction Coordinator** window, click the **General** tab, and then click **Stop** to stop the service.
5. In the **Distributed Transaction Coordinator** window, click the **Logon** tab, and set the logon account NT AUTHORITY\NetworkService.
6. Click **Apply** and **OK** to close the **Distributed Transaction Coordinator** window. Close the **Computer Management** window. Close the **Administrative Tools** window.

Using Extended Stored Procedures and COM Objects

When you use extended stored procedures with a failover clustering configuration, all extended stored procedures must be installed on a SQL Server-dependent cluster disk. Doing so ensures that when a node fails over, the extended stored procedures can still be used.

If the extended stored procedures use COM components, the administrator must register the COM components on each node of the cluster. The information for loading and executing COM components must be in the registry of the active node in order for the components to be created. Otherwise, the information remains in the registry of the

computer on which the COM components were first registered.

See Also





[View and Read SQL Server Setup Log Files](#)

[How Extended Stored Procedures Work](#)

[Execution Characteristics of Extended Stored Procedures](#)

Always On Failover Cluster Instances (SQL Server)

3/29/2017 • 9 min to read • [Edit Online](#)

THIS TOPIC APPLIES TO:  SQL Server (starting with 2016)  Azure SQL Database  Azure SQL Data Warehouse  Parallel Data Warehouse

As part of the SQL Server Always On offering, Always On Failover Cluster Instances leverages Windows Server Failover Clustering (WSFC) functionality to provide local high availability through redundancy at the server-instance level—a *failover cluster instance* (FCI). An FCI is a single instance of SQL Server that is installed across Windows Server Failover Clustering (WSFC) nodes and, possibly, across multiple subnets. On the network, an FCI appears to be an instance of SQL Server running on a single computer, but the FCI provides failover from one WSFC node to another if the current node becomes unavailable.

An FCI can leverage [Availability Groups](#) to provide remote disaster recovery at the database level. For more information, see [Failover Clustering and Availability Groups \(SQL Server\)](#).

NOTE

Windows Server 2016 Datacenter edition introduces support for Storage Spaces Direct (S2D). SQL Server Failover Cluster Instances support S2D for cluster storage resources. For more information, see [Storage Spaces Direct in Windows Server 2016](#).

Failover Cluster Instances also support Clustered Shared Volumes (CSV). For more information, see [Understanding Cluster Shared Volumes in a Failover Cluster](#).

In this Topic:

- [Benefits](#)
- [Recommendations](#)
- [Failover Cluster Instance Overview](#)
- [Elements of a Failover Cluster Instance](#)
- [SQL Server Failover Concepts and Tasks](#)
- [Related Topics](#)

Benefits of a Failover Cluster Instance

When there is hardware or software failure of a server, the applications or clients connecting to the server will experience downtime. When a SQL Server instance is configured to be an FCI (instead of a standalone instance), the high availability of that SQL Server instance is protected by the presence of redundant nodes in the FCI. Only one of the nodes in the FCI owns the WSFC resource group at a time. In case of a failure (hardware failures, operating system failures, application or service failures), or a planned upgrade, the resource group ownership is moved to another WSFC node. This process is transparent to the client or application connecting to SQL Server and this minimize the downtime the application or clients experience during a failure. The following lists some key benefits that SQL Server failover cluster instances provide:

- Protection at the instance level through redundancy
- Automatic failover in the event of a failure (hardware failures, operating system failures, application or service failures)

IMPORTANT

In an availability group, automatic failover from an FCI to other nodes within the availability group is not supported. This means that FCIs and standalone nodes should not be coupled together within an availability group if automatic failover is an important component your high availability solution. However, this coupling can be made for your *disaster recovery* solution.

- Support for a broad array of storage solutions, including WSFC cluster disks (iSCSI, Fiber Channel, and so on) and server message block (SMB) file shares.
- Disaster recovery solution using a multi-subnet FCI or running an FCI-hosted database inside an availability group. With the new multi-subnet support in Microsoft SQL Server 2012, a multi-subnet FCI no longer requires a virtual LAN, increasing the manageability and security of a multi-subnet FCI.
- Zero reconfiguration of applications and clients during failovers
- Flexible failover policy for granular trigger events for automatic failovers
- Reliable failovers through periodic and detailed health detection using dedicated and persisted connections
- Configurability and predictability in failover time through indirect background checkpoints
- Throttled resource usage during failovers

Recommendations

In a production environment, we recommend that you use static IP addresses in conjunction the virtual IP address of a Failover Cluster Instance. We recommend against using DHCP in a production environment. In the event of down time, if the DHCP IP lease expires, extra time is required to re-register the new DHCP IP address associated with the DNS name.

Failover Cluster Instance Overview

An FCI runs in a WSFC resource group with one or more WSFC nodes. When the FCI starts up, one of the nodes assume ownership of the resource group and brings its SQL Server instance online. The resources owned by this node include:

- Network name
- IP address
- Shared disks
- SQL Server Database Engine service
- SQL Server Agent service
- SQL Server Analysis Services service, if installed
- One file share resource, if the FILESTREAM feature is installed

At any time, only the resource group owner (and no other node in the FCI) is running its respective SQL Server services in the resource group. When a failover occurs, whether it be an automatic failover or a planned failover, the following sequence of events happen:

1. Unless a hardware or system failure occurs, all dirty pages in the buffer cache are written to disk.
2. All respective SQL Server services in the resource group are stopped on the active node.

3. The resource group ownership is transferred to another node in the FCI.
4. The new resource group owner starts its SQL Server services.
5. Client application connection requests are automatically directed to the new active node using the same virtual network name (VNN).

The FCI is online as long as its underlying WSFC cluster is in good quorum health (the majority of the quorum WSFC nodes are available as automatic failover targets). When the WSFC cluster loses its quorum, whether due to hardware, software, network failure, or improper quorum configuration, the entire WSFC cluster, along with the FCI, is brought offline. Manual intervention is then required in this unplanned failover scenario to reestablish quorum in the remaining available nodes in order to bring the WSFC cluster and FCI back online. For more information, see [WSFC Quorum Modes and Voting Configuration \(SQL Server\)](#).

Predictable Failover Time

Depending on when your SQL Server instance last performed a checkpoint operation, there can be a substantial amount of dirty pages in the buffer cache. Consequently, failovers last as long as it takes to write the remaining dirty pages to disk, which can lead to long and unpredictable failover time. Beginning with Microsoft SQL Server 2012, the FCI can use indirect checkpoints to throttle the amount of dirty pages kept in the buffer cache. While this does consume additional resources under regular workload, it makes the failover time more predictable as well as more configurable. This is very useful when the service-level agreement in your organization specifies the recovery time objective (RTO) for your high availability solution. For more information on indirect checkpoints, see [Indirect Checkpoints](#).

Reliable Health Monitoring and Flexible Failover Policy

After the FCI starts successfully, the WSFC service monitors both the health of the underlying WSFC cluster, as well as the health of the SQL Server instance. Beginning with Microsoft SQL Server 2012, the WSFC service uses a dedicated connection to poll the active SQL Server instance for detailed component diagnostics through a system stored procedure. The implication of this is three-fold:

- The dedicated connection to the SQL Server instance makes it possible to reliably poll for component diagnostics all the time, even when the FCI is under heavy load. This makes it possible to distinguish between a system that is under heavy load and a system that actually has failure conditions, thus preventing issues such as false failovers.
- The detailed component diagnostics makes it possible to configure a more flexible failover policy, whereby you can choose what failure conditions trigger failovers and which failure conditions do not.
- The detailed component diagnostics also enables better troubleshooting of automatic failovers retroactively. The diagnostic information is stored to log files, which are collocated with the SQL Server error logs. You can load them into the Log File Viewer to inspect the component states leading up to the failover occurrence in order to determine what cause that failover.

For more information, see [Failover Policy for Failover Cluster Instances](#)

Elements of a Failover Cluster Instance

An FCI consists of a set of physical servers (nodes) that contain similar hardware configuration as well as identical software configuration that includes operating system version and patch level, and SQL Server version, patch level, components, and instance name. Identical software configuration is necessary to ensure that the FCI can be fully functional as it fails over between the nodes.

WSFC Resource Group

A SQL Server FCI runs in a WSFC resource group. Each node in the resource group maintains a synchronized copy of the configuration settings and check-pointed registry keys to ensure full functionality of the FCI after a failover,

and only one of the nodes in the cluster owns the resource group at a time (the active node). The WSFC service manages the server cluster, quorum configuration, failover policy, and failover operations, as well as the VNN and virtual IP addresses for the FCI. In case of a failure (hardware failures, operating system failures, application or service failures) or a planned upgrade, the resource group ownership is moved to another node in the FCI. The number of nodes that are supported in a WSFC resource group depends on your SQL Server edition. Also, the same WSFC cluster can run multiple FCIs (multiple resource groups), depending on your hardware capacity, such as CPUs, memory, and number of disks.

SQL Server Binaries

The product binaries are installed locally on each node of the FCI, a process similar to SQL Server stand-alone installations. However, during startup, the services are not started automatically, but managed by WSFC.

Storage

Contrary to the availability group, an FCI must use shared storage between all nodes of the FCI for database and log storage. The shared storage can be in the form of WSFC cluster disks, disks on a SAN, Storage Spaces Direct (S2D), or file shares on an SMB. This way, all nodes in the FCI have the same view of instance data whenever a failover occurs. This does mean, however, that the shared storage has the potential of being the single point of failure, and FCI depends on the underlying storage solution to ensure data protection.

Network Name

The VNN for the FCI provides a unified connection point for the FCI. This allows applications to connect to the VNN without the need to know the currently active node. When a failover occurs, the VNN is registered to the new active node after it starts. This process is transparent to the client or application connecting to SQL Server and this minimize the downtime the application or clients experience during a failure.

Virtual IPs

In the case of a multi-subnet FCI, a virtual IP address is assigned to each subnet in the FCI. During a failover, the VNN on the DNS server is updated to point to the virtual IP address for the respective subnet. Applications and clients can then connect to the FCI using the same VNN after a multi-subnet failover.

SQL Server Failover Concepts and Tasks

CONCEPTS AND TASKS	TOPIC
Describes the failure detection mechanism and the flexible failover policy.	Failover Policy for Failover Cluster Instances
Describes concepts in FCI administration and maintenance.	Failover Cluster Instance Administration and Maintenance
Describes multi-subnet configuration and concepts	SQL Server Multi-Subnet Clustering (SQL Server)

Related Topics

TOPIC DESCRIPTIONS	TOPIC
Describes how to install a new SQL Server FCI.	Create a New SQL Server Failover Cluster (Setup)
Describes how to upgrade to a SQL Server 2016 failover cluster.	Upgrade a SQL Server Failover Cluster Instance
Describes Windows Failover Clustering Concepts and provides links to tasks related to Windows Failover Clustering	Windows Server 2008: Overview of Failover Clusters
	Windows Server 2008 R2: Overview of Failover Clusters

TOPIC DESCRIPTIONS	TOPIC
Describes the distinctions in concepts between nodes in an FCI and replicas within an availability group and considerations for using an FCI to host a replica for an availability group.	Failover Clustering and Availability Groups (SQL Server)

Failover Policy for Failover Cluster Instances

3/29/2017 • 6 min to read • [Edit Online](#)

In a SQL Server failover cluster instance (FCI), only one node can own the Windows Server Failover Cluster (WSFC) cluster resource group at a given time. The client requests are served through this node in the FCI. In the case of a failure and an unsuccessful restart, the group ownership is moved to another WSFC node in the FCI. This process is called failover. SQL Server 2016 increases the reliability of failure detection and provides a flexible failover policy.

A SQL Server FCI depends on the underlying WSFC service for failover detection. Therefore, two mechanisms determine the failover behavior for FCI: the former is native WSFC functionality, and the latter is functionality added by SQL Server setup.

- The WSFC cluster maintains the quorum configuration, which ensures a unique failover target in an automatic failover. The WSFC service determines whether the cluster is in optimal quorum health at all times and brings the resource group online and offline accordingly.
- The active SQL Server instance periodically reports a set of component diagnostics to the WSFC resource group over a dedicated connection. The WSFC resource group maintains the failover policy, which defines the failure conditions that trigger restarts and failovers.

This topic discusses the second mechanism above. For more information on the WSFC behavior for quorum configuration and health detection, see [WSFC Quorum Modes and Voting Configuration \(SQL Server\)](#).

IMPORTANT

Automatic failovers to and from an FCI are not allowed in an Always On availability group. However, manual failovers to and from an FCI are allowed in an Always On availability group.

Failover Policy Overview

The failover process can be broken down into the following steps:

1. [Monitor the Health Status](#)
2. [Determining Failures](#)
3. [Responding to Failures](#)

Monitor the Health Status

There are three types of health statuses that are monitored for the FCI:

- [State of the SQL Server service](#)
- [Responsiveness of the SQL Server instance](#)
- [SQL Server component diagnostics](#)

State of the SQL Server service

The WSFC service monitors the start state of the SQL Server service on the active FCI node to detect when the SQL Server service is stopped.

Responsiveness of the SQL Server instance

During SQL Server startup, the WSFC service uses the SQL Server Database Engine resource DLL to create a new connection to on a separate thread that is used exclusively for monitoring the health status. This ensures that there the SQL instance has the required resources to report its health status while under load. Using this dedicated connection, SQL Server runs the [sp_server_diagnostics \(Transact-SQL\)](#) system stored procedure in repeat mode to periodically report the health status of the SQL Server components to the resource DLL.

The resource DLL determines the responsiveness of the SQL instance using a health check timeout. The HealthCheckTimeout property defines how long the resource DLL should wait for the sp_server_diagnostics stored procedure before it reports the SQL instance as unresponsive to the WSFC service. This property is configurable using T-SQL as well as in the Failover Cluster Manager snap-in. For more information, see [Configure HealthCheckTimeout Property Settings](#). The following items describe how this property affects timeout and repeat interval settings:

- The resource DLL calls the sp_server_diagnostics stored procedure and sets the repeat interval to one-third of the HealthCheckTimeout setting.
- If the sp_server_diagnostics stored procedure is slow or is not returning information, the resource DLL will wait for the interval specified by HealthCheckTimeout before it reports to the WSFC service that the SQL instance is unresponsive.
- If the dedicated connection is lost, the resource DLL will retry the connection to the SQL instance for the interval specified by HealthCheckTimeout before it reports to the WSFC service that the SQL instance is unresponsive.

SQL Server component diagnostics

The system stored procedure sp_server_diagnostics periodically collects component diagnostics on the SQL instance. The diagnostic information that is collected is surfaced as a row for each of the following components and passed to the calling thread.

1. system
2. resource
3. query process
4. io_subsystem
5. events

The system, resource, and query process components are used for failure detection. The io_subsystem and events components are used for diagnostic purposes only.

Each rowset of information is also written to the SQL Server cluster diagnostics log. For more information, see [View and Read Failover Cluster Instance Diagnostics Log](#).

TIP

While the sp_server_diagnostic stored procedure is used by SQL Server Always On technology, it is available for use in any SQL Server instance to help detect and troubleshoot problems.

Determining Failures

The SQL Server Database Engine resource DLL determines whether the detected health status is a condition for failure using the FailureConditionLevel property. The FailureConditionLevel property defines which detected health statuses cause restarts or failovers. Multiple levels of options are available, ranging from no automatic restart or failover to all possible failure conditions resulting in an automatic restart or failover. For more information about how to configure this property, see [Configure FailureConditionLevel Property Settings](#).

The failure conditions are set on an increasing scale. For levels 1-5, each level includes all the conditions from the

previous levels in addition to its own conditions. This means that with each level, there is an increased probability of a failover or restart. The failure condition levels are described in the following table.

Review [sp_server_diagnostics \(Transact-SQL\)](#) as this system stored procedure plays an important role in the failure condition levels.

LEVEL	CONDITION	DESCRIPTION
0	No automatic failover or restart	Indicates that no failover or restart will be triggered automatically on any failure conditions. This level is for system maintenance purposes only.
1	Failover or restart on server down	Indicates that a server restart or failover will be triggered if the following condition is raised: SQL Server service is down.
2	Failover or restart on server unresponsive	Indicates that a server restart or failover will be triggered if any of the following conditions are raised: SQL Server service is down. SQL Server instance is not responsive (Resource DLL cannot receive data from sp_server_diagnostics within the HealthCheckTimeout settings).
3*	Failover or restart on critical server errors	Indicates that a server restart or failover will be triggered if any of the following conditions are raised: SQL Server service is down. SQL Server instance is not responsive (Resource DLL cannot receive data from sp_server_diagnostics within the HealthCheckTimeout settings). System stored procedure sp_server_diagnostics returns 'system error'.

LEVEL	CONDITION	DESCRIPTION
4	Failover or restart on moderate server errors	<p>Indicates that a server restart or failover will be triggered if any of the following conditions are raised:</p> <p>SQL Server service is down.</p> <p>SQL Server instance is not responsive (Resource DLL cannot receive data from sp_server_diagnostics within the HealthCheckTimeout settings).</p> <p>System stored procedure sp_server_diagnostics returns 'system error'.</p> <p>System stored procedure sp_server_diagnostics returns 'resource error'.</p>
5	Failover or restart on any qualified failure conditions	<p>Indicates that a server restart or failover will be triggered if any of the following conditions are raised:</p> <p>SQL Server service is down.</p> <p>SQL Server instance is not responsive (Resource DLL cannot receive data from sp_server_diagnostics within the HealthCheckTimeout settings).</p> <p>System stored procedure sp_server_diagnostics returns 'system error'.</p> <p>System stored procedure sp_server_diagnostics returns 'resource error'.</p> <p>System stored procedure sp_server_diagnostics returns 'query_processing error'.</p>

*Default Value

Responding to Failures

After one or more failure conditions are detected, how the WSFC service responds to the failures depends on the WSFC quorum state and the restart and failover settings of the FCI resource group. If the FCI has lost its WSFC quorum, then the entire FCI is brought offline and the FCI has lost its high availability. If the FCI still retains its WSFC quorum, then the WSFC service may respond by first attempting to restart the failed node and then failover if the restart attempts are unsuccessful. The restart and failover settings are configured in the Failover Cluster Manager snap-in. For more information these settings, see [<Resource> Properties: Policies Tab](#).

For more information on maintaining quorum health, see [WSFC Quorum Modes and Voting Configuration \(SQL Server\)](#).

See Also

[ALTER SERVER CONFIGURATION \(Transact-SQL\)](#)

Configure HealthCheckTimeout Property Settings

3/29/2017 • 1 min to read • [Edit Online](#)

The HealthCheckTimeout setting is used to specify the length of time, in milliseconds, that the SQL Server resource DLL should wait for information returned by the [sp_server_diagnostics](#) stored procedure before reporting the Always On Failover Cluster Instance (FCI) as unresponsive. Changes that are made to the timeout settings are effective immediately and do not require a restart of the SQL Server resource.

- **Before you begin:** [Limitations and Restrictions](#), [Security](#)
- **To Configure the HealthCheckTimeout setting, using:** [PowerShell](#), [Failover Cluster Manager](#), [Transact-SQL](#)

Before You Begin

Limitations and Restrictions

The default value for this property is 30,000 milliseconds (30 seconds). The minimum value is 15,000 milliseconds (15 seconds).

Security

Permissions

Requires ALTER SETTINGS and VIEW SERVER STATE permissions.

Using PowerShell

To configure HealthCheckTimeout settings

1. Start an elevated Windows PowerShell via **Run as Administrator**.
2. Import the **FailoverClusters** module to enable cluster cmdlets.
3. Use the **Get-ClusterResource** cmdlet to find the SQL Server resource, then use **Set-ClusterParameter** cmdlet to set the **HealthCheckTimeout** property for the failover cluster instance.

TIP

Every time you open a new PowerShell window, you need to import the **FailoverClusters** module.

Example (PowerShell)

The following example changes the HealthCheckTimeout setting on the SQL Server resource "SQL Server (INST1)" to 60000 milliseconds.

```
Import-Module FailoverClusters

$fci = "SQL Server (INST1)"
Get-ClusterResource $fci | Set-ClusterParameter HealthCheckTimeout 60000
```

Related Content (PowerShell)

- [Clustering and High-Availability](#) (Failover Clustering and Network Load Balancing Team Blog)
- [Getting Started with Windows PowerShell on a Failover Cluster](#)

- [Cluster resource commands and equivalent Windows PowerShell cmdlets](#)

Using the Failover Cluster Manager Snap-in

To configure HealthCheckTimeout setting

1. Open the Failover Cluster Manager snap-in.
2. Expand **Services and Applications** and select the FCI.
3. Right-click the **SQL Server resource** under **Other Resources** and select **Properties** from the right-click menu. The SQL Server resource **Properties** dialog box opens.
4. Select the **Properties** tab, enter the desired value for the **HealthCheckTimeout** property, and then click **OK** to apply the change.

Using Transact-SQL

Using the [ALTER SERVER CONFIGURATION](#) Transact-SQL statement, you can specify the HealthCheckTimeOut property value.

Example (Transact-SQL)

The following example sets the HealthCheckTimeout option to 15,000 milliseconds (15 seconds).

```
ALTER SERVER CONFIGURATION  
SET FAILOVER CLUSTER PROPERTY HealthCheckTimeout = 15000;
```

See Also

[Failover Policy for Failover Cluster Instances](#)

Configure FailureConditionLevel Property Settings

3/29/2017 • 2 min to read • [Edit Online](#)

Use the FailureConditionLevel property to set the conditions for the Always On Failover Cluster Instance (FCI) to fail over or restart. Changes to this property are applied immediately without requiring a restart of the Windows Server Failover Cluster (WSFC) service or the FCI resource.

- **Before you begin:** [FailureConditionLevel Property Settings, Security](#)
- **To configure the FailureConditionLevel property settings using,** [PowerShell](#), [Failover Cluster Manager](#), [Transact-SQL](#)

Before You Begin

FailureConditionLevel Property Settings

The failure conditions are set on an increasing scale. For levels 1-5, each level includes all the conditions from the previous levels in addition to its own conditions. This means that with each level, there is an increased probability of a failover or restart. For more information, see the "Determining Failures" section of the [Failover Policy for Failover Cluster Instances](#) topic.

Security

Permissions

Requires ALTER SETTINGS and VIEW SERVER STATE permissions.

Using PowerShell

To configure FailureConditionLevel settings

1. Start an elevated Windows PowerShell via **Run as Administrator**.
2. Import the **FailoverClusters** module to enable cluster cmdlets.
3. Use the **Get-ClusterResource** cmdlet to find the SQL Server resource, then use **Set-ClusterParameter** cmdlet to set the **FailureConditionLevel** property for a Failover Cluster Instance.

TIP

Every time you open a new PowerShell window, you need to import the **FailoverClusters** module.

Example (PowerShell)

The following example changes the FailureConditionLevel setting on the SQL Server resource "SQL Server (INST1)" to fail over or restart on critical server errors.

```
Import-Module FailoverClusters

$fci = "SQL Server (INST1)"
Get-ClusterResource $fci | Set-ClusterParameter FailureConditionLevel 3
```

Related Content (PowerShell)

- [Clustering and High-Availability](#) (Failover Clustering and Network Load Balancing Team Blog)
- [Getting Started with Windows PowerShell on a Failover Cluster](#)

- [Cluster resource commands and equivalent Windows PowerShell cmdlets](#)

Using the Failover Cluster Manager Snap-in

To configure FailureConditionLevel property settings:

1. Open the Failover Cluster Manager snap-in.
2. Expand the **Services and Applications** and select the FCI.
3. Right-click the **SQL Server resource** under **Other Resources**, and then select **Properties** from the menu.
The SQL Server resource **Properties** dialog box opens.
4. Select the **Properties** tab, enter the desired value for the **FailureConditionLevel** property, and then click **OK** to apply the change.

Using Transact-SQL

To configure FailureConditionLevel property settings:

Using the [ALTER SERVER CONFIGURATION](#) Transact-SQL statement, you can specify the FailureConditionLevel property value.

Example (Transact-SQL)

The following example sets the FailureConditionLevel property to 0, indicating that failover or restart will not be triggered automatically on any failure conditions.

```
ALTER SERVER CONFIGURATION SET FAILOVER CLUSTER PROPERTY FailureConditionLevel = 0;
```

See Also

[sp_server_diagnostics](#) (Transact-SQL)

[Failover Policy for Failover Cluster Instances](#)

View and Read Failover Cluster Instance Diagnostics Log

3/29/2017 • 3 min to read • [Edit Online](#)

All critical errors and warning events for the SQL Server Resource DLL are written to the Windows event log. A running log of the diagnostic information specific to SQL Server is captured by the [sp_server_diagnostics](#) (Transact-SQL) system stored procedure and is written to the SQL Server failover cluster diagnostics (also known as the *SQLDIAG* logs) log files.

- **Before you begin:** [Recommendations](#), [Security](#)
- **To View the Diagnostic Log, using:** [SQL Server Management Studio](#), [Transact-SQL](#)
- **To Configure Diagnostic Log settings, using:** [Transact-SQL](#)

Before You Begin

Recommendations

By default, the SQLDIAG are stored under a local LOG folder of the SQL Server instance directory, for example, 'C:\Program Files\Microsoft SQL Server\MSSQL13.<InstanceName>\MSSQL\LOG' of the owning node of the Always On Failover Cluster Instance (FCI). The size of each SQLDIAG log file is fixed at 100 MB. Ten such log files are stored on the computer before they are recycled for new logs.

The logs use the extended events file format. The **sys.fn_xe_file_target_read_file** system function can be used to read the files that are created by Extended Events. One event, in XML format, is returned per row. Query the system view to parse the XML data as a result-set. For more information, see [sys.fn_xe_file_target_read_file](#) (Transact-SQL).

Security

Permissions

VIEW SERVER STATE permission is needed to run **fn_xe_file_target_read_file**.

Open SQL Server Management Studio as Administrator

Using SQL Server Management Studio

To view the Diagnostic log files:

1. From the **File** menu, select **Open, File**, and choose the diagnostic log file you want to view.
2. The events are displayed as rows in the right pane, and by default **name**, and **timestamp** are the only two columns displayed.

This also activates the **ExtendedEvents** menu.

3. To see more columns, go the **ExtendedEvents** menu, and select **Choose Columns**.

A dialog box opens with the available columns allowing you to select the columns for display.

4. You can filter, and sort the event data using the **ExtendedEvents** menu and selecting the **Filter** option.

Using Transact-SQL

To view the Diagnostic log files:

To view all the log items in the SQLDIAG log file, use the following query:

```
SELECT
xml_data.value('(event/@name)[1]','varchar(max)') AS 'Name'
,xml_data.value('(event/@package)[1]','varchar(max)') AS 'Package'
,xml_data.value('(event/@timestamp)[1]','datetime') AS 'Time'
,xml_data.value('(event/data[@name='state']/value)[1]','int') AS 'State'
,xml_data.value('(event/data[@name='state_desc']/text)[1]','varchar(max)') AS 'State Description'
,xml_data.value('(event/data[@name='failure_condition_level']/value)[1]','int') AS 'Failure Conditions'
,xml_data.value('(event/data[@name='node_name']/value)[1]','varchar(max)') AS 'Node_Name'
,xml_data.value('(event/data[@name='instancename']/value)[1]','varchar(max)') AS 'Instance Name'
,xml_data.value('(event/data[@name='creation time']/value)[1]','datetime') AS 'Creation Time'
,xml_data.value('(event/data[@name='component']/value)[1]','varchar(max)') AS 'Component'
,xml_data.value('(event/data[@name='data']/value)[1]','varchar(max)') AS 'Data'
,xml_data.value('(event/data[@name='info']/value)[1]','varchar(max)') AS 'Info'
FROM
( SELECT object_name AS 'event'
, CONVERT(xml,event_data) AS 'xml_data'
FROM sys.fn_xe_file_target_read_file('C:\Program Files\Microsoft SQL
Server\MSSQL13.MSSQLSERVER\MSSQL\Log\SQLNODE1_MSSQLSERVER_SQLDIAG_0_129936003752530000.xel',NULL,NULL,NULL)
)
AS XEventData
ORDER BY Time;
```

NOTE

You can filter the results for specific components or state using the WHERE clause.

Using Transact-SQL

To configure the Diagnostic Log Properties

NOTE

For an example of this procedure, see [Example \(Transact-SQL\)](#), later in this section.

Using the Data Definition Language (DDL) statement, **ALTER SERVER CONFIGURATION**, you can start or stop logging diagnostic data captured by the [sp_server_diagnostics \(Transact-SQL\)](#) procedure, and set SQLDIAG log configuration parameters such as the log file rollover count, log file size, and file location. For syntax details, see [Setting diagnostic log options](#).

Examples (Transact-SQL)

Setting diagnostic log options

The examples in this section show how to set the values for the diagnostic log option.

A. Starting diagnostic logging

The following example starts the logging of diagnostic data.

```
ALTER SERVER CONFIGURATION SET DIAGNOSTICS LOG ON;
```

B. Stopping diagnostic logging

The following example stops the logging of diagnostic data.

```
ALTER SERVER CONFIGURATION SET DIAGNOSTICS LOG OFF;
```

C. Specifying the location of the diagnostic logs

The following example sets the location of the diagnostic logs to the specified file path.

```
ALTER SERVER CONFIGURATION  
SET DIAGNOSTICS LOG PATH = 'C:\logs';
```

D. Specifying the maximum size of each diagnostic log

The following example set the maximum size of each diagnostic log to 10 megabytes.

```
ALTER SERVER CONFIGURATION  
SET DIAGNOSTICS LOG MAX_SIZE = 10 MB;
```

See Also

[Failover Policy for Failover Cluster Instances](#)

Failover Cluster Instance Administration and Maintenance

3/29/2017 • 2 min to read • [Edit Online](#)

Maintenance tasks like adding or removing nodes from an existing Always On Failover Cluster Instance (FCI) are accomplished using the SQL Server Setup program. Other administration tasks like changing the IP address resource, recovering from certain FCI scenarios are accomplished using the Failover Cluster Manager snap-in, which is the management snap-in for the Windows Server Failover Clustering (WSFC) service.

Maintaining a Failover Cluster Instance

After you have installed an FCI, you can change or repair it using the SQL Server Setup program. For example, you can add additional nodes to an FCI, run an FCI as a stand-alone instance, or remove a node from a FCI configuration.

Adding a Node to an Existing Failover Cluster Instance

SQL Server Setup gives you the option of maintaining an existing FCI. If you choose this option, you can add other nodes to your FCI by running SQL Server Setup on the computer that you want to add to the FCI. For more information, see [Create a New SQL Server Failover Cluster \(Setup\)](#) and [Add or Remove Nodes in a SQL Server Failover Cluster \(Setup\)](#).

Removing a Node from an Existing Failover Cluster Instance

You can remove a node from an FCI by running SQL Server Setup on the computer that you want to remove from the FCI. Each node in an FCI is considered a peer without dependencies on other nodes on the FCI, and you can remove any node. A damaged node does not have to be available to be removed, and the removal process does not uninstall the SQL Server binaries from the unavailable node. A removed node can be added back to a FCI at any time. For more information, see [Add or Remove Nodes in a SQL Server Failover Cluster \(Setup\)](#).

Changing Service Accounts

You should not change passwords for any of the SQL Server service accounts when an FCI node is down or offline. If you must do this, you must reset the password again by using SQL Server Configuration Manager when all nodes are back online.

If the service account for SQL Server is not an administrator in your cluster, the administrative shares cannot be deleted on any nodes of the cluster. The administrative shares must be available in a cluster for SQL Server to function.

IMPORTANT

Do not use the same account for the SQL Server service account and the WSFC service account. If the password changes for the WSFC service account, your SQL Server installation will fail.

On Windows Server 2008, service SIDs are used for SQL Server service accounts. For more information, see [Configure Windows Service Accounts and Permissions](#).

Administering a Failover Cluster Instance

TASK DESCRIPTION	TOPIC LINK
Describes how to add dependencies to a SQL Server resource.	Add Dependencies to a SQL Server Resource
Kerberos is a network authentication protocol designed to provide strong authentication for client/server applications. Kerberos provides a foundation for interoperability and helps to enhance the security of enterprise-wide network authentication. You can use Kerberos authentication with SQL Server stand-alone instances or with Always On FCIs.	Register a Service Principal Name for Kerberos Connections.
Provides links to content that describes how to enable Kerberos authentication	
Describes the procedure used to recover from a SQL Server failover cluster failure.	Recover from Failover Cluster Instance Failure
Describe the procedure used to change the IP address resource for a SQL Server failover cluster instance.	Change the IP Address of a Failover Cluster Instance

See Also

[Configure HealthCheckTimeout Property Settings](#)

[Configure FailureConditionLevel Property Settings](#)

[View and Read Failover Cluster Instance Diagnostics Log](#)

Add Dependencies to a SQL Server Resource

3/29/2017 • 3 min to read • [Edit Online](#)

This topic describes how to add dependencies to an Always On Failover Cluster Instance (FCI) resource by using the Failover Cluster Manager snap-in. The Failover Cluster Manager snap-in is the cluster management application for the Windows Server Failover Clustering (WSFC) service.

- **Before you begin:** [Limitations and Restrictions](#), [Prerequisites](#)
- **To add a dependency to a SQL Server resource, using:** [Windows Failover Cluster Manager](#)

Before You Begin

Limitations and Restrictions

It is important to note that if you add any other resources to the SQL Server group, those resources must always have their own unique SQL network name resources and their own SQL IP address resources.

Do not use the existing SQL network name resources and SQL IP address resources for anything other than SQL Server. If SQL Server resources are shared with other resources, the following problems may occur:

- Outages that are not expected may occur.
- Service pack installations may not be successful.
- The SQL Server Setup program may not be successful. If this problem occurs, you cannot install additional instances of SQL Server or perform routine maintenance.

Consider these additional issues:

- **FTP with SQL Server replication:** For instances of SQL Server that use FTP with SQL Server replication, your FTP service must use one of the same physical disks as the installation of SQL Server that is set up to use the FTP service.
- **SQL Server resource dependencies:** If you add a resource to a SQL Server group and you have a dependency on the SQL Server resource to make sure that SQL Server is available, Microsoft recommends that you add a dependency on the SQL Server Agent resource. Do not add a dependency on the SQL Server resource. To make sure that the computer that is running SQL Server remains highly available, configure the SQL Server Agent resource so that it does not affect the SQL Server group if the SQL Server Agent resource fails.
- **File shares and printer resources:** When you install File Share resources or Printer cluster resources, they should not be put on the same physical disk resources as the computer that is running SQL Server. If they are put on the same physical disk resources, you may experience performance degradation and loss of service to the computer that is running SQL Server.
- **MS DTC considerations:** After you install the operating system and configure your FCI, you must configure Microsoft Distributed Transaction Coordinator (MS DTC) to work in a cluster by using the Failover Cluster Manager snap-in. Failure to cluster MS DTC will not block SQL Server Setup, but SQL Server application functionality may be affected if MS DTC is not properly configured.

If you install MS DTC in your SQL Server group and you have other resources that are dependent on MS DTC, MS DTC will not be available if this group is offline or during a failover. Microsoft recommends that you put MS DTC in its own group with its own physical disk resource, if it is possible.

Prerequisites

If you install SQL Server into a WSFC resource group with multiple disk drives and choose to place your data on one of the drives, the SQL Server resource will be set to be dependent only on that drive. To put data or logs on another disk, you must first add a dependency to the SQL Server resource for the additional disk.

Using the Failover Cluster Manager Snap-in

To add a dependency to a SQL Server resource

- Open the Failover Cluster Manager snap-in.
- Locate the group that contains the applicable SQL Server resource that you would like to make dependent.
- If the resource for the disk is already in this group, go to step 4. Otherwise, locate the group that contains the disk. If that group and the group that contains SQL Server are not owned by the same node, move the group containing the resource for the disk to the node that owns the SQL Server group.
- Select the SQL Server resource, open the **Properties** dialog box, and use the **Dependencies** tab to add the disk to the set of SQL Server dependencies.

Recover from Failover Cluster Instance Failure

3/29/2017 • 1 min to read • [Edit Online](#)

This topic describes how to recover from cluster failures by using the Failover Cluster Manager snap-in after a failover occurs in SQL Server 2016. The Failover Cluster Manager snap-in is the cluster management application for the Windows Server Failover Clustering (WSFC) service.

- [Recover from an irreparable failure](#)
- [Recover from a software failure](#)

Recover from an irreparable failure

Use the following steps to recover from an irreparable failure. The failure could be caused, for example, by the failure of a disk controller or the operating system. In this case, failure is caused by hardware failure in Node 1 of a two-node cluster.

1. After Node 1 fails, the SQL Server FCI fails over to Node 2.
2. Evict Node 1 from the FCI. To do this, from Node 2, open the Failover Cluster Manager snap-in, right-click Node1, click **Move Actions**, and then click **Evict Node**.
3. Verify that Node 1 has been evicted from the cluster definition.
4. Install new hardware to replace the failed hardware in Node 1.
5. Using the Failover Cluster Manager snap-in, add Node 1 to the existing cluster. For more information, see [Before Installing Failover Clustering](#).
6. Ensure that the administrator accounts are the same on all cluster nodes.
7. Run SQL Server Setup to add Node 1 to the FCI. For more information, see [Add or Remove Nodes in a SQL Server Failover Cluster \(Setup\)](#).

Recover from a reparable failure

Use the following steps to recover from a reparable failure. In this case, failure is caused by Node 1 being down or offline but not irretrievably broken. This could be caused by an operating system failure, hardware failure, or failure in the SQL Server instance itself.

1. After Node 1 fails, the FCI fails over to Node 2.
2. Resolve the problem with Node 1.
3. Ensure that all nodes are online and the WSFC service is working.
4. Fail over SQL Server to the recovered node.

Change the IP Address of a Failover Cluster Instance

3/29/2017 • 1 min to read • [Edit Online](#)

This topic describes how to change the IP address resource in an Always On Failover Cluster Instance (FCI) by using the Failover Cluster Manager snap-in. The Failover Cluster Manager snap-in is the cluster management application for the Windows Server Failover Clustering (WSFC) service.

- **Before you begin:** [Security](#)

Before You Begin

Before you begin, review the following SQL Server Books Online topic: [Before Installing Failover Clustering](#).

Security

Permissions

To maintain or update an FCI, you must be a local administrator with permission to logon as a service on all nodes of the FCI.

Using the Failover Cluster Manager Snap-in

To change the IP address resource for an FCI

1. Open the Failover Cluster Manager snap-in.
2. Expand the **Services and applications** node, in the left pane and click on the FCI.
3. On the right pane, under the **Server Name** category, right-click the SQL Server Instance, and select **Properties** option to open the **Properties** dialog box.
4. On the **General** tab, change the IP address resource.
5. Click **OK** to close the dialog box.
6. In the right-hand pane, right-click the SQL IP Address1(failover cluster instance name) and select **Take Offline**. You will see the SQL IP Address1(failover cluster instance name), SQL Network Name(failover cluster instance name), and SQL Server status change from Online to Offline Pending, and then to Offline.
7. In the right-hand pane, right-click SQL Server, and then select **Bring Online**. You will see the SQL IP Address1(failover cluster instance name), SQL Network Name(failover cluster instance name), and SQL Server status change from Offline to Online Pending, and then to Online.
8. Close the Failover Cluster Manager snap-in.

Upgrade a SQL Server Failover Cluster Instance

3/29/2017 • 3 min to read • [Edit Online](#)

SQL Server supports upgrading a SQL Server failover cluster to a new version of SQL Server, to a new SQL Serverservice pack or cumulative update, or when installing to a new Windows service pack or cumulative update separately on all failover cluster nodes with downtime limited to a single manual failover (or two manual failovers if failing back to the original primary).

Upgrading the Windows operating system of a failover cluster is not supported for operating systems before Windows Server 2012 R2. To upgrade a cluster node running on Windows Server 2012 R2, see [Cluster Operating System Rolling Upgrade](#)

Support details are as follows:

- SQL Serverupgrade is supported both through the user interface and from the command prompt. You can run upgrade from the command prompt on each failover cluster node, or by using the SQL Server setup UI to upgrade each cluster node. For more information, see [Upgrade a SQL Server Failover Cluster Instance \(Setup\)](#) and [Install SQL Server 2016 from the Command Prompt](#).
- The following scenarios are not supported as part of a SQL Server upgrade:
 - You cannot upgrade from a stand-alone instance of SQL Server to a failover cluster.
 - You cannot add features to a failover cluster. For example, you cannot add the Database Engine to an existing Analysis Services-only failover cluster.
 - You cannot downgrade a failover cluster node to a stand-alone instance.
 - Changing the edition of the failover cluster is limited to certain scenarios. For more information, see [Supported Version and Edition Upgrades](#).
- During the failover cluster upgrade, downtime is limited to failover time and the time that is required for upgrade scripts to run. If you follow the failover cluster rolling upgrade process below and meet all prerequisites on all nodes before you begin the upgrade process, your downtime is minimal. Upgrading SQL Server 2014 when memory-optimized tables are in use will take some extra time. For more information, see [Plan and Test the Database Engine Upgrade Plan](#).

Prerequisites

Before you begin, review the following important information:

- [Supported Version and Edition Upgrades](#): Verify that you can upgrade to SQL Server 2016 from your version of the Windows operating system and version of SQL Server. For example, you cannot upgrade directly from a SQL Server 2005 failover clustering instance to SQL Server 2016 or upgrade a failover cluster running on Windows Server 2003.
- [Choose a Database Engine Upgrade Method](#): Select the appropriate upgrade method and steps based on your review of supported version and edition upgrades and also based on other components installed in your environment to upgrade components in the correct order.
- [Plan and Test the Database Engine Upgrade Plan](#): Review the release notes and known upgrade issues, the pre-upgrade checklist, and develop and test the upgrade plan.
- [Hardware and Software Requirements for Installing SQL Server 2016](#): Review the software requirements

for installing SQL Server 2016. If additional software is required, install it on each node before you begin the upgrade process to minimize any downtime.

Performing a Rolling Upgrade or Update

To upgrade a SQL Server failover cluster to SQL Server 2016, use SQL Server setup to upgrade each failover cluster node, one at a time, starting with the passive nodes. As you upgrade each node, it is left out of the possible owners of the failover cluster. If there is an unexpected failover, the upgraded nodes do not participate in the failover until cluster resource group ownership is moved to an upgraded node by SQL Server setup.

By default, SQL Server setup automatically determines when to fail over to an upgraded node. This depends on the total number of nodes in the failover cluster instance and the number of nodes that have already been upgraded. When half of the nodes or more have already been upgraded, SQL Server setup causes a failover to an upgraded node when you perform upgrade on the next node. Upon failover to an upgraded node, the cluster group is moved to an upgraded node. All the upgraded nodes are put in the possible owners list and all the nodes that are not yet upgraded are removed from the possible owners list. As you upgrade each remaining node, it is added to the possible owners of the failover cluster.

This process results in downtime limited to one failover time and database upgrade script execution time during the whole failover cluster upgrade.

To control the failover behavior of cluster nodes during the upgrade process, run the upgrade operation at the command prompt and use the `/FAILOVERCLUSTERROLLOWNERSHIP` parameter. For more information, see [Install SQL Server 2016 from the Command Prompt](#).

See Also

[Upgrade to SQL Server 2016 Using the Installation Wizard \(Setup\)](#)

[Install SQL Server 2016 from the Command Prompt](#)

[Upgrade a SQL Server Failover Cluster Instance \(Setup\)](#)

Upgrade a SQL Server Failover Cluster Instance (Setup)

3/29/2017 • 6 min to read • [Edit Online](#)

You can upgrade a SQL Server failover cluster to a SQL Server 2016 failover cluster by using the SQL Server setup UI or from a command prompt.

For local installations, you must run SQL Server Setup as an administrator. If you install SQL Server from a remote share, you must use a domain account that has read permissions on the remote share.

Before upgrading, see [Upgrade a SQL Server Failover Cluster Instance](#).

To Upgrade a SQL Server Failover Cluster

To upgrade a SQL Server failover cluster

1. From the SQL Server installation media for the edition that matches the edition you are upgrading, double-click setup.exe in the root folder. You may be asked to install the prerequisites, if they are not previously installed.
2. After prerequisites are installed, the Installation Wizard starts the SQL Server Installation Center. To upgrade an existing instance of SQL Server, select your instance.
3. If SQL Server setup support files are required, SQL Server setup installs them. If you are instructed to restart your computer, restart before you continue.
4. The System Configuration Checker runs a discovery operation on your computer. To continue, Click **OK**.
5. On the Product Key page, enter the PID key for the new version edition that matches the edition of the old product version. For example, to upgrade an Enterprise failover cluster, you must supply a PID key for SQL Server Enterprise. Click **Next** to continue. Be aware that the PID key that you use for a failover cluster upgrade must be consistent across all failover cluster nodes in the same SQL Server instance.
6. On the License Terms page, read the license agreement, and then select the check box to accept the license terms and conditions. To help improve SQL Server, you can also enable the feature usage option and send reports to Microsoft. **Click Next to continue**. To end Setup, click **Cancel**.
7. On the Select Instance page, specify the SQL Server instance to upgrade to SQL Server 2016. **Click Next to continue**.
8. On the Feature Selection page, the features to upgrade are preselected. A description for each component group appears in the right pane after you select the feature name. Be aware that you cannot change the features to be upgraded, and you cannot add features during the upgrade operation. To add features to an upgraded instance of SQL Server 2014 after the upgrade operation is complete, see [Add Features to an Instance of SQL Server 2016 \(Setup\)](#).

The prerequisites for the selected features are displayed on the right-hand pane. SQL Server Setup will install the prerequisite that are not already installed during the installation step described later in this procedure. To save time, you should pre-install these prerequisites on each node.

9. On the Instance Configuration page, fields are automatically populated from the old instance. You can choose to specify the new InstanceID value.

Instance ID - By default, the instance name is used as the Instance ID. This is used to identify installation

directories and registry keys for your instance of SQL Server. This is the case for default instances and named instances. For a default instance, the instance name and instance ID would be MSSQLSERVER. To use a nondefault instance ID, select the **Instance ID** check box and provide a value. If you override the default value, you must specify the same Instance ID for the instance being upgraded on all the failover cluster nodes. The Instance ID for the upgraded instance must match across the nodes.

Detected instances and features - The grid shows instances of SQL Server that are on the computer where setup is running. **Click Next to continue.**

10. The Disk Space Requirements page calculates the required disk space for the features that you specify, and compares requirements to the available disk space on the computer where Setup is running.
11. On the Full-Text Search Upgrade page, specify the upgrade options for the databases being upgraded. For more information, see [Full-Text Search Upgrade Options](#).
12. On the **Error Reporting** page, specify the information that you want to send to Microsoft that will help improve SQL Server. By default, options for error reporting is enabled.
13. The System Configuration Checker runs one more set of rules to validate your computer configuration with the SQL Server features that you have specified, before the upgrade operation begins.
14. The Cluster Upgrade Report page displays the list of nodes in the failover cluster instance and the instance version information for SQL Server components on each node. It displays the database script status and replication script status. In addition, it also displays informational messages on what will occur when you click **Next**. Depending on the number of failover cluster nodes that have already been upgraded and total number of nodes, setup displays the failover behavior that happens when you click **Next**. It also warns about potential unnecessary downtime if you have not installed the prerequisites already.
15. The Ready to Upgrade page displays a tree view of installation options that were specified during Setup. To continue, click **Upgrade**. SQL Server Setup will first install the required prerequisites for the selected features followed by the feature installation.
16. During upgrade, the Progress page provides status so that you can monitor the upgrade progress on the current node as Setup continues.
17. After the upgrade of the current node, the Cluster Upgrade Report page displays an upgrade status information for all the failover cluster nodes, features on each failover cluster node, and their version information. Confirm the version information that is displayed and continue with the upgrade of the remaining nodes. If the failover to upgraded nodes occurred, this is also apparent on the status page. You can also check in the Windows Cluster administrator tool to confirm.
18. After upgrade, the Complete page provides a link to the summary log file for the installation and other important notes. To complete the SQL Server installation process, click **Close**.
19. If you are instructed to restart the computer, do so now. It is important to read the message from the Installation Wizard when you have finished with Setup. For more information about Setup log files, see [View and Read SQL Server Setup Log Files](#).
20. To complete the upgrade process, repeat these steps on all the other nodes on the SQL Server failover cluster.

To upgrade a SQL Server Multi-Subnet Failover Cluster

To upgrade to a SQL Server multi-subnet failover Cluster (Existing SQL Server cluster is a non multi-subnet cluster).

1. Follow the steps above to upgrade your cluster to SQL Server 2016.
2. Add a node on a different subnet using the AddNode Setup action and confirm the IP address resource dependency to OR on the **Cluster Network Configuration** page. For more information, see [Add or](#)

[Remove Nodes in a SQL Server Failover Cluster \(Setup\).](#)

To upgrade a multi-subnet cluster currently using Stretch V-Lan.

1. Follow the steps above to upgrade your cluster to SQL Server 2016.
2. Change the network settings to move the remote node to a different subnet.
3. Using the Windows Failover Cluster management tool, add a new IP address for the new subnet set the IP address resource dependency to OR.

Next Steps

After you upgrade to SQL Server 2016, complete the following tasks:

- [Complete the Database Engine Upgrade](#)
- [Change the Database Compatibility Mode and Use the Query Store](#)
- [Take Advantage of New SQL Server 2016 Features](#)

See Also

[Upgrade a SQL Server Failover Cluster Instance](#)

[View and Read SQL Server Setup Log Files](#)

[Add Features to an Instance of SQL Server 2016 \(Setup\)](#)