High Availability Solutions

SQL Server 2012 Books Online

Reference

Microsoft

High Availability Solutions

SQL Server 2012 Books Online

Summary: This book introduces two SQL Server high-availability solutions that improve the availability of servers or databases: AlwaysOn Failover Cluster Instances and AlwaysOn Availability Groups. 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.

Category: Reference
Applies to: SQL Server 2012
Source: SQL Server Books Online (link to source content)
E-book publication date: June 2012

Contents

High Availability Solutions

This topic introduces 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
For information about which editions of SQL Server support a given high availability solution, see the "High Availability (AlwaysOn)" section of Features Supported by the Editions of SQL Server 2012.

In this Topic:

- Overview of SQL Server High-Availability Solutions
- Recommended Solutions for Using SQL Server to Protect Data

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:

AlwaysOn Failover Cluster Instances

As part of the SQL Server AlwaysOn offering, AlwaysOn 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 SQL Server Failover Cluster Instances (AlwaysOn Failover Clustering).

AlwaysOn Availability Groups

AlwaysOn 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. AlwaysOn Availability Groups requires that the SQL Server instances reside on Windows Server Failover Clustering (WSFC) nodes. For more information, see AlwaysOn Availability Groups (SQL Server).

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

Recommended Solutions for Using SQL Server to Protect Data

Our recommendation for providing data protection for your SQL Server environment are as follows:

- For data protection through a third-party shared disk solution (a SAN), we recommend that you use AlwaysOn Failover Cluster Instances.
- For data protection through SQL Server, we recommend that you use AlwaysOn Availability Groups.

  Note
  If you are running an edition of SQL Server that does not support AlwaysOn Availability Groups, we recommend log shipping. For information about which editions of SQL Server support AlwaysOn Availability Groups, see the "High Availability (AlwaysOn)" section of Features Supported by the Editions of SQL Server 2012.

See Also

Availability Enhancements (Database Engine)

Windows Server Failover Clustering (WSFC) with SQL Server

High Availability: Interoperability and Coexistence

Deprecated Database Engine Features in SQL Server 2012

Windows Server Failover Clustering (WSFC) with SQL Server

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 2012 takes advantage of WSFC services and capabilities to support AlwaysOn Availability Groups and SQL Server Failover Cluster Instances.

In this topic:

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 Clusters in Windows Server 2008 R2

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.

For more information, see: Failover Cluster Glossary

SQL Server AlwaysOn Technologies and WSFC

SQL Server 2012 *AlwaysOn* is a new high availability and disaster recovery solution that takes advantage of WSFC. AlwaysOn provides an integrated, flexible solution that increases application availability, provides better returns on hardware investments, and simplifies high availability deployment and management.

Both AlwaysOn Availability Groups and AlwaysOn 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 AlwaysOn technologies, you should apply several WSFC-related prerequisites.
- For more information, see: Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups (SQL Server)

Instance-level High Availability with AlwaysOn Failover Cluster Instances

An AlwaysOn *Failover Cluster Instance* (FCI) is a SQL Server instance that is installed across nodes in a WSFC cluster. This type of instance has resource dependencies on shared disk storage (via Fibre Channel or iSCSI SAN) and on a virtual network name. The virtual network name has a resource dependency on one or more virtual IP addresses, each in a different subnet. The SQL Server service and the SQL Server Agent service are registered as resources, and both are made dependent upon the virtual network name resource.

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 AlwaysOn 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: AlwaysOn Failover Cluster Instances (FCI)

Database-level High Availability with AlwaysOn 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

- AlwaysOn 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: AlwaysOn Availability Groups (SQL Server)

WSFC Health Monitoring and Failover

High availability for an AlwaysOn solution is accomplished though 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: SQL Server Failover Policy

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: 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 AlwaysOn 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.

Important

- 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: Un-planned WSFC Quorum Failure with SQL Server

Relationship of SQL Server AlwaysOn Components to WSFC

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

AlwaysOn Availability Groups are hosted on SQL Server instances.

A client request to connect to a database **Primary Replica** or **Secondary Replica** on a logical **Availability Group Listener Network Name** is redirected to the appropriate **Instance**

Network Name of the underlying **SQL Server Instance** or **SQL Server Failover Cluster Instance**.

SQL Server instances are actively hosted on a single node.

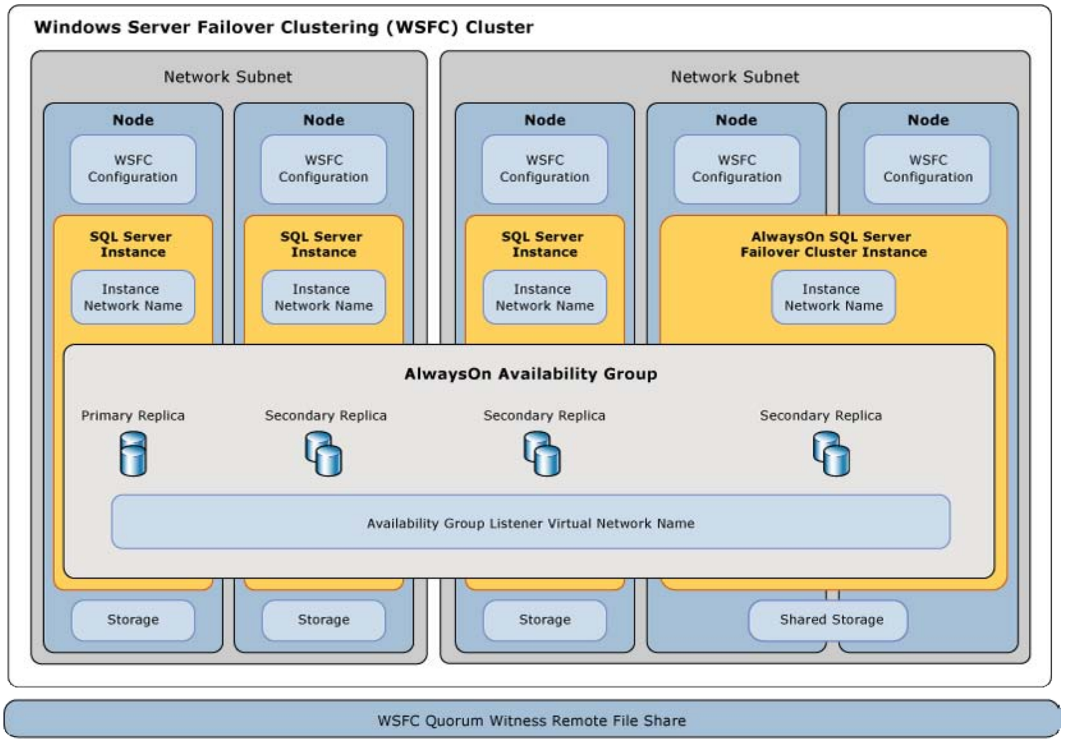
If present, a stand-alone **SQL Server Instance** always resides on a single **Node** with a static **Instance Network Name**. If present, a **SQL Server Failover Instance** is active on one of two or more possible failover **Nodes** with a single virtual **Instance Network Name**.

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.

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 witness constitute a quorum to vote on the health of the cluster.


Windows Server Failover Clustering (WSFC) Cluster
Network Subnet
Network Subnet
Node
Node
Node
Node
Node
WSFC Configuration
SQL Server Instance
Instance Network Name
AlwaysOn SQL Server Failover Cluster Instance
AlwaysOn Availability Group
Primary Replica
Secondary Replica
Secondary Replica
Secondary Replica
Availability Group Listener Virtual Network Name
Storage
Shared Storage
WSFC Quorum Witness Remote File Share

Related Tasks

Related Content

Microsoft SQL Server AlwaysOn Solutions Guide for High Availability and Disaster Recovery

Windows Server Technologies: Failover Clusters

Failover Clusters in Windows Server 2008 R2

See Also

SQL Server Failover Cluster Instances (AlwaysOn Failover Clustering)

AlwaysOn Availability Groups (SQL Server)

Quorum Modes and Voting Configuration (SQL Server)

SQL Server Failover Policy

Un-planned WSFC Quorum Failure with SQL Server

WSFC Quorum Modes and Voting Configuration

Both SQL Server AlwaysOn Availability Groups and AlwaysOn Failover Cluster Instances 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 AlwaysOn high availability and disaster recovery solution.

In this topic:

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 Quorum Failure with 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 AlwaysOn 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 node's vote, follow these guidelines:

- **Include all primary nodes.** Each node that hosts an AlwaysOn Availability Group primary replica or is the preferred owner of the AlwaysOn Failover Cluster Instance should have a vote.
- **Include possible automatic failover owners.** Each node that could host a primary replica or FCI, as the result of an automatic failover, should have a vote.

- **Exclude secondary site nodes.** In general, do not give votes to 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.

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 AlwaysOn Solutions Guide for High Availability and Disaster Recovery

Windows Server Technologies: Failover Clusters

Failover Cluster Step-by-Step Guide: Configuring the Quorum in a Failover Cluster

See Also

WSFC Quorum Failure with SQL Server

Windows Server Failover Clustering (WSFC) with SQL Server

View Cluster Quorum NodeWeight Settings

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 AlwaysOn Availability Groups and SQL Server Failover Cluster Instances.

- **Before you start:** Prerequisites, Security
- **To view quorum NodeWeight settings using:** Transact-SQL, PowerShell, 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

Quorum Modes and Voting Configuration

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

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 AlwaysOn Availability Groups and SQL Server Failover Cluster Instances.

- **Before you start:** Prerequisites, Security
- **To view quorum NodeWeight settings using:** PowerShell, 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 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 “AlwaysOnSrv1” node, and then outputs the settings for all nodes in the cluster.

```
Import-Module FailoverClusters


$node = “AlwaysOnSrv1”
```

```
(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 "AlwaysOnSrv1" node in the "Cluster001" cluster.

```
cluster.exe Cluster001 node AlwaysOnSrv1 /prop NodeWeight=0
```

See Also

Quorum Modes and Voting Configuration

View cluster quorum NodeWeight settings

Failover Cluster Cmdlets in Windows PowerShell Listed by Task Focus

WSFC Disaster Recovery through Forced Quorum

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** To Recover from Quorum Failure

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 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.

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 all nodes in the cluster were successfully restarted using the forced quorum procedure, and the root cause of the quorum failure has been corrected. Then changes to the original quorum mode and node vote configuration are not required.

   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 AlwaysOn 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 AlwaysOn 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:<network_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 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

See Also

Windows Server Failover Clustering (WSFC) with SQL Server

Force a WSFC Cluster to Start Without a Quorum

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 AlwaysOn Availability Groups and SQL Server Failover Cluster Instances.

- **Before you start:** Recommendations, Security
- **To force a cluster to start without a quorum using:** Failover Cluster Manager, PowerShell, net.exe
- **Follow up:** After forcing the 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 `–Propery NodeWieght = 1` to set the value the 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 AlwaysOnSrv02 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 = "AlwaysOnSrv02"

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

Warning

- 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 Quorum Modes and Voting Configuration.

Important

- 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 AlwaysOn 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, see Perform a Forced Manual Failover of an Availability Group (SQL Server).

See Also

Un-planned WSFC Quorum Failure with SQL Server

Configure Cluster Quorum NodeWeight Settings

Failover Cluster Cmdlets in Windows PowerShell Listed by Task Focus

SQL Server Multi-Subnet Clustering

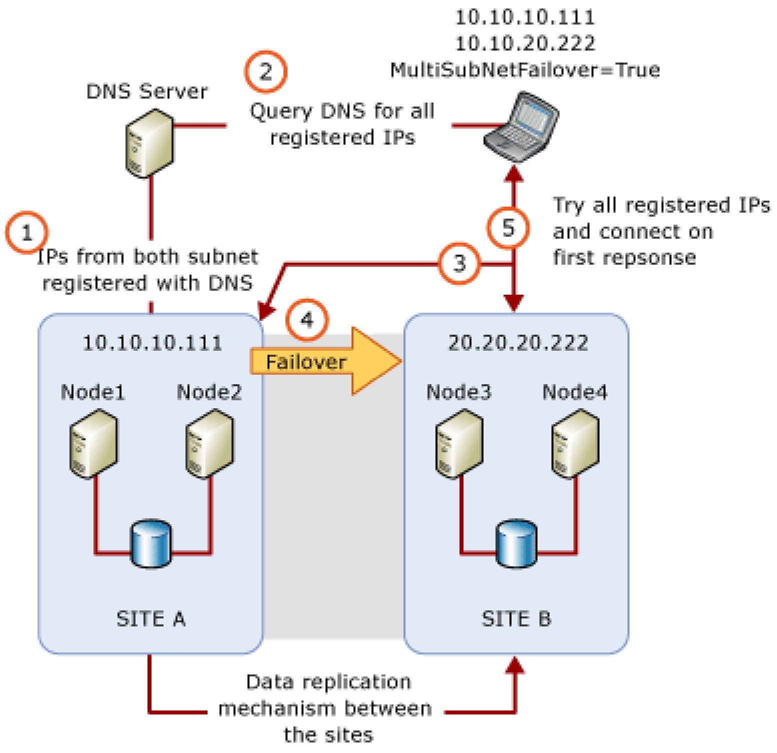
A SQL Server multi-subnet failover cluster is a configuration where each failover cluster node is connected to a different subnet or different set of subnets. These subnets can be in the same location or in geographically dispersed sites. Clustering across geographically dispersed sites is sometimes 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.

In this Topic:

- SQL Server Multi-Subnet Failover Cluster (Two-Nodes, Two-Subnets)
- IP Address Resource Considerations
- Client Recovery Latency During Failovers
- Related Content

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 2012.


10.10.10.111
10.10.20.222
MultiSubNetFailover=True
DNS Server
2
Query DNS for all registered IPs
1
IPs from both subnet registered with DNS
5
Try all registered IPs and connect on first repsonse
3
4
10.10.10.111
Failover
20.20.20.222
Node1
Node2
Node3
Node4
SITE A
SITE B
Data replication mechanism between the sites

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. This optional parameter is supported by the following data providers:

1. SQL Native Client 11.0
2. Data Provider for SQL Server in .NET Framework 4.02 or above

   Important
   For .NET Framework 4.02 specifically, you must also specify the TCP port of the database instance in your connection string.

3. Microsoft JDBC Driver 4.0 for 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 2012, 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	How to: Create a New SQL Server Failover Cluster (Setup)
In-place upgrade of your existing SQL Server Failover Cluster	How to: Upgrade a SQL Server Failover Cluster Instance (Setup)
Maintaining your existing SQL Server Failover Cluster	How to: Add or Remove Nodes in a SQL Server Failover Cluster (Setup)
Windows Failover Clustering	Windows 2008 R2 Failover Multi-Site Clustering

AlwaysOn Failover Cluster Instances

As part of the SQL Server AlwaysOn offering, AlwaysOn 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 AlwaysOn Availability Groups to provide remote disaster recovery at the database level. For more information, see Failover Clustering and AlwaysOn Availability Groups (SQL Server).

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 AlwaysOn 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 AlwaysOn 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

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 AlwaysOn 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, 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.	SQL Server Failover Cluster Adminstration and Maintenance

Concepts and Tasks	**Topic**
Describes multi-subnet configuration and concepts	SQL Server Multi-Subnet Clustering

Related Topics

Topic descriptions	Topic
Describes how to install a new SQL Server FCI.	How to: Create a New SQL Server Failover Cluster (Setup)
Describes how to upgrade to a SQL Server 2012 failover cluster.	Upgrading a SQL Server Failover Cluster
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
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 AlwaysOn Availability Groups (SQL Server)

Failover Policy for Failover Cluster Instances

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 2012 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 AlwaysOn availability group. However, manual failovers to and from and FCI are allowed in an AlwaysOn 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_subsytem 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 SQL Server Failover Cluster Diagnostics Log.

Tip
While the sp_server_diagnostic stored procedure is used by SQL Server AlwaysOn 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 in 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[1]	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'.
4	Failover or restart on	Indicates that a server restart or

Level	Condition	Description
	moderate server errors	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'. • System stored procedure sp_server_diagnostics returns 'resource error'.
5	Failover or restart on any qualified failure conditions	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'. • System stored procedure sp_server_diagnostics returns 'resource error'. • System stored procedure sp_server_diagnostics returns 'query_processing error'.

[1] 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

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 AlwaysOn 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 HeathCheckTimeout setting, using:** PowerShell,Failover Cluster Manager, Transact-SQL

Before You Begin

Limitations and Restrictions

The default value for this property is 60,000 milliseconds (60 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 commandlets.
3. Use the `Get-ClusterResource` object to set the `HealthCheckTimeout` property for the failover cluster instance.

Example (Powershell)

The following example changes the HealthCheckTimeout setting on the “AlwaysOnSrv1” failover cluster instance to 60000 milliseconds.

```
Import-Module FailoverClusters


$fci = “AlwaysOnSrv1”

Get-ClusterResource $fci | Set-ClusterParameter HealthCheckTimeout 60000
```

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

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 specify the **HealthCheckTimeOut** property value. For syntax details, see Setting failover cluster properties

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

Failure Detection in SQL Server Failover Cluster

Configure FailureConditionLevel Property Settings

Use the FailureConditionLevel property to set the conditions for the AlwaysOn 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:** Limitations and Restrictions, Security
- **To configure 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 Determining Failures

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 commandlets.
3. Use the `Get-ClusterResource` object to set the `FailureConditionLevel` property for Failover Cluster Instance.

Example (Powershell)

The following example changes the FailureConditionLevel setting on the “AlwaysOnSrv1” failover cluster instance to failover or restart on critical server errors.

```
Import-Module FailoverClusters


$fci = “AlwaysOnSrv1”

Get-ClusterResource $fci | Set-ClusterParameter FailureConditionLevel 3
```

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 **FaliureConditionLevel** property, and then click **OK** to apply the change.

Using Transact-SQL

To configure FailureConditionLevel property settings:

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 specify the **FailureConditionLevel** property value. For syntax details, see Setting failover cluster properties

Example (Transact-SQL)

The following example sets the `FailureConditionLevel` property to 0 indicating that no failover or restart will 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

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 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\MSSQL11.<InstanceName>\MSSQL\LOG' of the owning node of the AlwaysOn 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 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
( SELECTobject_name AS 'event'
,CONVERT(xml,event_data) AS 'xml_data'
  FROM
  sys.fn_xe_file_target_read_file('<path to the file>','<path to the metadata
file>',NULL,NULL)
)
```

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 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

Failure Detection in SQL Server Failover Cluster

Failover Cluster Instance Administration and Maintenance

Maintenance tasks like adding or removing nodes from an existing AlwaysOn 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 Before Installing Failover Clustering and How to: 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 How to: 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 Setting Up Windows Service Accounts.

Administering a Failover Cluster Instance

Task Description	Topic Link
Describes how to add dependencies to a SQL Server resource.	How to: 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 AlwaysOn FCIs.	Registering a Service Principal Name.
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 Failure
Describe the procedure used to change the IP address resource for a SQL Server failover cluster instance.	Change the IP Address of a SQL Server Failover Cluster

See Also

How to: Configure HealthCheckTimeout Settings

How to: Configure FailureConditionLevel Property Settings

How to: View and Read SQL Server Failover Cluster Diagnostics Log

Add Dependencies to a SQL Server Resource

This topic describes how to add dependencies to an AlwaysOn 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

This topic describes how to recover from cluster failures by using the Failover Cluster Manager snap-in after a failover occurs in SQL Server 2012. The Failover Cluster Manager snap-in is the cluster management application for the Windows Serer 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 How to: Read a SQL Server Setup Log File.
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 How to: Add or Remove Nodes in a SQL Server 2005 Failover Cluster (Setup).

Recover from a reparable failure

Us 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

This topic describes how to change the IP address resource in an AlwaysOn 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
- **To change the IP address resource, using:** Transact-SQL,

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.

AlwaysOn Availability Groups

The AlwaysOn Availability Groups feature is a high-availability and disaster-recovery solution that provides an enterprise-level alternative to database mirroring. Introduced in SQL Server 2012, AlwaysOn Availability Groups maximizes the availability of a set of user databases for an enterprise. An *availability group* supports a failover environment for a discrete set of user databases, known as *availability databases*, that fail over together. An availability group supports a set of read-write primary databases and one to four sets of corresponding secondary databases. Optionally, secondary databases can be made available for read-only access and/or some backup operations.

An availability group fails over at the level of an availability replica. Failovers are not caused by database issues such as a database becoming suspect due to a loss of a data file, deletion of a database, or corruption of a transaction log.

In this Topic:

- Benefits
- Terms and Definitions
- Interoperability and Coexistence with Other Database Engine Features
- Related Tasks
- Related Content

Benefits

AlwaysOn Availability Groups provides a rich set of options that improve database availability and that enable improved resource use. The key components are as follows:

- Supports up to five availability replicas. An *availability replica* is an instantiation of an availability group that is hosted by a specific instance of SQL Server and maintains a local copy of each availability database that belongs to the availability group. Each availability group supports one primary replica and up to four secondary replicas. For more information, see Overview of AlwaysOn Availability Groups.

  Important
  Each availability replica must reside on a different node of a single Windows Server Failover Clustering (WSFC) cluster. For more information about prerequisites, restrictions, and recommendations for availability groups, see Considerations for Deploying AlwaysOn Availability Groups (SQL Server).

- Supports alternative availability modes, as follows:
  - *Asynchronous-commit mode*. This availability mode is a disaster-recovery solution that works well when the availability replicas are distributed over considerable distances.
  - *Synchronous-commit mode*. This availability mode emphasizes high availability and data protection over performance, at the cost of increased transaction latency. A given

    availability group can support up to three synchronous-commit availability replicas, including the current primary replica.

  For more information, see Availability Modes (AlwaysOn Availability Groups).

- Supports several forms of availability-group failover: automatic failover, planned manual failover (generally referred as simply "manual failover"), and forced manual failover (generally referred as simply "forced failover"). For more information, see Failover Modes (AlwaysOn Availability Groups).
- Enables you to configure a given availability replica to support either or both of the following active-secondary capabilities:
    - Read-only connection access which enables read-only connections to the replica to access and read its databases when it is running as a secondary replica. For more information, see Read-Only Access to Secondary Replicas (AlwaysOn Availability Groups).
    - Performing backup operations on its databases when it is running as a secondary replica. For more information, see Backup on Secondary Replicas (AlwaysOn Availability Groups).

  Using active secondary capabilities improves your IT efficiency and reduce cost through better resource utilization of secondary hardware. In addition, offloading read-intent applications and backup jobs to secondary replicas helps to improve performance on the primary replica.
- Supports an availability group listener for each availability group. An *availability group listener* is a server name to which clients can connect in order to access a database in a primary or secondary replica of an AlwaysOn availability group. Availability group listeners direct incoming connections to the primary replica or to a read-only secondary replica. The listener provides fast application failover after an availability group fails over. For more information, see Configuring Client Connectivity (AlwaysOn Availability Groups).
- Supports a flexible failover policy for greater control over availability-group failover. For more information, see Failover Modes (AlwaysOn Availability Groups).
- Supports automatic page repair for protection against page corruption. For more information, see Automatic Page Repair (Availability Groups/Database Mirroring).
- Supports encryption and compression, which provide a secure, high performing transport.
- Provides an integrated set of tools to simplify deployment and management of availability groups, including:
    - Tsql DDL statements for creating and managing availability groups. For more information, see Overview of Transact-SQL Statements for AlwaysOn Availability Groups.
    - SQL Server Management Studio tools, as follows:
        - The New Availability Group Wizard creates and configures an availability group. In some environments, this wizard can also automatically prepare the secondary databases and start data synchronization for each of them. For more information, see Create and Configure an Availability Group (New Availability Group Wizard).
        - The Add Database to Availability Group Wizard adds one or more primary databases to an existing availability group. In some environments, this wizard can also

    automatically prepare the secondary databases and start data synchronization for each of them. For more information, see Use the Add Database to Availability Group Wizard (SQL Server).
    - The Add Replica to Availability Group Wizard adds one or more secondary replicas to an existing availability group. In some environments, this wizard can also automatically prepare the secondary databases and start data synchronization for each of them. For more information, see Use the Add Replica to Availability Group Wizard (SQL Server).
    - The Fail Over Availability Group Wizard initiates a manual failover on an availability group. Depending on the configuration and state of the secondary replica that you specify as the failover target, the wizard can perform either a planned or forced manual failover. For more information, see Use the Failover Availability Group Wizard (SQL Server).
- The AlwaysOn Dashboard monitors AlwaysOn availability groups, availability replicas, and availability databases and evaluates results for AlwaysOn policies. For more information, see Use the Availability Group Dashboard (SQL Server Management Studio).
- The Object Explorer Details pane displays basic information about existing availability groups. For more information, see Use Object Explorer Details to Monitor Availability Groups (SQL Server Management Studio).
- PowerShell cmdlets. For more information, see Overview of PowerShell Cmdlets for AlwaysOn Availability Groups (SQL Server).

Terms and Definitions

availability group

A container for a set of databases, *availability databases*, that fail over together.

availability database

A database that belongs to an availability group. For each availability database, the availability group maintains a single read-write copy (the *primary database*) and one to four read-only copies (*secondary databases*).

primary database

The read-write copy of an availability database.

secondary database

A read-only copy of an availability database.

availability replica

An instantiation of an availability group that is hosted by a specific instance of SQL Server and maintains a local copy of each availability database that belongs to the availability group. Two types of availability replicas exist: a single *primary replica* and one to four *secondary* *replicas*.

primary replica

The availability replica that makes the primary databases available for read-write connections from clients and, also, sends transaction log records for each primary database to every secondary replica.

secondary replica

An availability replica that maintains a secondary copy of each availability database, and serves as a potential failover targets for the availability group. Optionally, a secondary replica can support read-only access to secondary databases can support creating backups on secondary databases.

availability group listener

A server name to which clients can connect in order to access a database in a primary or secondary replica of an AlwaysOn availability group. Availability group listeners direct incoming connections to the primary replica or to a read-only secondary replica.

Note

For more information, see Overview of AlwaysOn Availability Groups (SQL Server).

Interoperability and Coexistence with Other Database Engine Features

AlwaysOn Availability Groups can be used with the following features or components of SQL Server:

- Change Data Capture
- Change Tracking
- Contained databases
- Database encryption
- Database snapshots
- FILESTREAM
- FileTable
- Log shipping
- Remote Blob Store (RBS)
- Replication
- Service Broker
- SQL Server Agent
- Reporting Services

Warning

For information about restrictions and limitations for using other features with AlwaysOn Availability Groups, see AlwaysOn Availability Groups: Interoperability and Coexistence.

Related Tasks

- Getting Started with AlwaysOn Availability Groups (SQL Server)

Related Content

- **Video—Introduction to AlwaysOn:** Microsoft SQL Server Code-Named "Denali" AlwaysOn Series,Part 1: Introducing the Next Generation High Availability Solution
- **Video—A Deep Dive into AlwaysOn:** Microsoft SQL Server Code-Named "Denali" AlwaysOn Series,Part 2: Building a Mission-Critical High Availability Solution Using AlwaysOn
- **Whitepaper:** Microsoft SQL Server AlwaysOn Solutions Guide for High Availability and Disaster Recovery
- **Blogs:** SQL Server AlwaysOn Team Blog: The official SQL Server AlwaysOn Team Blog

See Also

Overview of AlwaysOn Availability Groups (SQL Server)

Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups (SQL Server)

Deployment: Configuring a Server Instance for AlwaysOn Availability Groups (SQL Server)

Creation and Configuration of Availability Groups (SQL Server)

Administration of an Availability Group (SQL Server)

Monitoring of Availability Groups (SQL Server)

Overview of Transact-SQL Statements

Overview of PowerShell Cmdlets for AlwaysOn Availability Group (SQL Server)

Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups

This topic describes considerations for deploying AlwaysOn Availability Groups, including prerequisites, restrictions, and recommendations for host computers, Windows Server Failover Clustering (WSFC) clusters, server instances, and availability groups. For each of these components security considerations and required permissions, if any, are indicated.

Important
Before you deploy AlwaysOn Availability Groups, we strongly recommend that you read every section of this topic.

In this Topic:

- .Net Hotfixes that Support AlwaysOn Availability Groups

- Windows System Requirements and Recommendations
- SQL Server Instance Prerequisites and Restrictions
- Network Connectivity Recommendations
- Prerequisites and Restrictions for Using a SQL Server Failover Cluster Instance (FCI) to Host an Availability Replica
- Availability Group Prerequisites and Restrictions
- Availability Database Prerequisites and Restrictions
- Related Content

.Net Hotfixes that Support AlwaysOn Availability Groups

Depending on the SQL Server 2012 components and features you will use with AlwaysOn Availability Groups, you may need to install additional .Net hotfixes identified in the following table. The hotfixes can be installed in any order.

	Dependent Feature	Hotfix	Link
☐	Reporting Services	Hotfix for .Net 3.5 SP1 adds support to SQL Client for AlwaysOn features of Read-intent, readonly, and multisubnetfailover. The hotfix needs to be installed on each Reporting Services report server.	KB 2654347: Hotfix for .Net 3.5 SP1 to add support for AlwaysOn features

Windows System Requirements and Recommendations

In This Section:

- Checklist: Requirements
- Windows Hotfixes that Support AlwaysOn Availability Groups (Windows System)
- Recommendations for Computers That Host Availability Replicas (Windows System
- Permissions
- Related Tasks

Checklist: Requirements (Windows System)

To support the AlwaysOn Availability Groups feature, ensure that every computer that is to participate in one or more availability groups meets the following fundamental requirements:

	Requirement	Link
☐	Ensure that the system is not a domain controller.	Availability groups are not supported on domain controllers.
☐	Ensure that each computer is running either x86 (non-WOW64) or x64 Windows Server 2008 or later versions.	WOW64 (Windows 32-bit on Windows 64-bit) does not support AlwaysOn Availability Groups.
☐	Ensure that each computer is a node in a Windows Server Failover Clustering (WSFC) cluster.	Windows Server Failover Clustering (WSFC) with SQL Server
☐	Ensure that the WSFC cluster contains sufficient nodes to support your availability group configurations.	A WSFC node can host only one availability replica for a given availability group. On a given WSFC node, one or more instances of SQL Server can host availability replicas for many availability groups. Ask your database administrators how many WSFC nodes are required for to support the availability replicas of the planned availability groups. Conceptual Overview of AlwaysOn Availability Groups (SQL Server).
☐	Ensure that all applicable Window hotfixes have been installed on every node in the WSFC cluster.	**Important** A number of hotfixes are required or recommended for the nodes of a WSFC cluster on which AlwaysOn Availability Groups is being deployed. For more information, see Windows Hotfixes that Support AlwaysOn Availability Groups (Windows System), later in this section.

Important
Also ensure that your environment is correctly configured for connecting to an availability group. For more information, see Prerequisites, Restrictions, and Recommendations for AlwaysOn Client Connectivity.

Windows Hotfixes that Support AlwaysOn Availability Groups (Windows System)

Depending on your cluster topology, several additional Windows Server 2008 Service Pack 2 (SP2) or Windows Server 2008 R2 hotfixes might be applicable for supporting AlwaysOn Availability Groups. The following table identifies these hotfixes. They hotfixes can be installed in any order.

	Applies to Win 2008 SP2	Applies to Win 2008 R2 SP1	To Support…	Hotfix	Link
☐	√	√	**Configuring optimal WSFC quorum**	On each WSFC node, ensure that the hotfix described in Knowledge Base article 2494036 is installed. This hotfix supports configuring optimal quorum with non-automatic failover targets. This functionality improves multi-site clusters by enabling you to select which nodes vote.	KB 2494036: 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 For information about quorum voting, see WSFC Quorum Modes and Voting Configuration (SQL Server)
☐	√	√	**More efficient use of network bandwidth**	On each WSFC node, ensure that the hotfix described in Knowledge Base	KB 2616514: Cluster service sends unnecessary registry key change notifications among

	Applies to Win 2008 SP2	Applies to Win 2008 R2 SP1	To Support…	Hotfix	Link
				article 2616514 is installed. Without this hotfix, the Cluster service sends unnecessary registry notifications among cluster nodes. This behavior limits network bandwidth, which is a serious issue for AlwaysOn Availability Groups.	cluster nodes in Windows Server 2008 or in Windows Server 2008 R2
☐		√	**VPD storage testing on disks that are not available to all WSFC nodes**	If a WSFC node is running Windows Server 2008 R2 Service Pack 1 (SP1) and the Validate SCSI Device Vital Product Data (VPD) storage test fails after incorrectly running on disks that are online and not available to all nodes in the WSFC cluster, install the hotfix described in Knowledge Base article 2531907. This hotfix	KB 2531907: Validate SCSI Device Vital Product Data (VPD) test fails after you install Windows Server 2008 R2 SP1

	Applies to Win 2008 SP2	Applies to Win 2008 R2 SP1	To Support…	Hotfix	Link
				eliminates incorrect warnings or errors in the validation report when disks are online.	
☐		√	**Faster failover to local replicas**	If a WSFC node is running Windows Server 2008 R2 Service Pack 1 (SP1), ensure that the hotfix described in Knowledge Base article 2687741 is installed. This hotfix improves the performance of AlwaysOn Availability Groups failover to local replicas.	KB 2687741: A hotfix that improves the performance of the "AlwaysOn Availability Group" feature in SQL Server 2012 is available for Windows Server 2008 R2
☐	√	√	**Asymmetric storage—for Failover Cluster Instances (FCIs)**	If any Failover Cluster Instance (FCI) will be enabled for AlwaysOn Availability Groups, install the Windows Server 2008 hotfix 976097. This hotfix enables the Failover Cluster Management	KB 976097: Hotfix to add support for asymmetric storages to the Failover Cluster Management MMC snap-in for a failover cluster that is running Windows Server 2008 or Windows Server 2008 R2

	Applies to Win 2008 SP2	Applies to Win 2008 R2 SP1	To Support…	Hotfix	Link
				Microsoft Management Console (MMC) snap-in to support asymmetric storage—shared disks that are available on only some of the WSFC nodes.	
☐	√	√	**Internet Protocol Security (IPsec)**	If your environment uses IPsec connections, you could experience a long time delay (about two or three minutes) when a client computer reestablishes the IPsec connection to a virtual network name (in this context, to connect to the availability group listener). If you use IPsec connections, we recommend that you review the specific scenarios detailed in Knowledge Base article (KB 980915).	KB 980915: A long time delay occurs when you reconnect an IPSec connection from a computer that is running Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, or Windows Server 2008 R2

	Applies to Win 2008 SP2	Applies to Win 2008 R2 SP1	To Support...	Hotfix	Link
☐	√	√	**IPv6**	If you use IPv6, we recommend that you review the specific scenarios detailed in Knowledge Base article 2578103 or 2578113, depending on your Windows Server operating system. If your Windows Server topology uses IP version 6 (IPv6), the WSFC Cluster service requires about 30 seconds to fail over the IPv6 IP address. This causes clients to wait for about 30 seconds to reconnect to the IPv6 IP address.	• KB 2578103 (Windows Server 2008): The Cluster service takes about 30 seconds to fail over IPv6 IP addresses in Windows Server 2008 • KB 2578113 (Windows Server 2008 R2): **Windows Server 2008 R2:** The Cluster service takes about 30 seconds to fail over IPv6 IP addresses in Windows Server 2008 R2
☐	√	√	**No Router Between cluster and application server**	If no router exists between the failover cluster and the application server, the Cluster service fails over network-related resources slowly. This delays client	KB 2582281: Slow failover operation if no router exists between the cluster and an application server

	Applies to Win 2008 SP2	Applies to Win 2008 R2 SP1	To Support...	Hotfix	Link
				reconnections after an availability group fails over. In the absence of a router, we recommend that you review the specific scenarios detailed in Knowledge Base article 2582281 and install the hotfix, if applicable to your environment.	

Top

Recommendations for Computers That Host Availability Replicas (Windows System)

- **Comparable systems:** For a given availability group, all the availability replicas should run on comparable systems that can handle identical workloads.
- **Dedicated network adapters:** For best performance, use a dedicated network adapter (network interface card) for AlwaysOn Availability Groups.
- **Sufficient disk space:** Every computer on which a server instance hosts an availability replica must possess sufficient disk space for all the databases in the availability group. Keep in mind that as primary databases grow, their corresponding secondary databases grow the same amount.

Permissions (Windows System)

To administer a WSFC cluster, the user must be a system administrator on every cluster node.

For more information about account for administering the cluster, see Appendix A: Failover Cluster Requirements.

Related Tasks (Windows System)

Task	Topic
Set the HostRecordTTL (to the recommended 60 seconds)	• Configure DNS settings in a Multi-Site Failover Cluster • DNS Registration with Network Name Resource

Note
For more information, see Windows 2008 R2 Failover Multi-Site Clustering

SQL Server Instance Prerequisites and Restrictions

Each availability group requires a set of failover partners, known as *availability replicas*, which are hosted by instances of SQL Server. A given server instance can be a *stand-alone instance* or a SQL Server *failover cluster instance* (FCI).

In This Section:

- Checklist: Prerequisites
- Restrictions
- Permissions
- Related Tasks

Checklist: Prerequisites (Server Instance)

	Prerequisite	Links
☐	The host computer must be a Windows Server Failover Clustering (WSFC) node. The instances of SQL Server that host availability replicas for a given availability group must reside on separate nodes of a single WSFC cluster.	Windows Server Failover Clustering (WSFC) with SQL Server Failover Clustering and AlwaysOn Availability Groups (SQL Server)
☐	If you want an availability group to work with Kerberos: • All server instances that host an availability replica for the availability group must use the same SQL Server service	Register a Service Principal Name for Kerberos Connections **Brief explanation:** Kerberos and SPNs enforce mutual authentication. The SPN maps to the Windows account

	Prerequisite	Links
	account. • The domain administrator needs to manually register a Service Principal Name (SPN) with Active Directory on the SQL Server service account for the virtual network name (VNN) of the availability group listener. If the SPN is registered on an account other than the SQL Server service account, authentication will fail. **Important** If you change the SQL Server service account, the domain administrator will need to manually re-register the SPN.	that starts the SQL Server services. If the SPN is not registered correctly or if it fails, the Windows security layer cannot determine the account associated with the SPN, and Kerberos authentication cannot be used. **Note** NTLM does not have this requirement.
☐	If you plan to use a SQL Server failover cluster instance (FCI) to host an availability replica, ensure that you understand the FCI restrictions and that the FCI requirements are met.	Restrictions and Requirements on Using a SQL Server Failover Cluster Instance (FCI) to Host an Availability Replica (later in this topic)
☐	Each server instance must be running the Enterprise Edition of SQL Server 2012.	Features Supported by the Editions of SQL Server "Denali"
☐	All the server instances that host availability replicas for an availability group must use the same SQL Server collation.	Set or Change the Server Collation
☐	Enable the AlwaysOn Availability Groups feature on each server instance that will host an availability replica for any availability group. On a given computer, you can enable as many server instances for AlwaysOn Availability Groups as	Enable and Disable the AlwaysOn Availability Groups Feature (SQL Server) **Important** If you delete and re-create a WSFC cluster, you must disable and re-

	Prerequisite	Links
	your SQL Server installation supports.	enable the AlwaysOn Availability Groups feature on each server instance that was enabled for AlwaysOn Availability Groups on the original WSFC cluster.
☐	Each server instance requires a database mirroring endpoint. Note that this endpoint is shared by all the availability replicas and database mirroring partners and witnesses on the server instance. **noteDXDOC112778PADS Security Note** Transport security for AlwaysOn Availability Groups is the same as for database mirroring.	Database Mirroring Endpoint Transport Security for Database Mirroring and AlwaysOn Availability Groups
☐	If any databases that use FILESTREAM will be added to an availability group, ensure that FILESTREAM is enabled on every server instance that will host an availability replica for the availability group.	Enable and Configure FILESTREAM
☐	If any contained databases will be added to an availability group, ensure that the **contained database authentication** server option is set to **1** on every server instance that will host an availability replica for the availability group.	contained database authentication Option Set Server Configuration Options

Permissions (Server Instance)

Task	**Required Permissions**
Creating the database mirroring endpoint	Requires CREATE ENDPOINT permission, or membership in the **sysadmin** fixed server role. Also requires CONTROL ON ENDPOINT permission. For more information, see GRANT Endpoint Permissions (Transact-SQL).
Enabling AlwaysOn Availability Groups	Requires membership in the **Administrator** group on the local computer and full control on the WSFC cluster.

Related Tasks (Server Instance)

Task	**Topic**
Determining whether database mirroring endpoint exists	sys.database_mirroring_endpoints (Transact-SQL)
Creating the database mirroring endpoint (if it does not yet exist)	• Create a Mirroring Endpoint for Windows Authentication (Transact-SQL) • Use Certificates for a Database Mirroring Endpoint (SQL Server) • Create a Database Mirroring Endpoint for AlwaysOn Availability Groups (SQL Server PowerShell)
Enabling AlwaysOn Availability Groups	Enable and Disable the AlwaysOn Availability Groups Feature (SQL Server)

Network Connectivity Recommendations

We strongly recommend that you use the same network links for communications between WSFC cluster members and communications between availability replicas. Using separate network links can cause unexpected behaviors if some of links fail (even intermittently).

For example, for an availability group to support automatic failover, the secondary replica that is the automatic-failover partner must be in the SYNCHRONIZED state. If the network link to this secondary replica fails (even intermittently), the replica enters the UNSYNCHRONIZED state and cannot begin to resynchronize until the link is restored. If the WSFC cluster requests an automatic failover while the secondary replica is unsynchronized, automatic failover will not occur.

Prerequisites and Restrictions for Using a SQL Server Failover Cluster Instance (FCI) to Host an Availability Replica

In This Section:

- Restrictions
- Checklist: Prerequisites
- Related Tasks

Restrictions (FCIs)

- **The cluster nodes of an FCI can host only one replica for a given availability group:** If you add an availability replica on an FCI, the WSFC cluster nodes that are possible FCI owners cannot host another replica for the same availability group. Every other replica must be hosted by an instance of SQL Server 2012 that resides on a different WSFC node in the same WSFC cluster.
- **FCIs do not support automatic failover by availability groups:** FCIs do not support automatic failover by availability groups, so any availability replica that is hosted by an FCI can be configured for manual failover only.
- **Changing FCI network name:** If you need to change the network name of an FCI that hosts an availability replica, you will need to remove the replica from its availability group and then add the replica back into the availability group. You cannot remove the primary replica, so if you are renaming an FCI that is hosting the primary replica, you should fail over to a secondary replica and then remove the former primary replica and add it back. Note that renaming an FCI might alter the URL of its database mirroring endpoint. When you add the replica ensure that you specify the current endpoint URL.

Checklist: Prerequisites (FCIs)

	Prerequisite	**Link**
☐	Before you use an FCI to host an availability replica, ensure that your system administrator has installed the Windows Server 2008 hotfix described in Knowledge Base article KB 976097. This hotfix enables the Failover Cluster Management Microsoft Management	KB 976097: Hotfix to add support for asymmetric storages to the Failover Cluster Management MMC snap-in for a failover cluster that is running Windows Server 2008 or Windows Server 2008 R2

	Prerequisite	Link
	Console (MMC) snap-in to support asymmetric storage—shared disks that are available on only some of the WSFC nodes.	
☐	Ensure that each SQL Server failover cluster instance (FCI) possesses the required shared storage as per standard SQL Server failover cluster instance installation.	

Related Tasks (FCIs)

Task	Topic
Installing a SQL Server Failover Cluster	How to: Create a New SQL Server Failover Cluster (Setup)
In-place upgrade of your existing SQL Server Failover Cluster	How to: Upgrade a SQL Server Failover Cluster Instance (Setup)
Maintaining your existing SQL Server Failover Cluster	How to: Add or Remove Nodes in a SQL Server Failover Cluster (Setup)

For more information, see Failover Clustering and AlwaysOn Availability Groups (SQL Server).

Availability Group Prerequisites and Restrictions

In This Section:

- Restrictions
- Requirements
- Security
- Related Tasks

Restrictions (Availability Groups)

- **Availability replicas must be hosted by different nodes of one WSFC cluster:** For a given availability group, availability replicas must be hosted by server instances running on different nodes of the same WSFC cluster.

  Note
  Virtual machines on the same physical computer can each host an availability replica for the same availability group because each virtual machine acts as a separate computer.

- **Unique availability group name:** Each availability group name must be unique on the WSFC cluster. The maximum length for an availability group name is 128 characters.
- **Maximum number of availability groups and availability databases per computer:** The actual number of databases and availability groups you can put on a computer (VM or physical) depends on the hardware and workload, but there is no enforced limit. Microsoft has extensively tested with 10 AGs and 100 DBs per physical machine. Signs of overloaded systems can include, but are not limited to, worker thread exhaustion, slow response times for AlwaysOn system views and DMVs, and/or stalled dispatcher system dumps. Please make sure to thoroughly test your environment with a production-like workload to ensure it can handle peak workload capacity within your application SLAs. When considering SLAs be sure to consider load under failure conditions as well as expected response times.
- **Do not use the Failover Cluster Manager to manipulate availability groups:**

  For example:

  - Do not change any availability group properties, such as the possible owners.
  - Do not use the Failover Cluster Manager to fail over availability groups. You must use Transact-SQL or SQL Server Management Studio.

Prerequisites (Availability Groups)

When creating or reconfiguring an availability group configuration, ensure that you adhere to the following requirements.

	Prerequisite	**Description**
☐	If you plan to use a SQL Server failover cluster instance (FCI) to host an availability replica, ensure that you understand the FCI restrictions and that the FCI requirements are met.	Prerequisites and Restrictions for Using a SQL Server Failover Cluster Instance (FCI) to Host an Availability Replica (earlier in this topic)

Security (Availability Groups)

- Security is inherited from the Windows Server Failover Clustering (WSFC) cluster. WSFC provides two levels of user security at granularity of entire WSFC cluster APIs:

    - Read-only access
    - Full control

      AlwaysOn Availability Groups needs full control, and enabling AlwaysOn Availability Groups on an instance of SQL Server gives it full control of the WSFC cluster (through Service SID).

      You cannot directly add or remove security for a server instance in the WSFC Failover Cluster Manager. To manage WSFC security sessions, use the SQL Server Configuration Manager or the WMI equivalent from SQL Server.
- Each instance of SQL Server must have permissions to access the registry, cluster, and so forth.
- We recommend that you use encryption for connections between server instances that host AlwaysOn Availability Groups availability replicas.

Permissions (Availability Groups)

Task	**Required Permissions**
Creating an availability group	Requires membership in the **sysadmin** fixed server role and either CREATE AVAILABILITY GROUP server permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.
Altering an availability group	Requires ALTER AVAILABILITY GROUP permission on the availability group, CONTROL AVAILABILITY GROUP permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission. In addition, joining a database to an availability group requires membership in the **db_owner** fixed database role.
Dropping/deleting an availability group	Requires ALTER AVAILABILITY GROUP permission on the availability group, CONTROL AVAILABILITY GROUP permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission. To drop an availability group that is not hosted on the local replica location you need CONTROL SERVER permission or CONTROL permission on that

Task	Required Permissions
	Availability Group.

Related Tasks (Availability Groups)

Task	Topic
Creating an availability group	• Use the Availability Group (New Availability Group Wizard) • Create an Availability Group (Transact-SQL) • Create an Availability Group (SQL Server PowerShell) • Specify the Endpoint URL When Adding or Modifying an Availability Replica (SQL Server)
Modifying the number of availability replicas	• Add a Secondary Replica to an Availability Group (SQL Server) • Join a Secondary Replica to an Availability Group (SQL Server) • Remove a Secondary Replica from an Availability Group (SQL Server)
Creating an availability group listener	Create or Configure an Availability Group Listener (SQL Server)
Dropping an availability group	Deleting an Availability Group (SQL Server)

Availability Database Prerequisites and Restrictions

To be eligible to be added to an availability group, a database must meet the following prerequisites and restrictions.

In This Section:

- Requirements
- Restrictions
- Recommendations for Computers That Host Availability Replicas (Windows System
- Permissions

- Related Tasks

Checklist: Requirements (Availability Databases)

To be eligible to be added to an availability group, a database must:

	Requirements	**Link**
☐	Be a user database. System databases cannot belong to an availability group.	
☐	Reside on the instance of SQL Server where you create the availability group and be accessible to the server instance.	
☐	Be a read-write database. Read-only databases cannot be added to an availability group.	sys.databases (**is_read_only** = 0)
☐	Be a multi-user database.	sys.databases (**user_access** = 0)
☐	Not use AUTO_CLOSE.	sys.databases (**is_auto_close_on** = 0)
☐	Use the full recovery model (also known as, full recovery mode).	sys.databases (**recovery_model** = 1)
☐	Possess at least one full database backup. **Note** After setting a database to full recovery mode, a full backup is required to initiate the full-recovery log chain.	Create a Full Database Backup
☐	Not belong to any existing availability group.	sys.databases (**group_database_id** = NULL)
☐	Not be configured for database mirroring.	sys.database_mirroring (If the database does not participate in mirroring, all columns prefixed with "mirroring_" are NULL.)

	Requirements	Link
☐	Before adding a database that uses FILESTREAM to an availability group, ensure that FILESTREAM is enabled on every server instance that hosts or will host an availability replica for the availability group.	Enable and Configure FILESTREAM
☐	Before adding a contained database to an availability group, ensure that the **contained database authentication** server option is set to **1** on every server instance that hosts or will host an availability replica for the availability group.	contained database authentication Option Set Server Configuration Options

Note
AlwaysOn Availability Groups works with any supported database compatibility level.

Restrictions (Availability Databases)

- If the file path (including the drive letter) of a secondary database differs from the path of the corresponding primary database, the following restrictions apply:
    - **New Availability Group Wizard/Add Database to Availability Group Wizard:** The **Full** option is not supported (on the Select Initial Data Synchronization Page page),
    - **RESTORE WITH MOVE:** To create the secondary databases, the database files must be RESTORED WITH MOVE on each instance of SQL Server that hosts a secondary replica.
    - **Impact on add-file operations:** A later add-file operation on the primary replica might fail on the secondary databases. This failure could cause the secondary databases to be suspended. This, in turn, causes the secondary replicas to enter the NOT SYNCHRONIZING state.

      Note
      For information about responding to a failed ad-file operation, see Troubleshoot a Failed Add-File Operation (AlwaysOn Availability Groups).
- You cannot drop a database that currently belongs to an availability group.

Follow Up for TDE Protected Databases

If you use transparent data encryption (TDE), the service master key for creating and decrypting other keys must be the same on every server instance that hosts an availability replica for the availability group. For more information, see Moving a TDE Protected Database to Another SQL Server.

Permissions (Availability Databases)

Requires ALTER permission on the database.

Related Tasks (Availability Databases)

Task	Topic
Preparing a secondary database (manually)	Prepare a Secondary Database for an Availability Group (SQL Server)
Joining a secondary database to availability group (manually)	Join a Secondary Database to an Availability Group (SQL Server)
Modifying the number of availability databases	• Add a Database to an Availability Group (SQL Server) • Remove a Database from a Secondary Replica (AlwaysOn Availability Groups) • Remove a Database from an Availability Group (AlwaysOn Availability Groups)

Related Content

- Microsoft SQL Server AlwaysOn Solutions Guide for High Availability and Disaster Recovery
- SQL Server AlwaysOn Team Blog: The official SQL Server AlwaysOn Team Blog

See Also

AlwaysOn Availability Groups (SQL Server)

Failover Clustering and AlwaysOn Availability Groups (SQL Server)

Prerequisites, Restrictions, and Recommendations for AlwaysOn Client Connectivity

Failover Clustering and AlwaysOn Availability Groups

AlwaysOn Availability Groups, the high availability and disaster recovery solution introduced in SQL Server 2012, requires Windows Server Failover Clustering (WSFC). Also, though AlwaysOn

Availability Groups is not dependent upon SQL Server Failover Clustering, you can use a failover clustering instance (FCI) to host an availability replica for an availability group. It is important to know the role of each clustering technology, and to know what considerations are necessary as you design your AlwaysOn Availability Groups environment.

Note
For information about AlwaysOn Availability Groups concepts, see Overview of AlwaysOn Availability Groups.

In This Topic:

- Windows Server Failover Clustering
- SQL Server Failover Clustering
- Restrictions on Using The WSFC Failover Cluster Manager with Availability Groups

Windows Server Failover Clustering and Availability Groups

Deploying AlwaysOn Availability Groups requires a Windows Server Failover Clustering (WSFC) cluster. To be enabled for AlwaysOn Availability Groups, an instance of SQL Server must reside on a WSFC node, and the WSFC cluster and node must be online. Furthermore, each availability replica of a given availability group must reside on a different node of the same WSFC cluster. The quorum for AlwaysOn Availability Groups is based on all nodes in the WSFC cluster regardless of whether a given cluster node hosts any availability replicas. In contrast to database mirroring, there is no witness role in AlwaysOn Availability Groups.

AlwaysOn Availability Groups relies on the Windows Failover Clustering (WSFC) cluster to monitor and manage the current roles of the availability replicas that belong to a given availability group and to determine how a failover event affects the availability replicas. A WSFC resource group is created for every availability group that you create. The WSFC cluster monitors this resource group to evaluate the health of the primary replica.

The overall health of a WSFC cluster is determined by the votes of a quorum of nodes in the cluster. 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.

Important
If you delete and re-create a WSFC cluster, you must disable and re-enable the AlwaysOn Availability Groups feature on each instance of SQL Server that hosted an availability replica on the original WSFC cluster.

For information about running SQL Server on Windows Server Failover Clustering (WSFC) nodes and about WSFC quorum, see Windows Server Failover Clustering (WSFC) with SQL Server.

SQL Server Failover Cluster Instances (FCIs) and Availability Groups

You can set up a second layer of failover at the server-instance level by implementing SQL Server failover clustering together with the WSFC cluster. An availability replica can be hosted by either a standalone instance of SQL Server or an FCI instance. Only one FCI partner can host a replica for a given availability group. When an availability replica is running on an FCI, the possible owners list for the availability group will contain only the active FCI node.

AlwaysOn Availability Groups does not depend on any form of shared storage. However, if you use a SQL Server failover cluster instance (FCI) to host one or more availability replicas, each of those FCIs will require shared storage as per standard SQL Server failover cluster instance installation.

For more information about additional prerequisites, see the "Prerequisites and Restrictions for Using a SQL Server Failover Cluster Instance (FCI) to Host an Availability Replica" section of Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups (SQL Server).

Comparison of Failover Cluster Instances and Availability Groups

Regardless of the number of nodes in the FCI, an entire FCI hosts a single replica within an availability group. The following table describes the distinctions in concepts between nodes in an FCI and replicas within an availability group.

	Nodes within an FCI	**Replicas within an availability group**
Uses WSFC cluster	Yes	Yes
Protection level	Instance	Database
Storage type	Shared	Non-shared[1]
Storage solutions	Direct attached, SAN, mount points, SMB	Depends on node type
Readable secondaries	No[2]	Yes
Applicable failover policy settings	• WSFC quorum • FCI-specific • Availability group settings[3]	• WSFC quorum • Availability group settings
Failed-over resources	Server, instance, and database	Database only

[1]While the replicas in an availability group do not share storage, a replica that is hosted by an FCI uses a shared storage solution as required by that FCI. The storage solution is shared only by nodes within the FCI and not between replicas of the availability group.

[2]Whereas synchronous secondary replicas in an availability group are always running on their respective SQL Server instances, secondary nodes in an FCI actually have not started their

respective SQL Server instances and are therefore not readable. In an FCI, a secondary node starts its SQL Server instance only when the resource group ownership is transferred to it during an FCI failover. However, on the active FCI node, when an FCI-hosted database belongs to an availability group, if the local availability replica is running as a readable secondary replica, the database is readable.

[3]Failover policy settings for the availability group apply to all replicas, whether it is hosted in a standalone instance or an FCI instance.

Note

For more information about **Number of nodes** within Failover Clustering and **AlwaysOn Availability Groups** for different editions of SQL Server, see Features Supported by the Editions of SQL Server 2012 (http://go.microsoft.com/fwlink/?linkid=232473).

Considerations for hosting an Availability Replica on an FCI

Important

If you plan to host an availability replica on a SQL Server Failover Cluster Instance (FCI), ensure that the Windows Server 2008 host nodes meet the AlwaysOn prerequisites and restrictions for Failover Cluster Instances (FCIs). For more information, see Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups (SQL Server).

SQL Server Failover Cluster Instances (FCIs) do not support automatic failover by availability groups, so any availability replica that is hosted by an FCI can only be configured for manual failover.

You might need to configure a Windows Server Failover Clustering (WSFC) cluster to include shared disks that are not available on all nodes. For example, consider a WSFC cluster across two data centers with three nodes. Two of the nodes host a SQL Server failover clustering instance (FCI) in the primary data center and have access to the same shared disks. The third node hosts a stand-alone instance of SQL Server in a different data center and does not have access to the shared disks from the primary data center. This WSFC cluster configuration supports the deployment of an availability group if the FCI hosts the primary replica and the stand-alone instance hosts the secondary replica.

When choosing an FCI to host an availability replica for a given availability group, ensure that an FCI failover could not potentially cause a single WSFC node to attempt to host two availability replicas for the same availability group.

The following example scenario illustrates how this configuration could lead to problems:

Marcel configures two a WSFC cluster with two nodes, NODE01 and NODE02. He installs a SQL Server failover cluster instance, fciInstance1, on both NODE01 and NODE02 where NODE01 is the current owner for fciInstance1.

On NODE02, Marcel installs another instance of SQL Server, Instance3, which is a stand-alone instance.

On NODE01, Marcel enables fciInstance1 for AlwaysOn Availability Groups. On NODE02, he enables Instance3 for AlwaysOn Availability Groups. Then he sets up an availability group for which fciInstance1 hosts the primary replica, and Instance3 hosts the secondary replica.

At some point fciInstance1 becomes unavailable on NODE01, and the WSFC cluster causes a failover of fciInstance1 to NODE02. After the failover, fciInstance1 is a AlwaysOn Availability Groups-enabled instance running under the primary role on NODE02. However, Instance3 now resides on the same WSFC node as fciInstance1. This violates the AlwaysOn Availability Groups constraint.

To correct the problem that this scenario presents, the stand-alone instance, `Instance3`, must reside on another node in the same WSFC cluster as `NODE01` and `NODE02`.

For more information about SQL Server failover clustering, see AlwaysOn Failover Cluster Instances (FCI).

Restrictions on Using The WSFC Failover Cluster Manager with Availability Groups

Do not use the Failover Cluster Manager to manipulate availability groups, for example:

- Do not change any availability group properties, such as the possible owners.
- Do not use the Failover Cluster Manager to fail over availability groups. You must use Transact-SQL or SQL Server Management Studio.

See Also

Overview (AlwaysOn Availability Groups)

Enabling and Disabling Availability Groups (SQL Server)

Monitoring Availability Groups (Transact-SQL)

AlwaysOn Failover Cluster Instances (FCI)

Getting Started with AlwaysOn Availability Groups

This topic introduces the steps for configuring instances of SQL Server 2012 to support AlwaysOn Availability Groups and for creating, managing, and monitoring an availability group.

- **Before You Begin:**

  Recommended Reading
- **Getting started with:**

  Configuring an instance of SQL Server to support AlwaysOn Availability Groups

  Creating and configuring a new availability group

  Managing availability groups, replicas, and databases

  Monitoring availability groups

- Related Content

Before You Begin

Recommended Reading

Before you create your first availability group, we recommend that you read the following topics:

- Overview of AlwaysOn Availability Groups (SQL Server)
- Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups (SQL Server)

Configuring an Instance of SQL Server to Support AlwaysOn Availability Groups

	Step	Links
☐	**Enable AlwaysOn Availability Groups.** The AlwaysOn Availability Groups feature must be enabled on every instance of SQL Server 2012 that is to participate in an availability group. **Prerequisites:** See "SQL Server Instance Prerequisites and Restrictions" in Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups (SQL Server).	Enable and disable AlwaysOn Availability Groups
☐	**Create database mirroring endpoint (if none).** Ensure that each server instance possesses a database mirroring endpoint. The server instance uses this endpoint to receive AlwaysOn Availability Groups connections from other server instances.	To determine whether database mirroring endpoint exists: • sys.database_mirroring_endpoints **For Windows Authentication:** To create a database mirroring endpoint, using: • New Availability Group Wizard • Transact-SQL • SQL Server PowerShell **For certificate authentication:** To create a database mirroring endpoint, using: • Transact-SQL

Creating and Configuring a New Availability Group

	Step	Links
☐	**Create the availability group.** Create the availability group on the instance of SQL Server that hosts the databases to be added to the availability group. Minimally, create the initial primary replica on the instance of SQL Server where you create the availability group. You can specify from one to four secondary replicas. For information about availability group and replica properties, see CREATE AVAILABILITY GROUP (Transact-SQL). We strongly recommend that you create an availability group listener. **Prerequisites:** See "Availability Group Prerequisites and Restrictions", "Availability Database Prerequisites and Restrictions", and "SQL Server Instance Prerequisites and Restrictions" in Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups (SQL Server).	To create an availability group you can use any of the following tools: • New Availability Group Wizard • Transact-SQL • SQL Server PowerShell
☐	**Join secondary replicas to the availability group.** Connect to each instance of SQL Server 2012 that is hosting a secondary replica, and join the local secondary replica to the availability group.	Join a secondary replica to an availability group **Tip** If you use the New Availability Group Wizard, this step is automated.

	Step	Links
☐	**Prepare secondary databases.** On every server instance that is hosting a secondary replica, restore backups of the primary databases using RESTORE WITH NORECOVERY.	Manually prepare a secondary database **Tip** The New Availability Group Wizard can prepare the secondary databases for you. For more information, see "Prerequisites for using full initial data synchronization" in Select Initial Data Synchronization Page (AlwaysOn Availability Group Wizards).
☐	**Join secondary databases to the availability group.** On every server instance that is hosting a secondary replica, join each local secondary database to the availability group. On joining the availability group, a given secondary database initiates data synchronization with the corresponding primary database.	Join a secondary database to an availability group **Tip** The New Availability Group Wizard can perform this step if every secondary database exists on every secondary replica.
	Create an availability group listener. This step is necessary unless you already created the availability group listener while creating the availability group.	Create or Configure an Availability Group Listener (SQL Server)
☐	**Give the listener's DNS host name to application developers.** Developers needs to specify this DNS name in the connection strings to direct connection requests to the availability group listener. For more information, see Availability Group Listeners, Client	"Follow Up: After Creating an Availability Group Listener" in Create or Configure an Availability Group Listener (SQL Server)

	Step	Links
	Connectivity, and Application Failover (SQL Server).	
☐	**Configure Where Backup Jobs.** If you want to perform backups on secondary databases, you must create a backup job script that takes the automated backup preference into account. Create a script for each database in the availability group on every server instance that hosts an availability replica for the availability group.	"Follow Up: After Configuring Backup on Secondary Replicas" in Configure Backup on Availability Replicas (SQL Server)

Managing Availability Groups, Replicas, and Databases

Note
For information about availability group and replica properties, see CREATE AVAILABILITY GROUP (Transact-SQL).

Managing existing availability groups involves one or more of the following tasks:

Task	Link
Modify the flexible failover policy of the availability group to control the conditions that cause an automatic failover. This policy is relevant only when automatic failover is possible.	Configure the flexible failover policy of an availability group
Perform a planned manual failover or a forced manual failover (with possible data loss). For more information, see Failover and Failover Modes (AlwaysOn Availability Groups).	• Perform a planned manual failover • Perform a forced manual failover
Use a set of predefined policies to view the health of an availability group and its replicas and databases.	• Use policy-based management to view the health of availability groups • Use the AlwaysOn Group Dashboard

Task	Link
Add or remove a secondary replica.	• Add a secondary replica • Remove a secondary replica
Suspend or resume an availability database. Suspending a secondary database keeps at its current point in time until you resume it.	• Suspend a database • Resume a database
Add or remove a database.	• Add a database • Remove a secondary database • Remove a primary database
Reconfigure or create an availability group listener.	Create or configure an availability group listener
Delete an availability group.	Delete an availability group
Troubleshoot add file operations. This might be required if the primary database and a secondary database have different file paths.	Troubleshoot a failed add-file operation
Alter availability replica properties.	• Change the Availability Mode • Change the Failover Mode • Configure Backup Priority (and Automated Backup Preference) • Configure Read-Only Access • Configure Read-Only Routing • Change the Session-Timeout Period

Monitoring Availability Groups

To monitor the properties and state of an AlwaysOn availability group you can use the following tools.

Tool	Brief Description	Links
System Center Monitoring pack for SQL Server	The Monitoring pack for SQL Server (SQLMP) is the recommended solution for	To download the monitoring pack (SQLServerMP.msi) and *SQL Server Management Pack Guide*

Tool	Brief Description	Links
	monitoring availability groups, availability replica and availability databases for IT administrators. Monitoring features that are particularly relevance to AlwaysOn Availability Groups include the following: • Automatic discoverability of availability groups, availability replicas, and availability database from among hundreds of computers. This enables you to easily keep track of your AlwaysOn Availability Groups inventory. • Fully capable System Center Operations Manager (SCOM) alerting and ticketing. These features provide detailed knowledge that enables faster resolution to a problem. • A custom extension to AlwaysOn Health monitoring using Policy Based management (PBM). • Health roll ups from availability databases to availability replicas. • Custom tasks that manage AlwaysOn Availability Groups from the System Center Operations Manager console.	*for System Center Operations Manager* (SQLServerMPGuide.doc), see: System Center Monitoring pack for SQL Server
Transact-SQL	AlwaysOn Availability Groups catalog and dynamic management views provide a wealth of information about	Monitor Availability Groups (Transact-SQL)

Tool	Brief Description	Links
	your availability groups and their replicas, databases, listeners, and WSFC cluster environment.	
SQL Server Management Studio	The **Object Explorer Details** pane displays basic information about the availability groups hosted on the instance of SQL Server to which you are connected. **Tip** Use this pane to select multiple availability groups, replicas, or databases and to perform routine administrative tasks on the selected objects; for example, removing multiple availability replicas or databases from an availability group.	Use Object Explorer Details to monitor availability groups
SQL Server Management Studio	**Properties** dialog boxes enable you to view the properties of availability groups, replicas, or listeners and, in some cases, to change their values.	• Availability Group Properties • Availability Replica Properties • Availability Group Listener Properties
System Monitor	The **SQLServer:Availability Replica** performance object contains performance counters that report information about availability replicas.	SQL Server, HADR Availability Replica
System Monitor	The **SQLServer:Database Replica** performance object contains performance counters that report information about the secondary databases on a	SQL Server, HADR Database Replica SQL Server, Databases Object

Tool	Brief Description	Links
	given secondary replica. The **SQLServer:Databases** object in SQL Server contains performance counters that monitor transaction log activities, among other things. The following counters are particularly relevant for monitoring transaction-log activity on availability databases: **Log Flush Write Time (ms)**, **Log Flushes/sec**, **Log Pool Cache Misses/sec**, **Log Pool Disk Reads/sec**, and **Log Pool Requests/sec**.	

Related Content

- **Video—Introduction to AlwaysOn:** Microsoft SQL Server Code-Named "Denali" AlwaysOn Series,Part 1: Introducing the Next Generation High Availability Solution
- **Video—A Deep Dive into AlwaysOn:** Microsoft SQL Server Code-Named "Denali" AlwaysOn Series,Part 2: Building a Mission-Critical High Availability Solution Using AlwaysOn
- **Whitepaper:** Microsoft SQL Server AlwaysOn Solutions Guide for High Availability and Disaster Recovery
- **Blogs:** SQL Server AlwaysOn Team Blog: The official SQL Server AlwaysOn Team Blog

See Also

AlwaysOn Availability Groups (SQL Server)

Overview of AlwaysOn Availability Groups

Configuration of a Server Instance for AlwaysOn Availability Groups (SQL Server)

Creation and Configuration of Availability Groups (SQL Server)

Availability Group Monitoring (SQL Server)

Overview of Transact-SQL Statements for AlwaysOn Availability Group

Overview of PowerShell Cmdlets for Availability Groups (SQL Server)

Overview of AlwaysOn Availability Groups

This topic introduces the AlwaysOn Availability Groups concepts that are central for configuring and managing one or more availability groups in SQL Server 2012. For a summary of the benefits offered by availability groups and an overview of AlwaysOn Availability Groups terminology, see AlwaysOn Availability Groups (SQL Server).

An *availability group* supports a failover environment for a discrete set of user databases, known as *availability databases*, that fail over together. An availability group supports a set of primary databases and one to four sets of corresponding secondary databases. An availability group fails over at the level of an availability replica. Failovers are not caused by database issues such as a database becoming suspect due to a loss of a data file or corruption of a transaction log.

Each set of availability database is hosted by an *availability replica*. Two types of availability replicas exist: a single *primary replica*. which hosts the primary databases, and one to four *secondary replicas*, each of which hosts a set of secondary databases and serves as a potential failover targets for the availability group. The primary replica makes the primary databases available for read-write connections from clients. Also, in a process known as *data synchronization*, which occurs at the database level. The primary replica sends transaction log records of each primary database to every secondary database. Every secondary replica caches the transaction log records (*hardens* the log) and then applies them to its corresponding secondary database. Data synchronization occurs between the primary database and each connected secondary database, independently of the other databases. Therefore, a secondary database can be suspended or fail without affecting other secondary databases, and a primary database can be suspended or fail without affecting other primary databases.

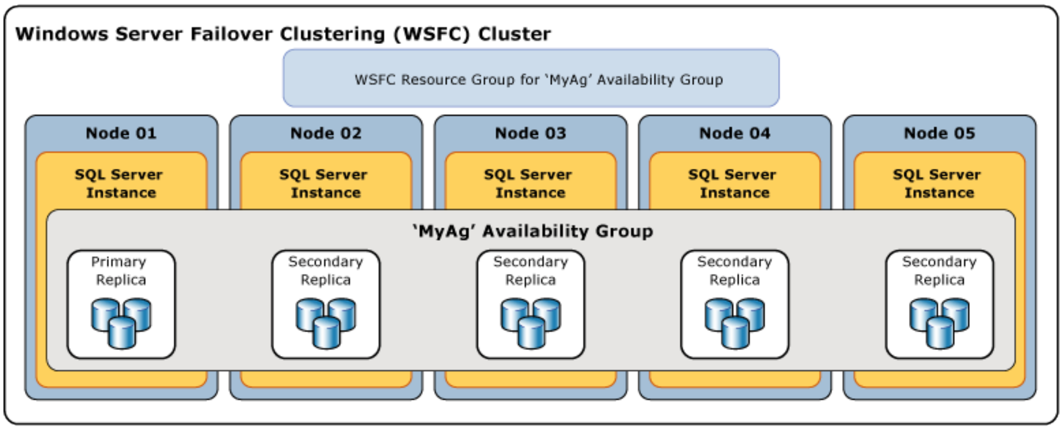
Optionally, you can configure one or more secondary replicas to support read-only access to secondary databases, and you can configure any secondary replica to permit backups on secondary databases.

Deploying AlwaysOn Availability Groups requires a Windows Server Failover Clustering (WSFC) cluster. Each availability replica of a given availability group must reside on a different node of the same WSFC cluster. A WSFC resource group is created for every availability group that you create. The WSFC cluster monitors this resource group to evaluate the health of the primary replica. The quorum for AlwaysOn Availability Groups is based on all nodes in the WSFC cluster regardless of whether a given cluster node hosts any availability replicas. In contrast to database mirroring, there is no witness role in AlwaysOn Availability Groups.

Note

For information about the relationship of SQL Server AlwaysOn components to the WSFC cluster, see Windows Server Failover Clustering (WSFC) with SQL Server.

The following illustration shows an availability group that contains the maximum number of availability replicas, one primary replica and four secondary replicas.



In this Topic:

- Availability Databases
- Availability Replicas
- Availability Modes
- Types of Failover
- Client Connections
- Active Secondary Replicas
- Session-Timeout Period
- Automatic Page Repair
- Related Tasks
- Related Content

Availability Databases

To add a database to an availability group, the database must be an online, read-write database that exists on the server instance that hosts the primary replica. When you add a database, it joins the availability group as a primary database, while remaining available to clients. No corresponding secondary database exists until backups of the new primary database are restored to the server instance that hosts the secondary replica (using RESTORE WITH NORECOVERY). The new secondary database is in the RESTORING state until it is joined to the availability group. For more information, see Start Data Movement on an AlwaysOn Secondary Database (SQL Server).

Joining places the secondary database into the ONLINE state and initiates data synchronization with the corresponding primary database. *Data synchronization* is the process by which changes to a primary database are reproduced on a secondary database. Data synchronization involves the primary database sending transaction log records to the secondary database.

Important

An availability database is sometimes called a *database replica* in Transact-SQL, PowerShell, and SQL Server Management Objects (SMO) names. For example, the term "database replica" is used in the names of the AlwaysOn dynamic management views that return information about availability databases: **sys.dm_hadr_database_replica_states** and **sys.dm_hadr_database_replica_cluster_states**. However, in SQL Server Books Online, the term "replica" typically refers to availability replicas. For example, "primary replica" and "secondary replica" always refer to availability replicas.

Availability Replicas

Each availability group defines a set of two or more failover partners known as availability replicas. *Availability replicas* are components of the availability group that are hosted by separate instances of SQL Server residing on different nodes of a WSFC cluster. Each of these server instances is either a SQL Server failover cluster instance (FCI) or stand-alone instance on which you have enabled AlwaysOn Availability Groups. Each availability replica hosts a copy of the availability databases in the availability group.

Every availability replica is assigned an initial role—either the *primary role* or the *secondary role*, which is inherited by the availability databases of that replica. The role of a given replica determines whether it hosts read-write databases or read-only databases. One replica, known as the *primary replica*, is assigned the primary role and hosts read-write databases, which are known as *primary databases*. At least one other replica, known as a *secondary replica*, is assigned the secondary role. A secondary replica hosts read-only databases, known as secondary databases.

Note

When the role of an availability replica is indeterminate, such as during a failover, its databases are temporarily in a NOT SYNCHRONIZING state. Their role is set to RESOLVING until the role of the availability replica has resolved. If an availability replica resolves to the primary role, its databases become the primary databases. If an availability replica resolves to the secondary role, its databases become secondary databases.

Availability Modes

The availability mode is a property of each availability replica. The availability mode determines whether the primary replica waits to commit transactions on a database until a given secondary replica has written the transaction log records to disk (hardened the log). AlwaysOn Availability Groups supports two availability modes—*asynchronous-commit mode* and *synchronous-commit mode*.

- **Asynchronous-commit mode**

  An availability replica that uses this availability mode is known as an *asynchronous-commit replica*. Under asynchronous-commit mode, the primary replica commits transactions without waiting for acknowledgement that an asynchronous-commit secondary replica has hardened the log. Asynchronous-commit mode minimizes transaction latency on the secondary databases but allows them to lag behind the primary databases, making some data loss possible.

- **Synchronous-commit mode**

  An availability replica that uses this availability mode is known as a *synchronous-commit replica*. Under synchronous-commit mode, before committing transactions, a synchronous-commit primary replica waits for a synchronous-commit secondary replica to acknowledge that it has finished hardening the log. Synchronous-commit mode ensures that once a given secondary database is synchronized with the primary database, committed transactions are fully protected. This protection comes at the cost of increased transaction latency.

For more information, see Availability Modes (AlwaysOn Availability Group).

Types of Failover

Within the context of a session between the primary replica and a secondary replica, the primary and secondary roles are potentially interchangeable in a process known as *failover*. During a failover the target secondary replica transitions to the primary role, becoming the new primary replica. The new primary replica brings its databases online as the primary databases, and client applications can connect to them. When the former primary replica is available, it transitions to the secondary role, becoming a secondary replica. The former primary databases become secondary databases and data synchronization resumes.

Three forms of failover exist—automatic, manual, and forced (with possible data loss). The form or forms of failover supported by a given secondary replica depends on its availability mode, and, for synchronous-commit mode, on the failover mode on the primary replica and target secondary replica, as follows.

- Synchronous-commit mode supports two forms of failover—*planned manual failover* and *automatic failover*, if the target secondary replica is currently synchronized with the avt1. The support for these forms of failover depends on the setting of the *failover mode property* on the failover partners. If failover mode is set to "manual" on either the primary or secondary replica, only manual failover is supported for that secondary replica. If failover mode is set to "automatic" on both the primary and secondary replicas, both automatic and manual failover are supported on that secondary replica.

  - **Planned manual failover** (without data loss)

    A manual failover occurs after a database administrator issues a failover command and causes a synchronized secondary replica to transition to the primary role (with guaranteed data protection) and the primary replica to transition to the secondary role. A manual failover requires that both the primary replica and the target secondary replica are running under synchronous-commit mode, and the secondary replica must already be synchronized.

  - **Automatic failover** (without data loss)

An automatic failover occurs in response to a failure that causes a synchronized secondary replica to transition to the primary role (with guaranteed data protection). When the former primary replica becomes available, it transitions to the secondary role. Automatic failover requires that both the primary replica and the target secondary replica are running under synchronous-commit mode with the failover mode set to "Automatic". In addition, the secondary replica must already be synchronized, have WSFC quorum, and meet the conditions specified by the flexible failover policy of the availability group.

Important
SQL Server Failover Cluster Instances (FCIs) do not support automatic failover by availability groups, so any availability replica that is hosted by an FCI can only be configured for manual failover.

- Under asynchronous-commit mode, the only form of failover is forced manual failover (with possible data loss), known as *forced failover*. Forced failover can only be initiated manually, and for this reason, is considered to be a kind of manual failover. However, forced failover is a disaster recovery option that is supported only when the secondary replica is not synchronized with the primary replica.   Forced failover is the only form of failover that is possible when the target secondary replica is not synchronized with the primary replica, even for synchronous-commit mode.

  Note
  If you issue a forced failover command on a synchronized secondary replica, the secondary replica behaves the same as for a manual failover.

For more information, see Failover Modes (AlwaysOn Availability Groups).

Client Connections

You can provide client connectivity to the primary replica of a given availability group by creating an availability group listener. An *availability group listener* provides a set of resources that is attached to a given availability group to direct client connections to the appropriate availability replica.

An availability group listener is associated with a unique DNS name that serves as a virtual network name (VNN), one or more virtual IP addresses (VIPs), and a TCP port number. For more information, see Client Connectivity and Application Failover (AlwaysOn Availability Groups).

Tip
If an availability group possesses only two availability replicas and is not configured to allow read-access to the secondary replica, clients can connect to the primary replica by using a database mirroring connection string. This approach can be useful temporarily after you migrate a database from database mirroring to AlwaysOn Availability Groups. Before you add additional secondary replicas, you will need to create an availability

group listener the availability group and update your applications to use the network name of the listener.

Active Secondary Replicas

AlwaysOn Availability Groups supports active secondary replicas. Active secondary capabilities include support for:

- **Performing backup operations on secondary replicas**

  The secondary replicas support performing log backups and copy-only backups of a full database, file, or filegroup. You can configure the availability group to specify a preference for where backups should be performed. It is important to understand that the preference is not enforced by SQL Server, so it has no impact on ad-hoc backups. The interpretation of this preference depends on the logic, if any, that you script into your back jobs for each of the databases in a given availability group. For an individual availability replica, you can specify your priority for performing backups on this replica relative to the other replicas in the same availability group. For more information, see Backup on Secondary Replicas (AlwaysOn Availability Groups).

- **Read-only access to one or more secondary replicas (readable secondary replicas)**

  Any availability replica can be configured to allow read-only access to its local databases when performing the secondary role, though some operations are not fully supported. Also, if you would like to prevent read-only workloads from running on the primary replica, you can configure the replicas to allow only read-write access when running under the primary role. For more information, see Readable Secondary Replicas (AlwaysOn Availability Groups).

  If an availability group currently possesses an availability group listener and one or more readable secondary replicas, SQL Server can route read-intent connection requests to one of them (*read-only routing*). For more information, see Availability Group Listeners, Client Connectivity, and Application Failover (SQL Server).

Session-Timeout Period

The session-timeout period is an availability-replica property that determines how long connection with another availability replica can remain inactive before the connection is closed. The primary and secondary replicas ping each other to signal that they are still active. Receiving a ping from the other replica during the timeout period indicates that the connection is still open and that the server instances are communicating. On receiving a ping, an availability replica resets its session-timeout counter on that connection.

The session-timeout period prevents either replica from waiting indefinitely to receive a ping from the other replica. If no ping is received from the other replica within the session-timeout period, the replica times out. Its connection is closed, and the timed-out replica enters the DISCONNECTED state. Even if a disconnected replica is configured for synchronous-commit mode, transactions will not wait for that replica to reconnect and resynchronize.

The default session-timeout period for each availability replica is 10 seconds. This value is user-configurable, with a minimum of 5 seconds. Generally, we recommend that you keep the time-out period at 10 seconds or greater. Setting the value to less than 10 seconds creates the possibility of a heavily loaded system declaring a false failure.

Note

In the resolving role, the session-timeout period does not apply because pinging does not occur.

Automatic Page Repair

Each availability replica tries to automatically recover from corrupted pages on a local database by resolving certain types of errors that prevent reading a data page. If a secondary replica cannot read a page, the replica requests a fresh copy of the page from the primary replica. If the primary replica cannot read a page, the replica broadcasts a request for a fresh copy to all the secondary replicas and gets the page from the first to respond. If this request succeeds, the unreadable page is replaced by the copy, which usually resolves the error.

For more information, see Automatic Page Repair (Availability Groups/Database Mirroring).

Related Tasks

- Getting Started with AlwaysOn Availability Groups (SQL Server)

Related Content

- **Video—Introduction to AlwaysOn:** Microsoft SQL Server Code-Named "Denali" AlwaysOn Series,Part 1: Introducing the Next Generation High Availability Solution
- **Video—A Deep Dive into AlwaysOn:** Microsoft SQL Server Code-Named "Denali" AlwaysOn Series,Part 2: Building a Mission-Critical High Availability Solution Using AlwaysOn
- **Whitepaper:** Microsoft SQL Server AlwaysOn Solutions Guide for High Availability and Disaster Recovery
- **Blogs:** SQL Server AlwaysOn Team Blog: The official SQL Server AlwaysOn Team Blog

See Also

Availability Modes (AlwaysOn Availability Group)

Failover Modes (AlwaysOn Availability Group)

Overview of Transact-SQL Statements

Overview of PowerShell Cmdlets for AlwaysOn Availability Group (SQL Server)

Considerations for Deploying AlwaysOn Availability Groups (SQL Server)

Creating and Configuring an Availability Group (AlwaysOn Availability Groups)

Read-Only Access to Secondary Replicas

[Backup on Secondary Replicas (AlwaysOn Availability Groups)]()

[Availability Group Listeners, Client Connectivity, and Application Failover (SQL Server)]()

Availability Modes (AlwaysOn Availability Groups)

In AlwaysOn Availability Groups, the *availability mode* is a replica property that determines whether a given availability replica can run in synchronous-commit mode. For each availability replica, the availability mode must be configured for either synchronous-commit mode or asynchronous-commit mode. If the primary replica is configured for *asynchronous-commit mode*, it does not wait for any secondary replica to write incoming transaction log records to disk (to *harden the log*). If a given secondary replica is configured for asynchronous-commit mode, the primary replica does not wait for that secondary replica to harden the log. If both the primary replica and a given secondary replica are both configured for *synchronous-commit mode*, the primary replica waits for the secondary replica to confirm that it has hardened the log (unless the secondary replica fails to ping the primary replica within the primary's *session-timeout period*).

Note

If primary's session-timeout period is exceeded by a secondary replica, the primary replica temporarily shifts into asynchronous-commit mode for that secondary replica. When the secondary replica reconnects with the primary replica, they resume synchronous-commit mode.

In this Topic:

- Supported Availability Modes
- Asynchronous-Commit Availability Mode
- Synchronous-Commit Availability Mode
- Related Tasks
- Related Content

Supported Availability Modes

AlwaysOn Availability Groups supports two availability modes—asynchronous-commit mode and synchronous-commit mode, as follows:

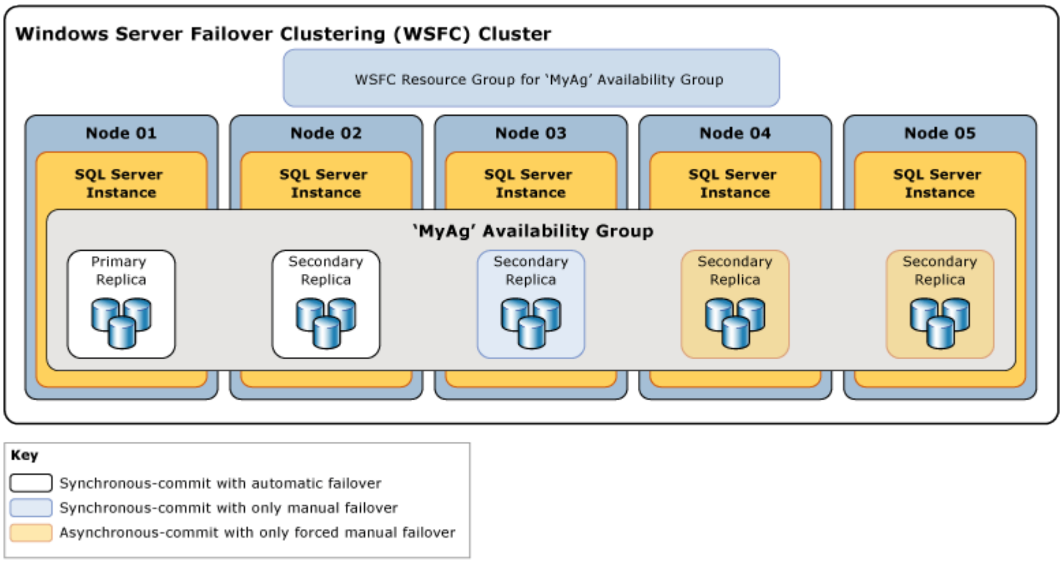
- *Asynchronous-commit mode* is a disaster-recovery solution that works well when the availability replicas are distributed over considerable distances. If every secondary replica is running under asynchronous-commit mode, the primary replica does not wait for any of the secondary replicas to harden the log. Rather, immediately after writing the log record to the local log file, the primary replica sends the transaction confirmation to the client. The primary replica runs with minimum transaction latency in relation to a secondary replica that is configured for asynchronous-commit mode. If the current primary is configured for asynchronous commit availability mode, it will commit transactions asynchronously for all secondary replicas regardless of their individual availability mode settings.

  For more information, see Asynchronous-Commit Availability Mode, later in this topic.

- *Synchronous-commit mode* emphasizes high availability over performance, at the cost of increased transaction latency. Under synchronous-commit mode, transactions wait to send the transaction confirmation to the client until the secondary replica has hardened the log to disk. When data synchronization begins on a secondary database, the secondary replica begins applying incoming log records from the corresponding primary database. As soon as every log record has been hardened, the secondary database enters the SYNCHRONIZED state. Thereafter, every new transaction is hardened by the secondary replica before the log record is written to the local log file. When all the secondary databases of a given secondary replica are synchronized, synchronous-commit mode supports manual failover and, optionally, automatic failover.

  For more information, see Synchronous-Commit Availability Mode, later in this topic.

The following illustration shows an availability group with five availability replicas. The primary replica and one secondary replica are configured for synchronous-commit mode with automatic failover. Another secondary replica is configured for synchronous-commit mode with only manual failover, and two secondary replicas are configured for asynchronous-commit mode, which supports only forced manual failover.


Windows Server Failover Clustering (WSFC) Cluster
WSFC Resource Group for 'MyAg' Availability Group
Node 01
Node 02
Node 03
Node 04
Node 05
SQL Server Instance
SQL Server Instance
SQL Server Instance
SQL Server Instance
SQL Server Instance
'MyAg' Availability Group
Primary Replica
Secondary Replica
Secondary Replica
Secondary Replica
Secondary Replica
Key
Synchronous-commit with automatic failover
Synchronous-commit with only manual failover
Asynchronous-commit with only forced manual failover

Asynchronous-Commit Availability Mode

Under *asynchronous-commit mode*, the secondary replica never becomes synchronized with the primary replica. Though a given secondary database might catch up to the corresponding primary database, any secondary database could lag behind at any point. Asynchronous-commit mode can be useful in a disaster-recovery scenario in which the primary replica and the secondary replica are separated by a significant distance and where you do not want small errors to impact the primary replica or in or situations where performance is more important than synchronized data protection. Furthermore, since the primary replica does not wait for acknowledgements from the secondary replica, problems on the secondary replica never impact the primary replica.

An asynchronous-commit secondary replica attempts to keep up with the log records received from the primary replica. But asynchronous-commit secondary databases always remain unsynchronized and are likely to lag somewhat behind the corresponding primary databases. Typically the gap between a asynchronous-commit secondary database and the corresponding primary datagase is small. But the gap can become substantial if the server hosting the secondary replica is over loaded or the network is slow.

The only form of failover supported by asynchronous-commit mode is forced failover (with possible data loss). Forcing failover is a last resort intended only for situations in which the current primary replica will remain unavailable for an extended period and immediate availability of primary databases is more critical than the risk of possible data loss. The target secondary replica transitions to the primary role, and its copies of the databases become the primary database. Any remaining secondary databases, along with the former primary databases, once they become available, are suspended until you manually resume them individually. Under asynchronous-commit mode, any transaction logs that the original primary replica had not yet sent to the former secondary replica are lost. This means that some or all of the new primary databases might be lacking recently committed transactions. For more information on how forced failover works and on best practices for using it, see Failover Modes (AlwaysOn Availability Groups).

Synchronous-Commit Availability Mode

Under synchronous-commit availability mode (*synchronous-commit mode*), after being joined to an availability group, a secondary database catches up to the corresponding primary database and enters the SYNCHRONIZED state. The secondary database remains SYNCHRONIZED as long as data synchronization continues. This guarantees that every transaction that is committed on a given primary database has also been committed on the corresponding secondary database. When every secondary database on a given secondary replica is synchronized, the synchronization-health state of the secondary replica as a whole is HEALTHY.

In This Section:

- Factors That Disrupt Data Synchronization
- How Synchronization Works on a Secondary Replica
- Synchronous-Commit Mode with Only Manual Failover
- Synchronous-Commit Mode with Automatic Failover

Factors That Disrupt Data Synchronization

Once all of its databases are synchronized, a secondary replica enters the HEALTHY state. The synchronized secondary replica will remain healthy unless one of the following occurs:

- A network or computer delay or glitch causes the session between the secondary replica and primary replica to timeout.

Note

For information about the session-time property of availability replicas, see Overview of AlwaysOn Availability Groups.

- You suspend a secondary database on the secondary replica. The secondary replica ceases to be synchronized, and its synchronization-health state is marked as NOT_HEALTHY. the secondary replica cannot become healthy again until the suspended secondary database is either resumed and resynchronized or removed from the availability group.
- You add a primary database the availability group. Previously synchronized secondary replicas enter the NOT_HEALTHY synchronization-health state. This state indicates that at least one database is in the NOT SYNCHRONIZING synchronization state. A given secondary replica cannot be HEALTHY again until a corresponding secondary database has been prepared on the replica, has been joined to the availability group, and has become synchronized with the new primary database.
- You change the primary replica or the secondary replica to asynchronous-commit availability mode. After changing to asynchronous-commit mode, the secondary replica will remain in the HEALTHY synchronization-health state as long as data synchronization continues. However, if only the primary replica is changed to asynchronous-commit mode, the synchronous-commit secondary replica will enter the PARTIALLY_HEALTHY synchronization-health state. This state indicates that at least one database is in the SYNCHRONIZING synchronization state, but none of the databases are in the NOT SYNCHRONIZING state.
- You change any secondary replica to synchronous-commit availability mode. This causes that secondary replica to be marked as in the PARTIALLY_HEALTHY synchronization-health state. until all of its databases are in the SYNCHRONIZED synchronization state.

Tip
To view the synchronization health of an availability group, availability replica, or availability database, query the **synchronization_health** or **synchronization_health_desc** column of sys.dm_hadr_availability_group_states, sys.dm_hadr_availability_replica_states, or sys.dm_hadr_database_replica_states, respectively.

How Synchronization Works on a Secondary Replica

Under the synchronous-commit mode, after a secondary replica joins the availability group and establishes a session with the primary replica, the secondary replica writes incoming log records to disk (*hardens the log*) and sends a confirmation message to the primary replica. Once the hardened log on the secondary database has caught up the end of log on the primary database, the state of the secondary database is set to SYNCHRONIZED. The time required for synchronization depends essentially on how far the secondary database was behind the primary database at the start of the session (measured by the number of log records initially received from the primary replica), the work load on the primary database, and the speed of the computer of the server instance that hosts the secondary replica.

Synchronous operation is maintained in the following manner:

1. On receiving a transaction from a client, the primary replica writes the log for the transaction to the transaction log and concurrently sends the log record to the secondary replicas.
2. Once a log record is written to the transaction log of the primary database, the transaction can be undone only if there is a failover at this point to a secondary that did not receive the log. The primary replica waits for confirmation from the synchronous-commit secondary replica.
3. The secondary replica hardens the log and returns an acknowledgement to the primary replica.
4. On receiving the confirmation from the secondary replica, the primary replica finishes the commit processing and sends a confirmation message to the client.

   Note
   If a synchronous-commit secondary replica times out without confirming that it has hardened the log, the primary marks that secondary replica as failed. The connected state of the secondary replica changes to DISCONNECTED, and the primary replica stops waiting for confirmation from the secondary replica. This behavior ensures that a failed synchronous-commit secondary replica does not prevent hardening of the transaction log on the primary replica.

Synchronous-commit mode protects your data by requiring the data to be synchronized between two places, at the cost of somewhat increasing the latency of the transaction.

Synchronous-Commit Mode with Only Manual Failover

When these replicas are connected and the database is synchronized, manual failover is supported. If the secondary replica goes down, the primary replica is unaffected. The primary replica runs exposed if no SYNCHRONIZED replicas exist (that is, without sending data to any secondary replica). If the primary replica is lost, the secondary replicas enter the RESOLVING state, but the database owner can force a failover to the secondary replica (with possible data loss). For more information, see Failover Modes (AlwaysOn Availability Groups).

Synchronous-Commit Mode with Automatic Failover

Automatic failover provides high availability by ensuring that the database is quickly made available again after the loss of the primary replica. To configure an availability group for automatic failover, you need to set both the current primary replica and one secondary replica to synchronous-commit mode with automatic failover.

Furthermore, for an automatic failover to be possible at a given time, this secondary replica must be synchronized with the primary replica (that is, the secondary databases are all synchronized), and the Windows Server Failover Clustering (WSFC) cluster must have quorum. If the primary replica becomes unavailable under these conditions, automatic failover occurs. The secondary replica switches to the role of primary, and it offers its database as the primary database. For more information, see the "Automatic Failover " section of the Failover Modes (AlwaysOn Availability Groups) topic.

Note

For information about WSFC quorum and AlwaysOn Availability Groups, see For more information, see WSFC Quorum Modes and Voting Configuration (SQL Server).

Related Tasks

To change the availability mode and failover mode

- Set the Availability Mode of an Availability Replica (SQL Server)
- Set the Failover Mode of an Availability Replica (SQL Server)

To adjust quorum votes

- View Cluster Quorum NodeWeight Settings
- Configure Cluster Quorum NodeWeight Settings
- Force a WSFC Cluster to Start Without a Quorum

To perform a manual failover

- Perform a Planned Manual Fail Over of an Availability Group (SQL Server)
- Perform a Forced Manual Failover of an Availability Group (SQL Server)
- Use the Fail Over Availability Group Wizard (SQL Server Management Studio)

To view availability group, availability replica, and database states

- sys.dm_hadr_availability_group_states
- sys.dm_hadr_availability_replica_states
- sys.dm_hadr_database_replica_states

Related Content

- Microsoft SQL Server AlwaysOn Solutions Guide for High Availability and Disaster Recovery
- SQL Server AlwaysOn Team Blog: The official SQL Server AlwaysOn Team Blog

See Also

AlwaysOn Availability Groups (SQL Server)

Failover Modes (AlwaysOn Availability Groups)

Windows Server Failover Clusters (WSFC) with SQL Server

Change the Availability Mode of an Availability Replica

This topic describes how to change the availability mode of an availability replica in an AlwaysOn availability group in SQL Server 2012 by using SQL Server Management Studio, Transact-SQL, or PowerShell. The availability mode is a replica property that controls the whether the replica commits asynchronously or synchronously. *Asynchronous-commit mode* maximizes performance at the expense of high availability and supports forced failover (with possible data loss). This *Synchronous-commit mode* emphasizes high availability over performance and, once the secondary replica is synchronized, supports manual failover and, optionally, automatic failover. This replica property applies only when an availability replica is performing the secondary role.

- **Before you begin:**

  Prerequisites

  Security

- **To change the availability mode of an availability replica, using:**

  SQL Server Management Studio

  Transact-SQL

  PowerShell

Before You Begin

Prerequisites

- You must be connected to the server instance that hosts the primary replica.

Security

Permissions

Requires ALTER AVAILABILITY GROUP permission on the availability group, CONTROL AVAILABILITY GROUP permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.

Using SQL Server Management Studio

To change the availability mode of an availability group

1. In Object Explorer, connect to the server instance that hosts the primary replica, and expand the server tree.
2. Expand the **AlwaysOn High Availability** node and the **Availability Groups** node.
3. Click the availability group whose replica you want to change.
4. Right-click the replica, and click **Properties**.
5. In the **Availability Replica Properties** dialog box, use the **Availability mode** drop list to change the availability mode of this replica.

Using Transact-SQL

To change the availability mode of an availability group

1. Connect to the server instance that hosts the primary replica.
2. Use the ALTER AVAILABILITY GROUP statement, as follows:

   ALTER AVAILABILITY GROUP *group_name* MODIFY REPLICA ON 'server_name'

   ```
     WITH ( {
        AVAILABILITY_MODE = { SYNCHRONOUS_COMMIT | ASYNCHRONOUS_COMMIT }
      | FAILOVER_MODE = { AUTOMATIC | MANUAL }
   ```

```
    })
```

where group_name is the name of the availability group and server_name is the name of the server instance that hosts the replica to be modified.

Note
FAILOVER_MODE = AUTOMATIC is supported only if you also specify AVAILABILITY_MODE = SYNCHRONOUS_COMMIT.

The following example, entered on the primary replica of the `AccountsAG` availability group, changes the availability and failover modes to synchronous commit and automatic failover, respectively, for the replica hosted by the `INSTANCE09` server instance.

```
ALTER AVAILABILITY GROUP AccountsAG MODIFY REPLICA ON 'INSTANCE09'
   WITH (AVAILABILITY_MODE = SYNCHRONOUS_COMMIT);
ALTER AVAILABILITY GROUP AccountsAG MODIFY REPLICA ON 'INSTANCE09'
   WITH (FAILOVER_MODE = AUTOMATIC);
```

Using PowerShell

To change the availability mode of an availability group

1. Change directory (**cd**) to the server instance that hosts the primary replica.
2. Use the **Set-SqlAvailabilityReplica** cmdlet with the **AvailabilityMode** parameter and, optionally, the **FailoverMode** parameter.

   For example, the following command modifies the replica `MyReplica` in the availability group `MyAg` to use synchronous-commit availability mode and to support automatic failover.

   ```
   Set-SqlAvailabilityReplica -AvailabilityMode "SynchronousCommit" -
   FailoverMode "Automatic" `
   -Path
   SQLSERVER:\Sql\PrimaryServer\InstanceName\AvailabilityGroups\MyAg\Repl
   icas\MyReplica
   ```

Note
To view the syntax of a cmdlet, use the **Get-Help** cmdlet in the SQL Server PowerShell environment. For more information, see Get Help SQL Server PowerShell.

To set up and use the SQL Server PowerShell provider

- SQL Server PowerShell Provider

See Also

Overview of AlwaysOn Availability Groups

Availability Modes (AlwaysOn Availability Groups)

Failover Modes (AlwaysOn Availability Groups)

Failover and Failover Modes (AlwaysOn Availability Groups)

Within the context of an availability group, the primary role and secondary role of availability replicas are typically interchangeable in a process known as *failover*. Three forms of failover exist: automatic failover (without data loss), planned manual failover (without data loss), and forced manual failover (with possible data loss). Automatic and planned manual failover preserve all your data. An availability group fails over at the availability-replica level. That is, an availability group fails over to one of its secondary replicas (the current *failover target*).

Note

Issues at the database level, such as a database becoming suspect due to the loss of a data file, deletion of a database, or corruption of a transaction log, do not cause an availability group to failover.

During the failover, the failover target takes over the primary role, recovers its databases, and brings them online as the new primary databases. The former primary replica, when available, switches to the secondary role, and its databases become secondary databases. Potentially, these roles can switch back and forth (or to a different failover target) in response to multiple failures or for administrative purposes.

The form(s) of failover that a given availability replica supports is specified by the *failover mode* property. For a given availability replica, the possible failover modes depends on the availability mode of the replica, as follows:

- **Synchronous-commit replicas** support two settings—automatic or manual. The "automatic" setting supports both automatic failover and manual failover. To prevent data loss, automatic failover and planned failover require that the failover target be a synchronous-commit secondary replica with a healthy synchronization state (this indicates that every secondary database on the failover target is synchronized with its corresponding primary database). Whenever an secondary replica does not meet both of these conditions, it supports only forced manual failover.
- **Asynchronous-commit replicas** support only the manual failover mode. Moreover, because they are never synchronized, they support only forced manual failover— often referred to as simply "forced failover".

Note

After a failover, client applications that need to access the primary databases must connect to the new primary replica. Also, if the new secondary replica is configured to allow read-only access, read-only client applications can connect to it. For information about how clients connect to an availability group, see Availability Group Listeners, Client Connectivity, and Application Failover.

Sections in This Topic:

- Terms and Definitions

- Overview of Failover
- Automatic Failover
- Planned Manual Failover (Without Data Loss)
- Forced Manual Failover (with Possible Data Loss)
- Related Tasks
- Related Content

Terms and Definitions

Automatic failover

A failover that occurs automatically on the loss of the primary replica. Automatic failover is supported only when the current primary and one secondary replica are both configured with failover mode set to AUTOMATIC and the secondary replica currently synchronized. If the failover mode of either the primary or secondary replica is MANUAL, automatic failover cannot occur.

Planned manual failover (without data loss)

Planned manual failover, or *manual failover*, is a failover that is initiated by a database administrator, typically, for administrative purposes. A planned manual failover is supported only if both the primary replica and secondary replica are configured for synchronous-commit mode and the secondary replica is currently synchronized (in the SYNCHRONIZED state). When the target secondary replica is synchronized, manual failover (without data loss) is possible even if the primary replica has crashed because the secondary databases are ready for failover. A database administrator manually initiates a manual failover.

Forced manual failover (with possible data loss)

A failover that can be initiated by a database administrator when a planned manual failover is not possible, because either no secondary replica is SYNCHRONIZED with the primary replica (that is, no secondary replica is ready for failover) or the primary replica is not running. Forced manual failover, or *forced failover*, risks possible data loss and is recommended strictly for disaster recovery. This is the only form of failover supported by in asynchronous-commit availability mode.

Automatic failover set

Within a given availability group, a pair of availability replicas (including the current primary replica) that are configured for synchronous-commit mode with automatic failover, if any. An automatic failover set takes effect only if the secondary replica is currently SYNCHRONIZED with the primary replica.

Synchronous-commit failover set

Within a given availability group, a set of two or three availability replicas (including the current primary replica) that are configured for synchronous-commit mode. A synchronous-commit failover set takes effect only if the secondary replicas are configured for manual failover mode and at least one secondary replica is currently SYNCHRONIZED with the primary replica.

Entire failover set

Within a given availability group, the set of all availability replicas whose operational state is currently ONLINE, regardless of availability mode and of failover mode. The entire failover set becomes relevant when no secondary replica is currently SYNCHRONIZED with the primary replica.

Overview of Failover

The following table summarizes which forms of failover are supported under different availability and failover modes. For each pairing, the effective availability mode and failover mode is determined by the intersection of the modes of the primary replica plus the modes of one or more secondary replicas.

	Asynchronous-commit mode	**Synchronous-commit mode with manual-failover mode**	**Synchronous-commit mode with automatic-failover mode**
Automatic failover	No	No	Yes
Manual failover	No	Yes	Yes
Forced failover	Yes	Yes	Yes*

* If you issue a forced failover command on a synchronized secondary replica, the secondary replica behaves the same as for a manual failover.

The amount of time that the database will be unavailable during a failover depends on the type of failover and its cause.

Important
To support client connections after failover, except for contained databases, logins and jobs defined on any of the former primary databases must be manually recreated on the new primary database. For more information, see Management of Logins and Jobs After an Availability Group Fails Over (SQL Server).

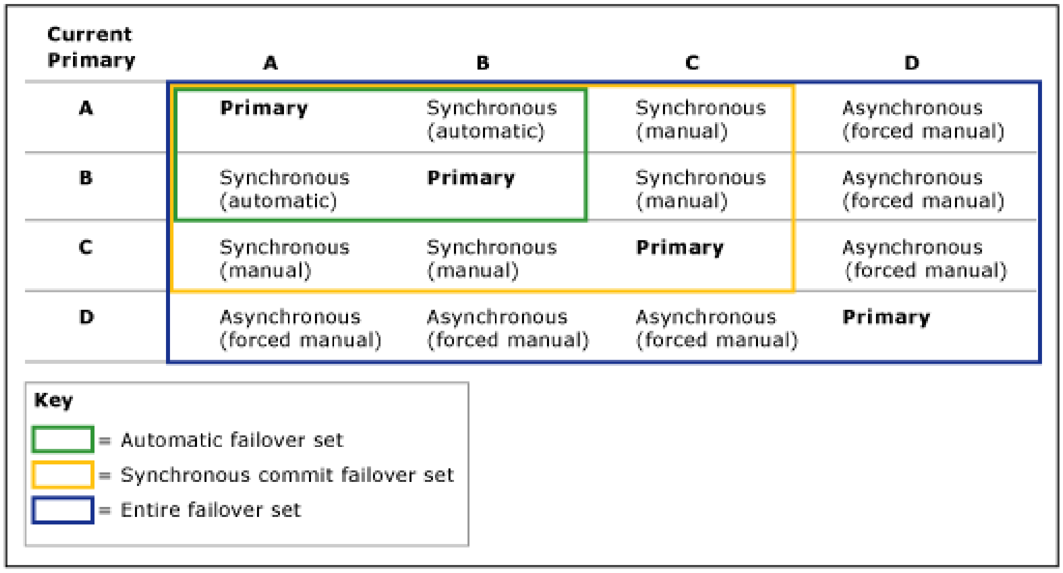
Failover Sets

When you configure an availability replica as synchronous commit with automatic failover, the availability replica becomes part of the automatic failover set. However whether the set takes effect depends the current primary. The forms of failover that are actually possible at a given time depends on what failover sets are currently in effect.

For example, consider an availability group that has four availability replicas, as follows:

Replica	Availability Mode and Failover Mode Settings
A	Synchronous commit with automatic failover
B	Synchronous commit with automatic failover
C	Synchronous commit with manual failover only
D	Asynchronous commit (with only forced manual failover)

The failover behavior for each secondary replica depends on which availability replica is currently the primary replica. Basically, for a given secondary replica, the failover behavior is the worst case given the current primary replica. The following figure illustrates how the failover behavior of secondary replicas varies depending on the current primary replica, and whether it is configured for asynchronous-commit mode (with only forced manual failover) or synchronous-commit mode (with or without automatic failover).



Current Primary	A	B	C	D
A	**Primary**	Synchronous (automatic)	Synchronous (manual)	Asynchronous (forced manual)
B	Synchronous (automatic)	**Primary**	Synchronous (manual)	Asynchronous (forced manual)
C	Synchronous (manual)	Synchronous (manual)	**Primary**	Asynchronous (forced manual)
D	Asynchronous (forced manual)	Asynchronous (forced manual)	Asynchronous (forced manual)	**Primary**

Key

🟩 = Automatic failover set

🟨 = Synchronous commit failover set

🟦 = Entire failover set

Automatic Failover

An automatic failover causes a qualified secondary replica to automatically transition to the primary role after the primary replica becomes unavailable. Automatic failover is best suited when the WSFC node that hosts the primary replica is local to the node that hosts the secondary replica. This is because data synchronization works best with low message latency between computers and because client connections can remain local.

In This Section:

- Conditions Required for an Automatic Failover
- How Automatic Failover Works
- To Enable Automatic Failover

Conditions Required for an Automatic Failover

Automatic failover occurs only under the following conditions:

- Both the primary replica and a secondary replica (the *automatic failover target*) are configured for synchronous-commit mode and are both set to AUTOMATIC failover. If either the primary or secondary replica is set MANUAL failover, automatic failover cannot occur.
- The automatic failover target has a healthy synchronization state (this indicates that every secondary database on the failover target is synchronized with its corresponding primary database).

  For more information, see Availability Modes (AlwaysOn Availability Groups).
- The Windows Server Failover Clustering (WSFC) cluster has quorum. For more information, see WSFC Quorum Modes and Voting Configuration (SQL Server).
- The primary replica has become unavailable, and the failover-condition levels defined by your the flexible failover policy have been met. For information about failover-condition levels, see Flexible Failover Policy for Automatic Failover of an Availability Group (SQL Server).

How Automatic Failover Works

An automatic failover initiates the following sequence of actions:

1. If the server instance that is hosting the current primary replica is still running, it changes the state of the primary databases to DISCONNECTED and disconnects all clients.
2. If any log records are waiting in recovery queues on the target secondary replica, the secondary replica applies the remaining log records to finish rolling forward the secondary databases.

   Note
   The amount of time required to apply the log to a given database depends on the speed of the system, the recent work load, and the amount of log in the recovery queue.
3. The former secondary replica transitions to the primary role. Its databases become the primary databases. The new primary replica rolls back any uncommitted transactions (the undo phase of recovery) as quickly as possible. Locks isolate these uncommitted transactions, allowing roll back to occur in the background while clients use the database. This process does not roll back any committed transactions.

   Until a given secondary database is connected, it is briefly marked as NOT_SYNCHRONIZED. Before the rollback recovery starts, secondary databases can connect to the new primary databases and quickly transition to the SYNCHRONIZED state. The best case is usually for a third synchronous-commit replica that remains in the secondary role after the failover.

4. Later, when the server instance that is hosting the former primary replica restarts, it recognizes that another availability replica now owns the primary role. The former primary replica transitions to the secondary role, and its databases become secondary databases. The new secondary replica connects to the current primary replica and catches its database up to the current primary databases as quickly as possible. As soon as the new secondary replica has resynchronized its databases, failover is again possible, in the reverse direction.

To Configure Automatic Failover

An availability replica can be configured to support automatic failover at any point.

To configure automatic failover

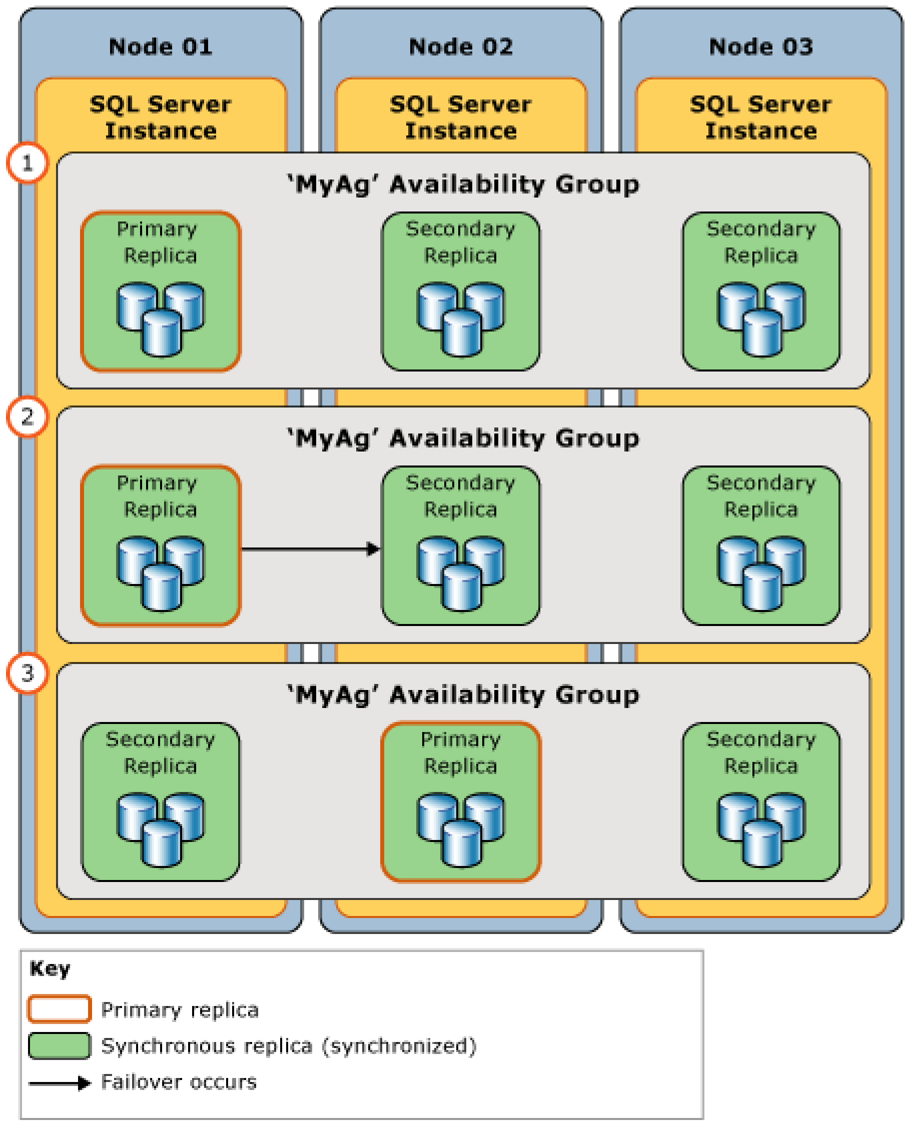
1. Ensure that the secondary replica is configured to use the synchronous-commit availability mode. For more information, see Set the Availability Mode of an Availability Replica (SQL Server).
2. Set the failover mode to automatic. For more information, see Set the Failover Mode of an Availability Replica (SQL Server).
3. Optionally, change the flexible failover policy of the availability group to specify the sorts of failures that can cause an automatic failover to occur. For more information, see Configure the Flexible Failover Policy to Control Conditions for Automatic Failover (AlwaysOn Availability Groups) and Failover Policy for Failover Cluster Instances.

Planned Manual Failover (Without Data Loss)

A manual failover causes a synchronized secondary replica to transition to the primary role after a database administrator issues a manual-failover command on the server instance that hosts the target secondary replica. To support manual failover, the secondary replica and the current primary replica must both be configured for synchronous-commit mode. Every secondary database on the availability replica must be joined to the availability group and synchronized with its corresponding primary database (that is, the secondary replica must be synchronized). This guarantees that every transaction that was committed on a former primary database has also been committed on the new primary database. Therefore, the new primary databases are identical to the old primary databases.

The following figure illustrates the stages of a planned failover:

1. Before the failover, the primary replica is hosted by the server instance on `Node01`.
2. A database administrator initiates a planned failover. The failover target is the availability replica hosted by the server instance on `Node02`.
3. The failover target (on `Node02`) becomes the new primary replica. Because this is a planned failover, the former primary replica switches to the secondary role during the failover and brings its databases online as secondary databases immediately.



In This Section:

- Conditions Required for a Manual Failover
- How Manual Failover Works
- Maintaining Availability During Upgrades

Conditions Required for a Manual Failover

To support a manual failover, the current primary replica must be set to synchronous-commit mode and a secondary replica must be:

- Configured for synchronous-commit mode.
- Currently synchronized with the primary replica.

To manually fail over an availability group, you must be connected to the secondary replica that is to become the new primary replica.

How a Planned Manual Failover Works

A planned manual failover, which must be initiated on the target secondary replica, initiates the following sequence of actions:

1. To ensure that no new user transactions occur on the original primary databases, the WSFC cluster sends a request to the primary replica to go offline.
2. If any log is waiting in the recovery queue of any secondary database, the secondary replica finishes rolling forward that secondary database. The amount of time required depends on the speed of the system, the recent workload, and the amount of log in the recovery queue. To learn the current size of the recovery queue, use the **Recovery Queue** performance counter. For more information, see SQL Server, HADR Database Replica.

   Note
   The failover time can be regulated by limiting the size of the recovery queue. However, this can cause the primary replica to slow down to allow the secondary replica to keep up.
3. The secondary replica becomes the new primary replica, and the former primary replica becomes the new secondary replica.
4. The new primary replica rolls back any uncommitted transactions and brings its databases online as the primary databases. All secondary databases are briefly marked as NOT SYNCHRONIZED until they connect and resynchronize to the new primary databases. This process does not roll back any committed transactions.
5. When the former primary replica comes back online, it takes on the secondary role, and the former primary database becomes the secondary database. The new secondary replica quickly resynchronizes the new secondary databases with the corresponding primary databases.

   Note
   As soon as the new secondary replica has resynchronized the databases, failover is again possible, but in the reverse direction.

After failover, clients must reconnect to the current primary database. For more information, see Configuring Client Connectivity for an Availability Group (SQL Server).

Maintaining Availability During Upgrades

The database administrator for your availability groups can use manual failovers to maintain database availability when you upgrade hardware or software. To use an availability group for software upgrades, the server instance and/or computer node that hosts the target secondary replica must have already received the upgrades.

Note

Failing over availability groups should support a rolling upgrade, but this is not guaranteed.

Forced Manual Failover (with Possible Data Loss)

Forcing failover of an availability group (with possible data loss) is a disaster recovery method that allows you to use a secondary replica as a warm standby server. Because forcing failover risks possible data loss, it should be used cautiously and sparingly. We recommend forcing failover only if you must restore service to your availability databases immediately and are willing to risk losing data. For more information about the prerequisites and recommendations for forcing failover and for an example scenario that uses a forced failover to recover from a catastrophic failure, see Perform a Forced Manual Failover of an Availability Group (SQL Server).

Warning

Forcing failover requires that the WSFC cluster have quorum. For information about configuring quorum and forcing quorum, see Windows Server Failover Clustering (WSFC) with SQL Server.

In This Section:

- How Forced Failover Works
- Risks of Forcing Failover
- Managing the Potential Data Loss

How Forced Failover Works

Forcing failover initiates a smooth transition of the primary role to the target secondary replica which becomes the new primary replica and immediately serves its copies of the databases to clients. When the former primary replica becomes available, it will transition to the secondary role and its databases will become secondary databases.

Every secondary database (including the former primary databases, when they become available). Depending on the previous data synchronization state of a suspended secondary database it might be suitable for salvaging missing committed data for that primary database. On a secondary replica that is configured for read-only access, you can query the secondary databases to manually discover missing data. Then you can issue Transact-SQL statements on the new primary databases to make any necessary changes.

Risks of Forcing Failover

It is essential to understand that forcing failover can cause data loss. Data loss is possible because the secondary replica cannot communicate with the primary replica and, therefore, cannot guarantee that the databases are synchronized. Forcing failover starts a new recovery fork. Because the original primary databases and secondary databases are on different recovery forks, each of them now contains data that the other database does not contain: each original primary database contains whatever changes were not yet sent from its send queue to the former secondary database (the unsent log); the former secondary databases contain whatever changes occur after failover was forced.

If failover is forced because the primary replica has failed, potential data loss is depends on whether any transaction logs had not been sent to the secondary replica before the failure. Under the asynchronous-commit mode, accumulated unsent log is always a possibility. Under synchronous-commit mode, this is possible only until the secondary databases becomes synchronized.

The following table summarizes the possibility of data loss for a particular database on a secondary replica to which you force failover.

Availability mode of Secondary Replica	**Is database synchronized?**	**Is data loss possible?**
Synchronous-commit	Yes	No
Synchronous-commit	No	Yes
Asynchronous-commit	No	Yes

Secondary databases track only two recovery forks, so if you perform multiple forced failovers, any secondary database that did start data synchronization with the previous force failover might not be able to resume. If this occurs, any secondary databases that cannot be resumed will need to be removed from the availability group, restored to the correct point in time, and rejoined to the availability group. A restore will not work across multiple recovery forks, therefore, be sure to perform a log backup after performing more than one forced failover.

Managing the Potential Data Loss

After failover is forced, once the former primary replica is available, assuming that its databases are undamaged, you can attempt to manage the potential data loss. The available approach for managing potential data loss depends on whether the original primary replica has connected to the new primary replica. Assuming that the original primary replica can access the new primary instance, reconnecting occurs automatically and transparently.

The Original Primary Replica Has Reconnected

Typically, after a failure, when the original primary replica restarts it quickly reconnects to its partner. On reconnecting, the original primary replica becomes the secondary replica. Its databases becomes the secondary databases and enter the SUSPENDED state. The new secondary databases will not be not rolled back unless you resume them.

However, the suspended databases are inaccessible; therefore, you cannot inspect them to evaluate what data would be lost if you were to resume a given database. Therefore, the decision on whether to resume or remove a secondary database depends on whether you are willing to accept any data loss, as follows:

- If losing any data would be unacceptable, you should remove the databases from the availability group to salvage them.

The database administrator can now recover the former primary databases and attempt to recover the data that would have been lost. However, when a former primary database comes online, it is divergent from the current primary database, so the database administrator needs to make either the removed database or the current primary database inaccessible to clients to avoid further divergence of the databases and to prevent client-failover issues.

- If losing data would be acceptable to your business goals, you can resume the secondary databases.

  Resuming a new secondary database causes it to be rolled back as the first step in synchronizing the database. If any log records were waiting in the send queue at the time of failure, the corresponding transactions are lost, even if they were committed.

The Original Primary Replica Has Not Reconnected

If you can temporarily prevent the original primary replica from reconnecting over the network to the new primary replica, you can inspect the original primary databases to evaluate what data would be lost if they were resumed.

- If the potential data loss is acceptable

  Allow the original primary replica to reconnect to the new primary replica. Reconnecting causes the new secondary databases to be suspended. To start data synchronization on a database, simply resume it. The new secondary replica drops the original recovery fork for that database, losing any transactions that were never sent to or received by the former secondary replica.

- If the data loss is unacceptable

  If the original primary database contains critical data that would be lost if you resumed the suspended database, you can preserve the data on the original primary database by removing it from the availability group. This causes the database to enter the RESTORING state. At this point, we recommend that you attempt to back up the tail of the removed database's log. Then, you can update the current primary (the former secondary database) by exporting the data you want to salvage from the original primary database and importing it into the current primary database. We recommend taking a full database backup of the updated primary database as quickly as possible.

  Then, on the server instance that hosts the new secondary replica, you can delete the suspended secondary database and create a new secondary database by restoring this backup (and least one subsequent log backup) using RESTORE WITH NORECOVERY. We recommend delaying additional log backups of the current primary databases until the corresponding secondary databases are resumed.

Related Tasks

To configure failover behavior

- Set the Availability Mode of an Availability Replica (SQL Server)

- Set the Failover Mode of an Availability Replica (SQL Server)
- Configure the Flexible Failover Policy to Control Conditions for Automatic Failover (AlwaysOn Availability Groups)

To perform a manual fail over

- Perform a Planned Manual Failover of an Availability Group (SQL Server)
- Perform a Forced Failover of an Availability Group (SQL Server)
- Use the Failover Availability Group Wizard (SQL Server)
- Management of Logins and Jobs for the Databases of an Availability Group (SQL Server)

To configure WSFC Quorum Configuration

- Configure Cluster Quorum NodeWeight Settings
- View Quorum Node Weights
- Force Quorum Election

Related Content

- Microsoft SQL Server AlwaysOn Solutions Guide for High Availability and Disaster Recovery
- SQL Server AlwaysOn Team Blog: The official SQL Server AlwaysOn Team Blog

See Also

AlwaysOn Availability Groups

Availability Modes (AlwaysOn Availability Groups)

Windows Server Failover Clusters (WSFC) with SQL Server

Cross-Database Transactions Not Supported For Database Mirroring or AlwaysOn Availability Groups (SQL Server)

Failover Policy for Failover Cluster Instances

Flexible Failover Policy for Automatic Failover of an Availability Group (SQL Server)

Change the Failover Mode of an Availability Replica

This topic describes how to change the failover mode of an availability replica in an AlwaysOn availability group in SQL Server 2012 by using SQL Server Management Studio, Transact-SQL, or PowerShell. The failover mode is a replica property that determines the failover mode for replicas that run under synchronous-commit availability mode. For more information, see Failover Modes (AlwaysOn Availability Groups) and Availability Modes (AlwaysOn Availability Groups).

- **Before you begin:**

  Prerequisites and Restrictions

  Security

- **To change the availability mode of an availability replica, using:**

  SQL Server Management Studio

Transact-SQL

PowerShell

Before You Begin

Prerequisites and Restrictions

- This task is supported only on primary replicas. You must be connected to the server instance that hosts the primary replica.
- SQL Server Failover Cluster Instances (FCIs) do not support automatic failover by availability groups, so any availability replica that is hosted by an FCI can only be configured for manual failover.

Security

Permissions

Requires ALTER AVAILABILITY GROUP permission on the availability group, CONTROL AVAILABILITY GROUP permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.

Using SQL Server Management Studio

To change the failover mode of an availability replica

1. In Object Explorer, connect to the server instance that hosts the primary replica, and expand the server tree.
2. Expand the **AlwaysOn High Availability** node and the **Availability Groups** node.
3. Click the availability group whose replica you want to change.
4. Right-click the replica, and click **Properties**.
5. In the **Availability Replica Properties** dialog box, use the **Failover mode** drop list to change the failover mode of this replica.

Using Transact-SQL

To change the failover mode of an availability replica

1. Connect to the server instance that hosts the primary replica.
2. Use the ALTER AVAILABILITY GROUP statement, as follows:

   ALTER AVAILABILITY GROUP *group_name* MODIFY REPLICA ON 'server_name'
   ```
     WITH ( {
        AVAILABILITY_MODE = { SYNCHRONOUS_COMMIT | ASYNCHRONOUS_COMMIT }
       | FAILOVER_MODE = { AUTOMATIC | MANUAL }
        } )
   ```
   where

   - group_name is the name of the availability group.
   - { 'system_name[\instance_name]' | 'FCI_network_name[\instance_name]' }

Specifies the address of the instance of SQL Server that hosts the availability replica to be altered. The components of this address are as follows:

system_name

Is the NetBIOS name of the computer system on which a stand-alone server instance resides.

FCI_network_name

Is the network name that is used to access a SQL Server failover cluster in which a target server instance is a SQL Server failover partner (an FCI).

instance_name

Is the name of the instance of SQL Server that hosts the target availability replica. For a default server instance, instance_name is optional.

For more information about these parameters, see ALTER AVAILABILITY GROUP (Transact-SQL).

The following example, entered on the primary replica of the *MyAG* availability group, changes the failover mode to automatic failover on the availability replica that is located on the default server instance on a computer named *COMPUTER01*.

```
ALTER AVAILABILITY GROUP MyAG MODIFY REPLICA ON 'COMPUTER01' WITH
   (FAILOVER_MODE = AUTOMATIC);
```

Using PowerShell

To change the failover mode of an availability replica

1. Change directory (**cd**) to the server instance that hosts the primary replica.
2. Use the **Set-SqlAvailabilityReplica** cmdlet with the **FailoverMode** parameter. When setting a replica to automatic failover, you might need to use the **AvailabilityMode** parameter to change the replica to synchronous-commit availability mode.

   For example, the following command modifies the replica `MyReplica` in the availability group `MyAg` to use synchronous-commit availability mode and to support automatic failover.

   ```
   Set-SqlAvailabilityReplica -AvailabilityMode "SynchronousCommit" -
   FailoverMode "Automatic" `
   -Path
   SQLSERVER:\Sql\PrimaryServer\InstanceName\AvailabilityGroups\MyAg\Repl
   icas\MyReplica
   ```

   Note
   To view the syntax of a cmdlet, use the **Get-Help** cmdlet in the SQL Server PowerShell environment. For more information, see Get Help SQL Server PowerShell.

To set up and use the SQL Server PowerShell provider

- SQL Server PowerShell Provider

See Also

Overview of AlwaysOn Availability Groups

Availability Modes (AlwaysOn Availability Groups)

Failover Modes (AlwaysOn Availability Groups)

Flexible Failover Policy for Automatic Failover of an Availability Group

A flexible failover policy provides granular control over the conditions that cause automatic failover for an availability group. By changing the failure conditions that trigger an automatic failover and the frequency of health checks, you can increase or decrease the likelihood of an automatic failover to support your SLA for high availability.

The flexible failover policy of an availability group is defined by its failure-condition level and health-check timeout threshold. On detecting that an availability group has exceeded its failure condition level or its health-check timeout threshold, the availability group's resource DLL responds back to the Windows Server Failover Clustering (WSFC) cluster. The WSFC cluster then initiates an automatic failover to the secondary replica.

Important

If an availability group exceeds its WSFC failure threshold, the WSFC cluster will not attempt an automatic failover for the availability group. Furthermore, the WSFC resource group of the availability group remains in a failed state until either the cluster administrator manually brings the failed resource group online or the database administrator performs a manual failover of the availability group. The *WSFC failure threshold* is defined as the maximum number of failures supported for the availability group during a given time period. The default time period is six hours, and the default value for the maximum number of failures during this period is n-1, where n is the number of WSFC nodes. To change the failure-threshold values for a given availability group, use the WSFC Failover Manager Console.

This topic contains the following sections:

- Health-Check Timeout Threshold
- Failure-Condition Level
- Related Tasks
- Related Content

Health-Check Timeout Threshold

WSFC resource DLL of the availability group performs a *health check* of the primary replica by calling the sp_server_diagnostics stored procedure on the instance of SQL Server that hosts the primary replica. **sp_server_diagnostics** returns results at an interval that equals 1/3 of the health-check timeout threshold for the availability group. The default health-check timeout threshold is 30 seconds, which causes **sp_server_diagnostics** to return at a 10-second interval. If **sp_server_diagnostics** is slow or is not returning information, the resource DLL will wait for the full interval of the health-check timeout threshold before determining that the primary replica is unresponsive. If the primary replica is unresponsive, an automatic failover is initiated, if currently supported.

Important
sp_server_diagnostics does not perform health checks at the database level.

Failure-Condition Level

Whether the diagnostic data and health information returned by **sp_server_diagnostics** warrants an automatic failover depends on the failure-condition level of the availability group. The *failure-condition level* specifies what failure conditions trigger an automatic failover. There are five failure-condition levels, which range from the least restrictive (level one) to the most restrictive (level five). A given level encompasses the less restrictive levels. Thus, the strictest level, five, includes the four less restrictive conditions, and so forth.

Important
Damaged databases and suspect databases are not detected by any failure-condition level. Therefore, a database that is damaged or suspect (whether due to a hardware failure, data corruption, or other issue) never triggers an automatic failover.

The following table describes the failure-conditions that corresponds to each level.

Level	Failure Condition	Tsql Value	PowerShell Value
One	On server down. Specifies that an automatic failover is initiated when any of the following occurs: • The SQL Server service is down. • The lease of the availability group for connecting to the WSFC cluster expires because no ACK is received from the server instance. This is the least restrictive level.	1	**OnServerDown**
Two	On server	2	**OnServerUnresponsive**

Level	Failure Condition	Tsql Value	PowerShell Value
	unresponsive. Specifies that an automatic failover is initiated when any of the following occurs: • The instance of SQL Server does not connect to cluster, and the user-specified health check timeout threshold of the availability group is exceeded. • The availability replica is in failed state.		
Three	On critical server error. Specifies that an automatic failover is initiated on critical SQL Server internal errors, such as orphaned spinlocks, serious write-access violations, or too much dumping. This is the default level.	3	**OnCriticalServerError**
Four	On moderate server error. Specifies that an automatic failover is initiated on moderate SQL Server internal errors, such as a persistent out-of-memory condition in the SQL Server internal resource	4	**OnModerateServerError**

Level	Failure Condition	Tsql Value	PowerShell Value
	pool.		
Five	On any qualified failure conditions. Specifies that an automatic failover is initiated on any qualified failure conditions, including: • Exhaustion of SQL Engine worker-threads. • Detection of an unsolvable deadlock. This is the most restrictive level.	5	**OnAnyQualifiedFailureConditions**

Note
Lack of response by an instance of SQL Server to client requests is irrelevant to availability groups.

Related Tasks

To configure automatic failover

- Set the Availability Mode of an Availability Replica (SQL Server) (automatic failover requires synchronous-commit availability mode)
- Set the Failover Mode of an Availability Replica (SQL Server)
- Configure the Flexible Failover Policy to Control Conditions for Automatic Failover (AlwaysOn Availability Groups)

Related Content

- Microsoft SQL Server AlwaysOn Solutions Guide for High Availability and Disaster Recovery
- SQL Server AlwaysOn Team Blog: The official SQL Server AlwaysOn Team Blog

See Also

Overview (AlwaysOn Availability Groups)

Availability Modes (AlwaysOn Availability Groups)

[Failover Modes (AlwaysOn Availability Group)](#)

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

[Failover Policy for Failover Cluster Instances](#)

[sp_server_diagnostics (Transact-SQL)](#)

Configure the Flexible Failover Policy to Control Conditions for Automatic Failover (AlwaysOn Availability Groups)

This topic describes how to configure the flexible failover policy for an AlwaysOn availability group by using Transact-SQL or PowerShell in SQL Server 2012. A flexible failover policy provides granular control over the conditions that cause automatic failover for an availability group. By changing the failure conditions that trigger an automatic failover and the frequency of health checks, you can increase or decrease the likelihood of an automatic failover to support your SLA for high availability.

- **Before you begin:**

  [Limitations on Automatic Failovers](#)

  Prerequisites

  Security

- **To configure the flexible failover policy, using:**

  Transact-SQL

  PowerShell

> **Note**
> The flexible failover policy of an availability group cannot be configured by using SQL Server Management Studio.

Before You Begin

Limitations on Automatic Failovers

- For an automatic failover to occur, the current primary replica and one secondary replica must be configured for synchronous-commit availability mode with automatic failover and the secondary replica must be synchronized with the primary replica.
- If an availability group exceeds its WSFC failure threshold, the WSFC cluster will not attempt an automatic failover for the availability group. Furthermore, the WSFC resource group of the availability group remains in a failed state until either the cluster administrator manually brings the failed resource group online or the database administrator performs a manual failover of the availability group. The *WSFC failure threshold* is defined as the maximum number of failures supported for the availability group during a given time period. The default time period is six hours, and the default value for the maximum number of failures during this period is n-1, where n is the number of WSFC nodes. To change the failure-threshold values for a given availability group, use the WSFC Failover Manager Console.

Prerequisites

- You must be connected to the server instance that hosts the primary replica.

Security

Permissions

Task	Permissions
To configure the flexible failover policy for a new availability group	Requires membership in the **sysadmin** fixed server role and either CREATE AVAILABILITY GROUP server permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.
To modify the policy of an existing availability group	Requires ALTER AVAILABILITY GROUP permission on the availability group, CONTROL AVAILABILITY GROUP permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.

Using Transact-SQL

To configure the flexible failover policy

1. Connect to the server instance that hosts the primary replica.
2. For a new availability group, use the CREATE AVAILABILITY GROUP Transact-SQL statement. If you are modifying an existing availability group, use the ALTER AVAILABILITY GROUP Transact-SQL statement.
   - To set the failover condition level, use the FAILURE_CONDITION_LEVEL = n option, where, n is an integer from 1 to 5.

     For example, the following Transact-SQL statement changes the failure-condition level of an existing availability group, `AG1`, to level one:

     ```
     ALTER AVAILABILITY GROUP AG1 SET (FAILURE_CONDITION_LEVEL = 1);
     ```

     The relationship of these integer values to the failure condition levels is as follows:

Tsql Value	Level	Automatic Is Failover Initiated When...
1	One	On server down. The SQL Server service stops because of a failover or restart.
2	Two	On server unresponsive. Any condition of lower value is satisfied, the SQL Server service is connected to the cluster and the health check timeout threshold is exceeded, or the current primary replica is in a failed state.
3	Three	On critical server error. Any condition of lower value is satisfied or an internal critical server error occurs. This is the default level.
4	Four	On moderate server error. Any condition of lower value is satisfied or a moderate Server error occurs.
5	Five	On any qualified failure conditions. Any condition of lower value is satisfied or a qualifying failure condition occurs.

For more information about the failover condition levels, see Flexible Failover Policy for Automatic Failover of an Availability Group (SQL Server).

- To configure the health check timeout threshold, use the HEALTH_CHECK_TIMEOUT = n option, where, n is an integer from 15000 milliseconds (15 seconds) to 4294967295 milliseconds. The default value is 30000 milliseconds (30 seconds)

  For example, the following Transact-SQL statement changes the health-check timeout threshold of an existing availability group, `AG1`, to 60,000 milliseconds (one minute).

```
ALTER AVAILABILITY GROUP AG1 SET (HEALTH_CHECK_TIMEOUT = 60000);
```

Using PowerShell

To configure the flexible failover policy

1. Set default (**cd**) to the server instance that hosts the primary replica.
2. When adding an availability replica to an availability group, use the **New-SqlAvailabilityGroup** cmdlet. When modifying an existing availability replica, use the **Set-SqlAvailabilityGroup** cmdlet.
   - To set the failover condition level, use the **FailureConditionLevel** level parameter, where, level is one of the following values:

Value	Level	Automatic Is Failover Initiated When...
OnServerDown	One	On server down. The SQL Server service stops because of a failover or restart.
OnServerUnresponsive	Two	On server unresponsive. Any condition of lower value is satisfied, the SQL Server service is connected to the cluster and the health check timeout threshold is exceeded, or the current primary replica is in a failed state.
OnCriticalServerError	Three	On critical server error. Any condition of lower value is satisfied or an internal critical server error occurs. This is the default level.
OnModerateServerError	Four	On moderate server error. Any condition of lower value is satisfied or a moderate Server error occurs.
OnAnyQualifiedFailureConditions	Five	On any qualified failure conditions. Any condition of lower value is satisfied or a qualifying failure condition occurs.

For more information about the failover condition levels, see Flexible Failover Policy for Automatic Failover of an Availability Group (SQL Server).

For example, the following command changes the failure-condition level of an existing availability group, `AG1`, to level one.

```
Set-SqlAvailabilityGroup `
```

```
-Path SQLSERVER:\Sql\PrimaryServer\InstanceName\AvailabilityGroups\MyAg
`

-FailureConditionLevel OnServerDown
```

- To set the health check timeout threshold, use the **HealthCheckTimeout** n parameter, where, n is an integer from 15000 milliseconds (15 seconds) to 4294967295 milliseconds. The default value is 30000 milliseconds (30 seconds).

  For example, the following command changes the health-check timeout threshold of an existing availability group, `AG1`, to 120,000 milliseconds (two minutes).

  ```
  Set-SqlAvailabilityGroup `

  -Path SQLSERVER:\Sql\PrimaryServer\InstanceName\AvailabilityGroups\MyAG
  `

  -HealthCheckTimeout 120000
  ```

Note
To view the syntax of a cmdlet, use the **Get-Help** cmdlet in the SQL Server PowerShell environment. For more information, see SQL Server PowerShell Help.

To set up and use the SQL Server PowerShell provider

- Using the SQL Server PowerShell Provider
- SQL Server PowerShell Help

See Also

Overview of AlwaysOn Availability Groups (SQL Server)

Availability Modes (AlwaysOn Availability Groups)

Failover Modes (AlwaysOn Availability Group)

Windows Server Failover Clustering (WSFC) with SQL Server

Failover Policy for Failover Cluster Instances

sp_server_diagnostics (Transact-SQL)

Possible Failures During Sessions Between Availability Replicas

Physical, operating system, or SQL Server problems can cause a failure in a session between two availability replicas. An availability replica does not regularly check the components on which Sqlservr.exe relies to verify whether they are functioning correctly or have failed. However, for some types of failures, the affected component reports an error to Sqlservr.exe. An error reported by another component is called a *hard error*. To detect other failures that would otherwise go unnoticed, AlwaysOn Availability Groups implements its own session-timeout mechanism. Specifies the session-timeout period in seconds. This time-out period is the maximum time that a server instance waits to receive a PING message from another instance before considering that other instance to be disconnected. When a session timeout occurs between two availability replicas, the availability replicas assume that a failure has occurred and declares a *soft error*.

Important

Failures in databases other than the primary database are not detectable. Moreover, a data disk failure is unlikely to be detected unless the database is restarted because of a data disk failure.

The speed of error detection and, therefore, the reaction time to a failure, depends on whether the error is hard or soft. Some hard errors, such as network failures are reported immediately. However, in some cases, component-specific time-out periods can delay the reporting of some hard errors. For soft errors, the length of the session-timeout period determines the speed of error detection. By default, this period is 10 seconds. This is the minimum recommended value.

Failures Due to Hard Errors

Possible causes of hard errors include (but are not limited to) the following conditions:

- A broken connection or wire
- A bad network card
- A router change
- Changes in the firewall
- Endpoint reconfiguration
- Loss of the drive where the transaction log resides
- Operating system or process failure

For example, when the log drive on the primary database becomes unresponsive and fails, the operating system informs Sqlservr.exe that a serious error has occurred.

Some components, such as network components and some IO subsystems, have their own time-outs to determine failures. Such time-outs are independent of AlwaysOn Availability Groups, which has no knowledge of them and is completely unaware of their behavior. In these cases, the time-out delay increases the time between a failure and when the availability replica receives the resulting hard error.

Note

The only active error checking performed for availability replicas occurs for soft error cases. For more information, see "Failures Due to Soft Errors," later in this topic.

To help you interpret the error conditions that occur on the network, ask a network engineer what error messages are sent to a port when the following events occur on a TCP connection:

- DNS is not working.
- Cables are unplugged.
- Microsoft Windows has a firewall that blocks a specific port.
- The application that is monitoring a port fails.
- A Windows-based server is renamed.

- A Windows-based server is rebooted.

Note
AlwaysOn Availability Groups does not protect against problems specific to client accessing the servers. For example, consider a case in which a public network adapter handles client connections to the primary replica, while a private network interface card handles traffic among the server instances that are hosting the replicas of an availability group. In this case, failure of the public network adapter would prevent clients from accessing the databases.

Failures Due to Soft Errors

Conditions that might cause session timeouts include (but are not limited to) the following:

- Network errors such as TCP link time-outs, dropped or corrupted packets, or packets that are in an incorrect order.
- A hanging operating system, server, or database state.
- A Windows server timing out.
- Insufficient computing resources, such as a CPU or disk overload, the transaction log filling up, or the system is running out of memory or threads. In these cases, you must increase the time-out period, reduce the workload, or change the hardware to handle the workload.

The Session-Timeout Mechanism

Because soft errors are not detectable directly by a server instance, a soft error could potentially cause an availability replica to wait indefinitely for a response from the other availability replica in a session. To prevent this, AlwaysOn Availability Groups implements a session time-out mechanism, based on the connected availability replicas sending out a ping on each open connection at a fixed interval. Receiving a ping during the time-out period indicates that the connection is still open and that the server instances are communicating over it. On receiving a ping, a replica resets its time-out counter on that connection. For information about the relationship of availability mode and session timeouts, see Availability Modes (AlwaysOn Availability Groups).

The primary and secondary replicas ping each other to signal that they are still active, and a session-timeout limit prevents either replica from waiting indefinitely to receive a ping from the other replica. The session-timeout limit is a user-configurable replica property with a default value of 10 seconds. Receiving a ping during the time-out period indicates that the connection is still open and that the server instances are communicating over it. On receiving a ping, an availability replica resets its time-out counter on that connection.

If no ping is received from the other replica within the session-timeout period, the connection times out. The connection is closed, and the timed-out replica enters the DISCONNECTED state. Even if a disconnected replica is configured for synchronous-commit mode, transactions will not wait for that replica to reconnect and resynchronize.

Responding to an Error

Regardless of the type of error, a server instance that detects an error responds appropriately based on the role of the instance, the availability mode of the session, and the state of any other connection in the session. For information about what occurs on the loss of a partner, see Availability Modes (AlwaysOn Availability Groups).

Related Tasks

To change the time-out value (synchronous-commit availability mode only)

- Set the Session-Timeout Period for an Availability Replica (SQL Server)

To view the current time-out value

- Query **session_timeout** in sys.availability_replicas (Transact-SQL).

See Also

AlwaysOn Availability Groups (SQL Server)

Active Secondaries: Backup on Secondary Replicas (AlwaysOn Availability Groups)

The AlwaysOn Availability Groups active secondary capabilities include support for performing backup operations on secondary replicas. Backup operations can put significant strain on I/O and CPU (with backup compression). Offloading backups to a synchronized or synchronizing secondary replica allows you to use the resources on server instance that hosts the primary replica for your tier-1 workloads.

Note
RESTORE statements are not allowed on either the primary or secondary databases of an availability group.

- Supported Backup Types
- Configuring Where Backup Jobs Run
- Related Tasks

Supported Backup Types

Only BACKUP LOG is fully supported on secondary replicas. BACKUP DATABASE supports only copy-only full backups of the database, files, or filegroups. Differential backups are not supported on secondary replicas.

Note
Copy-only backups do not impact the log chain. Also, copy-only backups do not clear the differential bitmap.

Configuring Where Backup Jobs Run

Performing backups on a secondary replica to offload the backup workload from the primary production server is a great benefit. But it introduces significant complexity to the process of determining where backup jobs should run. To address this, you need to configure where backup jobs run, as follows:

1. Configure the availability group to specify which availability replicas where you would prefer backups to be performed. For more information, see Specifying Where You Would Prefer to Perform Backups, later in this section.
2. Create scripted backup jobs for every availability database on every server instance that hosts an availability replica that is a candidate for performing backups. For more information, see Scripting of Backup Jobs, later in this section.

Configuring Where You Would Prefer to Perform Backups

To configure the availability group use the following AlwaysOn Availability Groups settings:

- *Automated backup preference* (configured for the availability group as a whole)

  At an availability group level, specify whether backups should or should not run on the primary replica. The possible preferences for where backups should run are as follows:

Preference	Description
Only on the primary replica	Backups should always occur on the primary replica. This alternative is useful if you need backup features, such as creating differential backups, that are not supported when backup is run on a secondary replica.
On secondary replicas	Backups should occur on a secondary replica except when the primary replica is the only replica online. In that case, the backup should occur on the primary replica. This is the default behavior.
Only on secondary replicas	Backups should never be performed on the primary replica. If the primary replica is the only replica online, the backup should not occur.
No preference	Backup jobs should ignore the role of the availability replicas when choosing the replica to perform backups. Note backup jobs might evaluate other factors such as backup priority of each availability replica in combination with its operational state and connected state.

  For information about how to specify these settings, see Configure Backup on Availability Replicas (SQL Server).

- *Backup priority* (configured individually for each availability replica)

To specify whether an availability replica is a candidate for running backup jobs, specify its backup priority. By specifying different backup priorities for different availability replicas, you can specify an ordered preference among secondary for running backup jobs. To indicate the priority for a given availability replica, set backup priority to a value from 0 to 100, as follows:

Setting	Description
1..100	The relative priority of a given replica relative to the backup priorities of the other replicas in the availability group. 100 is the highest priority. By default, all secondary replicas have the same backup priority (50), making all replicasle equal candidates for running backup jobs unless you specify a different value for at least one replica.
0	The availability replica will never be chosen for performing backups. This is useful, for example, for a remote secondary replica to which you do not want your production backup jobs to fail over. This is also useful for a computer that lacks the facilities to handle backups.

For information about how to specify these settings, see Configure Backup on Availability Replicas (SQL Server).

Scripting of Backup Jobs

The availability group and replica configuration settings have no effect unless you script backup jobs to use these setting. On every availability replica whose backup priority is greater than zero (>0), you need to script backup jobs for the databases in the availability group.

You can determine whether the current replica is the preferred backup replica by calling the sys.fn_hadr_backup_is_preferred_replica function. If the availability replica that is hosted by the current server instance is the preferred replica for backups, this function returns 1. If not, the function returns 0. By running a simple script on each availability replica that that queries this function, to determine which replica should run a given backup job, and if the function returns '1', runs a backup job.

The logic for this script is as follows:

If (top-priority replica is local)

Run backup job

Else

Exit with success

Tip

If you use the Maintenance Plan Wizard to create a given backup job, the job will automatically include the scripting logic that calls and checks the **sys.fn_hadr_backup_is_preferred_replica** function. However, the backup job will not return the "This is not the preferred replica…" message. Be sure to create the job(s) for each availability database on every server instance that hosts an availability replica for the availability group.

Scripting a backup job using this sort of logic enables you to schedule the job to run on every availability replica on the same schedule. Each of these jobs looks at the same data to determine which job should run, so only one of the scheduled job actually proceeds to the backup stage. In the event of a failover, none of the scripts or jobs need to be touched. Also, if you reconfigure an availability group to add an availability replica, managing the backup job requires simply copying or scheduling the backup job. If you remove an availability replica, simply delete the backup job from the server instance that hosted that replica.

For a sample script, see the "Follow Up: After Configuring Backup on Secondary Replicas" section of Configure Backup on Availability Replicas (SQL Server).

Related Tasks

To configure backup on secondary replicas

- Configure Backup on Availability Replicas (SQL Server)

To determine whether the current replica is the preferred backup replica

- sys.fn_hadr_backup_is_preferred_replica

To create a backup job

- Use the Maintenance Plan Wizard
- Implement Jobs

See Also

Overview (AlwaysOn Availability Groups)

Copy-Only Backups (SQL Server)

CREATE AVAILABILITY GROUP (Transact-SQL)

ALTER AVAILABILITY GROUP (Transact-SQL)

Configure Backup on Availability Replicas

This topic describes how to configure backup on secondary replicas for an AlwaysOn availability group by using SQL Server Management Studio, Transact-SQL, or PowerShell in SQL Server 2012.

Note
For an introduction to backup on secondary replicas, see Backup on Secondary Replicas (AlwaysOn Availability Groups).

- **Before you begin:**

  Limitations and Restrictions

  Prerequisites

  Security
- **To configure backup on secondary replicas , using:**

  SQL Server Management Studio

  Transact-SQL

  PowerShell
- **Follow Up:** After configuring backup on secondary replicas
- To Obtain Information About Backup Preference Settings
- Related Content

Before You Begin

Limitations and Restrictions

- Only BACKUP LOG is fully supported on secondary replicas. BACKUP DATABASE supports only copy-only full backups of the database, files, or filegroups. Differential backups are not supported on secondary replicas.

  Note
  Copy-only backups do not impact the log chain. Also, copy-only backups do not clear the differential bitmap.
- To back up a secondary database, a secondary replica must be able to communicate with the primary replica and must be SYNCHRONIZED or SYNCHRONIZING.
- Configuring an availability group to support backup on secondary replicas merely establishes your backup preferences for where to perform backups. It is important to understand that the preference is not enforced by SQL Server, so the automated backup preference has no impact on ad-hoc backups. To take the automated backup preference into account, on each availability replica whose backup priority is greater than zero (>0), you need to script backup jobs for the databases in the availability group. For more information, see Follow Up: After Configuring Backup on Secondary Replicas, later in this topic.

Prerequisites

- You must be connected to the server instance that hosts the primary replica.

Security

Permissions

Task	Permissions
To configure backup on secondary replicas when creating an availability group	Requires membership in the **sysadmin** fixed server role and either CREATE AVAILABILITY GROUP server permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.
To modify an availability group or availability replica	Requires ALTER AVAILABILITY GROUP permission on the availability group, CONTROL AVAILABILITY GROUP permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.

Using SQL Server Management Studio

To configure backup on secondary replicas

1. In Object Explorer, connect to the server instance that hosts the primary replica, and click the server name to expand the server tree.
2. Expand the **AlwaysOn High Availability** node and the **Availability Groups** node.
3. Click the availability group whose backup preferences you want to configure, and select the **Properties** command.
4. In the **Availability Group Properties** dialog box, select **Backup Preferences** page.
5. On the **Where should backups occur?** panel, select the automated backup preference for the availability group, one of:

Prefer Secondary

Specifies that backups should occur on a secondary replica except when the primary replica is the only replica online. In that case, the backup should occur on the primary replica. This is the default option.

Secondary only

Specifies that backups should never be performed on the primary replica. If the primary replica is the only replica online, the backup should not occur.

Primary

Specifies that the backups should always occur on the primary replica. This option is useful if you need backup features, such as creating differential backups, that are not supported when backup is run on a secondary replica.

Important

If you plan to use log shipping to prepare any secondary databases for an availability group, set the automated backup preference to **Primary** until all the secondary databases have been prepared and joined to the availability group.

Any Replica

Specifies that you prefer that backup jobs ignore the role of the availability replicas when choosing the replica to perform backups. Note backup jobs might evaluate other factors such as backup priority of each availability replica in combination with its operational state and connected state.

Important

There is no enforcement of the backup-preference setting. The interpretation of this preference depends on the logic, if any, that you script into back jobs for the databases in a given availability group. For more information, see Backup on Secondary Replicas (AlwaysOn Availability Groups).

6. Use the **Replica backup priorities** grid to change the backup priority of the availability replicas. This grid displays the current backup priority of each server instance that hosts a replica for the availability group. The grid columns are as follows:

   Server Instance

   The name of the instance of SQL Server that hosts the availability replica.

   Backup Priority (Lowest=1, Highest=100)

   Specifies your priority for performing backups on this replica relative to the other replicas in the same availability group. The value is an integer in the range of 0..100. 1 indicates the lowest priority, and 100 indicates the highest priority. If **Backup Priority** = 1, the availability replica would be chosen for performing backups only if no higher priority availability replicas are currently available.

   Exclude Replica

   Select if you never want this availability replica to be chosen for performing backups. This is useful, for example, for a remote availability replica to which you never want backups to fail over.

7. To commit your changes, click **OK**.

Alternative ways to access the Backup Preferences page

- Use the New Availability Group Wizard
- Use the Add Replica to Availability Group Wizard
- Use the New Availability Group Dialog Box (SQL Server Management Studio)

Using Transact-SQL

To configure backup on secondary replicas

1. Connect to the server instance that hosts the primary replica.

2. For a new availability group, use the CREATE AVAILABILITY GROUP Transact-SQL statement. If you are modifying an existing availability group, use the ALTER AVAILABILITY GROUP Transact-SQL statement.

   - Optionally, configure the automated backup preference for the availability group. The default setting is to prefer secondary replicas. To change this setting, use the AUTOMATED_BACKUP_PREFERENCE option, as follows:

     … AUTOMATED_BACKUP_PREFERENCE = { PRIMARY | SECONDARY_ONLY | SECONDARY | NONE }

     where,

     PRIMARY

     Specifies that the backups should always occur on the primary replica. This option is useful if you need backup features, such as creating differential backups, that are not supported when backup is run on a secondary replica.

     Important
     If you plan to use log shipping to prepare any secondary databases for an availability group, set the automated backup preference to PRIMARY until all the secondary databases have been prepared and joined to the availability group.

     SECONDARY_ONLY

     Specifies that backups should never be performed on the primary replica. If the primary replica is the only replica online, the backup should not occur.

     SECONDARY

     Specifies that backups should occur on a secondary replica except when the primary replica is the only replica online. In that case, the backup should occur on the primary replica. This is the default behavior.

     NONE

     Specifies that you prefer that backup jobs ignore the role of the availability replicas when choosing the replica to perform backups. Note backup jobs might evaluate other factors such as backup priority of each availability replica in combination with its operational state and connected state.

     For example, the following command changes the automated backup preference of an existing availability group, `AG1`, to PRIMARY:

     ```
     ALTER AVAILABILITY GROUP [AG1] SET ( AUTOMATED_BACKUP_PREFERENCE =
     PRIMARY );
     ```

     For an example of setting AUTOMATED_BACKUP_PREFERENCE for a new availability group, see CREATE AVAILABILITY GROUP (Transact-SQL).

   - To specify your priority for performing backups on a given availability replica relative to the other replicas in the same availability group, specify the BACKUP_PRIORITY option when you add or modify an availability replica with the CREATE AVAILABILITY GROUP or ALTER AVAILABILITY GROUP Transact-SQL statement, as follows:

… [ADD | MODIFY] REPLICA ON <server_instance> WITH (BACKUP_PRIORITY = n)

where, n is an integer in the range of 0..100. These values have the following meanings:

- 1..100 indicates that the availability replica could be chosen for performing backups. 1 indicates the lowest priority, and 100 indicates the highest priority. If BACKUP_PRIORITY = 1, the availability replica would be chosen for performing backups only if no higher priority availability replicas are currently available.
- 0 indicates that this availability replica will never be chosen for performing backups. This is useful, for example, for a remote availability replica to which you never want backups to fail over.

For example, to modify an existing availability replica to support backups when running under the secondary role: `ALTER AVAILABILITY GROUP [AG1] MODIFY REPLICA ON computer01 WITH (BACKUP_PRIORITY = 70);`

For an example of setting BACKUP_PRIORITY when creating an availability group, see CREATE AVAILABILITY GROUP (Transact-SQL).

Using PowerShell

To configure backup on secondary replicas

1. Set default (**cd**) to the server instance that hosts the primary replica.
2. Optionally, configure the backup priority of each availability replica that you are adding or modifying. This priority is used by the server instance that hosts the primary replica to decide which replica should service an automated backup request on a database in the availability group (the replica with highest priority is chosen). This priority can be any number between 0 and 100, inclusive. A priority of 0 indicates that the replica should not be considered as a candidate for servicing backup requests. The default setting is 50.

   When adding an availability replica to an availability group, use the **New-SqlAvailabilityReplica** cmdlet. When modifying an existing availability replica, use the **Set-SqlAvailabilityReplica** cmdlet. In either case, specify the **BackupPriority** n parameter, where n is a value from 0 to 100.

   For example, the following command sets the backup priority of the availability replica `MyReplica` to **60**.

   ```
   Set-SqlAvailabilityReplica -BackupPriority 60 `
   -Path
   SQLSERVER:\Sql\Computer\Instance\AvailabilityGroups\MyAg\AvailabilityR
   eplicas\MyReplica
   ```

3. Optionally, configure the automated backup preference for the availability group that you are creating or modifying. This preference indicates how a backup job should evaluate the primary replica when choosing where to perform backups. The default setting is to prefer secondary replicas.

When creating an availability group, use the **New-SqlAvailabilityGroup** cmdlet. When modifying an existing availability group, use the **Set-SqlAvailabilityGroup** cmdlet. In either case, specify the **AutomatedBackupPreference** parameter.

where,

Primary

Specifies that the backups should always occur on the primary replica. This option is useful if you need backup features, such as creating differential backups, that are not supported when backup is run on a secondary replica.

Important

If you plan to use log shipping to prepare any secondary databases for an availability group, set the automated backup preference to **Primary** until all the secondary databases have been prepared and joined to the availability group.

SecondaryOnly

Specifies that backups should never be performed on the primary replica. If the primary replica is the only replica online, the backup should not occur.

Secondary

Specifies that backups should occur on a secondary replica except when the primary replica is the only replica online. In that case, the backup should occur on the primary replica. This is the default behavior.

None

Specifies that you prefer that backup jobs ignore the role of the availability replicas when choosing the replica to perform backups. Note backup jobs might evaluate other factors such as backup priority of each availability replica in combination with its operational state and connected state.

For example, the following command sets the **AutomatedBackupPreference** property on the availability group `MyAg` to **SecondaryOnly**. Automated backups of databases in this availability group will never occur on the primary replica, but will be redirected to the secondary replica with the highest backup priority setting.

```
Set-SqlAvailabilityGroup `
-Path
SQLSERVER:\Sql\PrimaryServer\InstanceName\AvailabilityGroups\MyAg `
-AutomatedBackupPreference SecondaryOnly
```

Note

To view the syntax of a cmdlet, use the **Get-Help** cmdlet in the SQL Server PowerShell environment. For more information, see SQL Server PowerShell Help.

To set up and use the SQL Server PowerShell provider

- Using the SQL Server PowerShell Provider

- SQL Server PowerShell Help

Follow Up: After Configuring Backup on Secondary Replicas

To take the automated backup preference into account for a given availability group, on each server instance that hosts an availability replica whose backup priority is greater than zero (>0), you need to script backup jobs for the databases in the availability group. To determine whether the current replica is the preferred backup replica, use the sys.fn_hadr_backup_is_preferred_replica function in your backup script. For example, a typical snippet of a backup-job script would look like:

```
IF (NOT sys.fn_hadr_backup_is_preferred_replica(@DBNAME))
BEGIN
      Select 'This is not the preferred replica, exiting with success';
      RETURN 0 - This is a normal, expected condition, so the script returns
success
END
BACKUP DATABASE @DBNAME TO DISK=<disk>
   WITH COPY_ONLY;
```

Tip

If you use the Maintenance Plan Wizard to create a given backup job, the job will automatically include the scripting logic that calls and checks the **sys.fn_hadr_backup_is_preferred_replica** function. However, the backup job will not return the "This is not the preferred replica…" message. Be sure to create the job(s) for each availability database on every server instance that hosts an availability replica for the availability group.

To Obtain Information About Backup Preference Settings

The following are useful for obtaining information that is relevant for backup on secondary.

View	Information	Relevant Columns
sys.fn_hadr_backup_is_preferred_replica	Is the current replica the preferred backup replica?	Not applicable.
sys.availability_groups	Automated backup preference	**automated_backup_preference** **automated_backup_preference_desc**

View	Information	Relevant Columns
sys.availability_replicas	Backup priority of a given availability replica	**backup_priority**
sys.dm_hadr_availability_replica_states	Is replica local to the server instance? Current role Operational state Connected state Synchronization health of an availability replica	**is_local** **role**, **role_desc** **operational_state**, **operational_state_desc** **connected_state**, **connected_state_desc** **synchronization_health**, **synchronization_health_desc**

Related Content

- Microsoft SQL Server AlwaysOn Solutions Guide for High Availability and Disaster Recovery
- SQL Server AlwaysOn Team Blog: The official SQL Server AlwaysOn Team Blog

See Also

AlwaysOn Availability Groups (SQL Server)

Backup on Secondary Replicas

Active Secondaries: Readable Secondary Replicas (AlwaysOn Availability Groups)

The AlwaysOn Availability Groups active secondary capabilities include support for read-only access to one or more secondary replicas (*readable secondary replicas*). A readable secondary replica allows read-only access to all its secondary databases. However, readable secondary databases are not set to read-only. They are dynamic. A given secondary database changes as changes on the corresponding primary database are applied to the secondary database. For a typical secondary replica, the data in the secondary databases is in near real time. Furthermore, full-text indexes are synchronized with the secondary databases. In many circumstances, data latency between a primary database and the corresponding secondary database is only a few seconds.

Security settings that occur in the primary databases are persisted to the secondary databases. This includes users, database roles, and applications roles together with their respective permissions and transparent data encryption (TDE), if enabled on the primary database.

Note

Though you cannot write data to secondary databases, you can write to read-write databases on the server instance that hosts the secondary replica, including user databases and system databases such as tempdb.

AlwaysOn Availability Groups also supports the re-routing of read-intent connection requests to a readable secondary replica (*read-only routing*). For information about read-only routing, see Using a Listener to Connect to a Read-Only Secondary Replica (Read-Only Routing).

In this Topic:

- Benefits
- Prerequisites for the Availability Group
- Limitations and Restrictions
- Performance Considerations
- Capacity Planning Considerations
- Related Tasks
- Related Content

Benefits

Directing read-only connections to readable secondary replicas provides the following benefits:

- Offloads your secondary read-only workloads from your primary replica, which conserves its resources for your mission critical workloads. If you have mission critical read-workload or the workload that cannot tolerate latency, you should run it on the primary.
- Improves your return on investment for the systems that host readable secondary replicas.

In addition, readable secondaries provide robust support for read-only operations, as follows:

- Temporary statistics on readable secondary database optimize read-only queries. For more information, see Statistics for Read-Only Access Databases, later in this topic.
- Read-only workloads use row versioning to remove blocking contention on the secondary databases. All queries that run against the secondary databases are automatically mapped to snapshot isolation transaction level, even when other transaction isolation levels are explicitly set. Also, all locking hints are ignored. This eliminates reader/writer contention.

Prerequisites for the Availability Group

- **Readable secondary replicas (required)**

The database administrator needs to configure one or more replicas so that, when running under the secondary role, they allow either all connections (just for read-only access) or only read-intent connections.

Note
Optionally, the database administrator can configure any of the availability replicas to exclude read-only connections when running under the primary role.

For more information, see About Client Connection Access to Availability Replicas (SQL Server).

- **Availability group listener**

  To support read-only routing, an availability group must possess an availability group listener. The read-only client must direct its connection requests to this listener, and the client's connection string must specify the application intent as "read-only." That is, they must be *read-intent connection requests*.

- **Read only routing**

  Read-only routing refers to the ability of SQL Server to route incoming read-intent connection requests, that are directed to an availability group listener, to an available readable secondary replica. The prerequisites for read-only routing are as follows:

  - To support read-only routing, a readable secondary replica requires a read-only routing URL. This URL takes effect only when the local replica is running under the secondary role. The read-only routing URL must be specified on a replica-by-replica basis, as needed. Each read-only routing URL is used for routing read-intent connection requests to a specific readable secondary replica. Typically, every readable secondary replica is assigned a read-only routing URL.
  - Each availability replica that is to support read-only routing when it is the primary replica requires a read-only routing list. A given read-only routing list takes effect only when the local replica is running under the primary role. This list must be specified on a replica-by-replica basis, as needed. Typically, each read-only routing list would contain every read-only routing URL, with the URL of the local replica at the end of the list.

    Note
    Read-intent connection requests are routed to the first available readable secondary on the read-only routing list of the current primary replica. There is no load balancing.

  For more information, see Configure Read-Only Routing for an Availability Group (SQL Server).

Note
For information about availability group listeners and more information about read-only routing, see Availability Group Listeners, Client Connectivity, and Application Failover (AlwaysOn Availability Groups).

Limitations and Restrictions

Some operations are not fully supported, as follows:

- As soon as a readable secondary replica joins the availability group, the secondary replica can start accepting connections to its secondary databases. However, if any active transactions exist on a primary database, row versions will not be fully available immediately on the corresponding secondary database. Any active transactions that existed on the primary replica when the secondary replica was configured must be committed or rolled back. Until this process completes, the transaction isolation level mapping on the secondary database is incomplete and queries are temporarily blocked.

  Note
  Running long transaction will impact the number of versioned rows kept.
- Change tracking and change data capture are not supported on secondary databases that belong to a readable secondary replica:
  - Change tracking is explicitly disabled on secondary databases.
  - Change data capture can be enabled on a secondary database, but this is not supported.
- Because read operations are mapped to snapshot isolation transaction level, the cleanup of ghost records on the primary replica can be blocked by transactions on one or more secondary replicas. The ghost record cleanup task will automatically clean up the ghost records on the primary replica when they are no longer needed by any secondary replica. This is similar to what is done when you run transaction(s) on the primary replica. In the extreme case on the secondary database, you will need to kill a long running read-query that is blocking the ghost cleanup. Note, the ghost clean can be blocked if the secondary replica gets disconnected or when data movement is suspended on the secondary database. This state also prevents log truncation, so if this state persists, we recommend that you remove this secondary database from the availability group.
- The DBCC SHRINKFILE operation might fail on the primary replica if the file contains ghost records that are still needed on a secondary replica.

Note
If you query the sys.dm_db_index_physical_stats dynamic management view on a server instance that is hosting a readable secondary replica, you might encounter a REDO blocking issue. This is because this dynamic management view acquires an IS lock on the specified user table or view that can block requests by a REDO thread for an X lock on that user table or view.

Performance Considerations

This section discusses several performance considerations for readable secondary databases

In This Section:

- Data Latency
- Read-Only Workload Impact

- Indexing
- Statistics for Read-Only Access Databases

Data Latency

Implementing read-only access to secondary replicas is useful if your read-only workloads can tolerate some data latency. In situations where data latency is unacceptable, consider running read-only workloads against the primary replica.

The primary replica sends log records of changes on primary database to the secondary replicas. On each secondary database, a dedicated redo thread applies the log records. On a read-access secondary database, a given data change does not appear in query results until the log record that contains the change has been applied to the secondary database and the transaction has been committed on primary database.

This means that there is some latency, usually only a matter of seconds, between the primary and secondary replicas. In unusual cases, however, for example if network issues reduce throughput, latency can become significant. Latency increases when I/O bottlenecks occur and when data movement is suspended. To monitor suspended data movement, you can use the AlwaysOn Dashboard or the sys.dm_hadr_database_replica_states dynamic management view.

Read-Only Workload Impact

When you configure a secondary replica for read-only access, your read-only workloads on the secondary databases consume system resources, such as CPU and I/O from redo threads, especially if the read-only workloads are highly I/O-intensive.

Also, read-only workloads on the secondary replicas can block data definition language (DDL) changes that are applied through log records. Even though the read operations do not take shared locks because of row versioning, these operations take schema stability (Sch-S) locks, which can block redo operations that are applying DDL changes.

Be aware of best practices around building queries, and exercise those best practices in the secondary databases. For example, schedule long-running queries such as aggregations of data during times of low activity.

Note
If a redo thread is blocked by queries on a secondary replica, the **sqlserver.lock_redo_blocked** XEvent is raised.

Indexing

To optimize read-only workloads on the readable secondary replicas, you may want to create indexes on the tables in the secondary databases. Because you cannot make schema or data changes on the secondary databases, create indexes in the primary databases and allow the changes to transfer to the secondary database through the redo process.

To monitor index usage activity on an secondary replica, query the **user_seeks**, **user_scans**, and **user_lookups** columns of the sys.dm_db_index_usage_stats dynamic management view.

Statistics for Read-Only Access Databases

Statistics on columns of tables and indexed views are used to optimize query plans. For availability groups, statistics that are created and maintained on the primary databases are automatically persisted on the secondary databases as part of applying the transaction log records. However, the read-only workload on the secondary databases may need different statistics than those that are created on the primary databases. However, because secondary databases are restricted to read-only access, statistics cannot be created on the secondary databases.

To address this problem, the secondary replica creates and maintains temporary statistics for secondary databases in tempdb. The suffix _readonly_database_statistic is appended to the name of temporary statistics to differentiate them from the permanent statistics that are persisted from the primary database.

Only SQL Server can create and update temporary statistics. However, you can delete temporary statistics and monitor their properties using the same tools that you use for permanent statistics:

- Delete temporary statistics using the DROP STATISTICS Transact-SQL statement.
- Monitor statistics using the **sys.stats** and **sys.stats_columns** catalog views. **sys_stats** includes a column, **is_temporary**, to indicate which statistics are permanent and which are temporary.

For more information about SQL Server statistics, see Using Statistics to Improve Query Performance.

In This Section:

- Stale Permanent Statistics on Secondary Databases
- Limitations and Restrictions

Stale Permanent Statistics on Secondary Databases

SQL Server detects when permanent statistics on a secondary database are stale. But changes cannot be made to the permanent statistics except through changes on the primary database. For query optimization, SQL Server creates temporary statistics on the secondary database and uses these statistics instead of the stale permanent statistics.

When the permanent statistics are updated on the primary database, they are automatically persisted to the secondary database. Then SQL Server uses the updated permanent statistics, which are more current than the temporary statistics.

If the availability group fails over, temporary statistics are deleted on all of the secondary replicas.

Limitations and Restrictions

- Because temporary statistics are stored in tempdb, a restart of the SQL Server service causes all temporary statistics to disappear.
- The suffix _readonly_database_statistic is reserved for statistics generated by SQL Server. You cannot use this suffix when creating statistics on a primary database. For more information, see Statistics.

Capacity Planning Considerations

- Readable secondary replicas can require space in tempdb for two reasons:
    - Snapshot isolation level copies row versions into tempdb.
    - Temporary statistics for secondary databases are created and maintained in tempdb. The temporary statistics can cause a slight increase in the size of tempdb. For more information, see Statistics for Read-Only Access Databases, later in this section.
- When you configure read-access for one or more secondary replicas, the primary databases add 14 bytes of overhead on deleted, modified, or inserted data rows to store pointers to row versions on the secondary databases. This 14-byte overhead is carried over to the secondary databases. As the 14-byte overhead is added to data rows, page splits might occur.

  The row version data is not generated by the primary databases. Instead, the secondary databases generate the row versions. However, row versioning increases data storage in both the primary and secondary databases.

  The addition of the row version data depends on the snapshot isolation or read-committed snapshot isolation (RCSI) level setting on the primary database. The table below describes the behavior of versioning on a readable secondary database under different settings.

Readable secondary replica?	**Snapshot isolation or RCSI level enabled?**	**Primary Database**	**Secondary Database**
No	No	No row versions or 14-byte overhead	No row versions or 14-byte overhead
No	Yes	Row versions and 14-byte overhead	No row versions, but 14-byte overhead
Yes	No	No row versions, but 14-byte overhead	Row versions and 14-byte overhead
Yes	Yes	Row versions and 14-byte overhead	Row versions and 14-byte overhead

Related Tasks

- Configure Read-Only Connection Access on an Availability Replica (SQL Server)
- Configure Read-Only Routing on an Availability Group (SQL Server)
- Create or Configure an Availability Group Listener (SQL Server
- Monitor Availability Groups (Transact-SQL)
- View and Change Availability Replica Properties (SQL Server Management Studio)
- Create an Availability Group (SQL Server)

Related Content

- SQL Server AlwaysOn Team Blog: The official SQL Server AlwaysOn Team Blog

See Also

Overview of AlwaysOn Availability Groups (SQL Server)

Client Connection Access to Availability Replicas (SQL Server)

Client Connectivity and Application Failover (AlwaysOn Availability Groups)

Using Statistics to Improve Query Performance

About Client Connection Access to Availability Replicas

In an AlwaysOn availability group, you can configure one or more availability replicas to allow read-only connections when running under the secondary role (that is, when running as a secondary replica). You can also configure each availability replica to allow or exclude read-only connections when running under the primary role (that is, when running as the primary replica).

To facilitate client access to primary or secondary databases of a given availability group, you should define an availability group listener. By default, the availability group listener directs incoming connections to the primary replica. However, you can configure an availability group to support read-only routing, which enables its availability group listener to redirect the connection requests of read-intent applications to a readable secondary replica. For more information, see Configure Read-Only Routing on an Availability Group.

During a failover, a secondary replica transitions to the primary role and the former primary replica transitions to the secondary role. During the failover process, all client connections to both the primary replica and secondary replicas are terminated. After the failover, when a client reconnects to the availability group listener, the listener reconnects the client to the new primary replica, except for a read-intent connect request. If read-only routing is configured on the client and on the server instances that hosts the new primary replica and on at least one readable secondary replica, read-intent connection requests are re-routed to a secondary replica that supports the type of connection access that the client requires. To ensure a graceful client experience after a failover, it is important to configure connection access for both the secondary and primary roles of every availability replica.

Note
For information about the availability group listener, which handles client connection requests, see Client Connectivity and Application Failover (AlwaysOn Availability Groups).

In This Topic:

- Types of Connection Access Supported by the Secondary Role
- Types of Connection Access Supported by the Primary Role
- How the Connection Access Configuration Affects Client Connectivity

- Related Tasks
- Related Content

Types of Connection Access Supported by the Secondary Role

The secondary role supports three alternatives for client connections, as follows:

No connections

No user connections are allowed. Secondary databases are not available for read access. This is the default behavior in the secondary role.

Only read-intent connections

The secondary database(s) are available only for connection for which the **Application Intent** connection property is set to **ReadOnly** (*read-intent connections*).

For information about this connection property, see SQL Server Native Client Support for High Availability, Disaster Recovery.

Allow any read-only connection

The secondary database(s) are all available for read access connections. This option allows lower versioned clients to connect.

For more information, see Configure Connection Access on an Availability Replica (SQL Server).

Types of Connection Access Supported by the Primary Role

The primary role supports two alternatives for client connections, as follows:

All connections are allowed

Both read-write and read-only connections are allowed to primary databases. This is the default behavior for the primary role.

Allow only read-write connections

When the **Application Intent** connection property is set to **ReadWrite** or is not set, the connection is allowed. Connections for which the **Application Intent** connection string keyword is set to **ReadOnly** are not allowed. Allowing only read-write connections can help prevent your customers from connecting a read-intent work load to the primary replica by mistake.

For information about this connection property, see Using Connection String Keywords with SQL Server Native Client.

For more information, see Configure Connection Access on an Availability Replica (SQL Server).

How the Connection Access Configuration Affects Client Connectivity

The connection access settings of a replica determine whether a connection attempt fails or succeeds. The following table summarizes whether a given connection attempt succeeds or fails for each the connection-access setting.

Replica Role	Connection Access Supported on Replica	Connection Intent	Connection-Attempt Result
Secondary	All	Read-intent, read-write, or no connection intent specified	Success
Secondary	None (This is the default secondary behavior.)	Read-intent, read-write, or no connection intent specified	Failure
Secondary	Read-intent only	Read-intent	Success
Secondary	Read-intent only	Read-write or no connection intent specified	Failure
Primary	All (This is the default primary behavior.)	Read-only, read-write, or no connection intent specified	Success
Primary	Read-write	Read-intent only	Failure
Primary	Read-write	Read-write or no connection intent specified	Success

For information about configuring an availability group to accept client connections to its replicas, see Client Connectivity and Application Failover (AlwaysOn Availability Groups).

Example Connection-Access Configuration

Depending on how different availability replicas are configured for connection access, support for client connections might change after an availability group fails over. For example, consider an availability group for which reporting is performed on remote asynchronous-commit secondary replicas. All of the read-only applications for the databases in this availability group set their **Application Intent** connection property to **ReadOnly**, so that all read-only connections are read-intent connections.

This example availability group possesses two synchronous-commit replicas at the main computing center and two asynchronous-commit replicas at a satellite site. For the primary role, all the replicas are configured for read-write access, which prevents read-intent connections to the primary replica in all situations. The synchronous commit secondary role uses the default connection-access configuration ("none"), which prevents all client connections under the secondary role.   In contrast, the asynchronous commit replicas are configured to permit read-intent connections under the secondary role. The following table summarize this example configuration:

Replica	Commit Mode	Initial Role	Connection Access for Secondary Role	Connection Access for Primary Role
Replica1	Synchronous	Primary	None	Read-write
Replica2	Synchronous	Secondary	None	Read-write
Replica3	Asynchronous	Secondary	Read-intentonly	Read-write
Replica4	Asynchronous	Secondary	Read-intent only	Read-write

Typically, in this example scenario, failovers occur only between the synchronous-commit replicas, and immediately after the failover, read-intent applications are able to reconnect to one of the asynchronous-commit secondary replicas. However, when a disaster occurs at the main computing center both synchronous-commit replicas are lost. The database administrator at the satellite site responds by performing a forced manual failover to an asynchronous-commit secondary replica. The secondary databases on the remaining secondary replica are suspended by the forced failover, making them unavailable for read-only workloads. The new primary replica, which is configured for read-write connections, prevents the read-intent workload from competing with the read-write workload. This means that until the database administrator resumes the secondary databases on the remaining asynchronous-commit secondary replica, read-intent clients cannot connect to any availability replica.

Related Tasks

- Configure Connection Access on an Availability Replica (SQL Server)
- Configure Read-Only Routing on an Availability Group
- Monitor Availability Groups (Transact-SQL)
- View and Change Availability Replica Properties (SQL Server Management Studio)
- Create an Availability Group (SQL Server)

Related Content

- Microsoft SQL Server AlwaysOn Solutions Guide for High Availability and Disaster Recovery
- SQL Server AlwaysOn Team Blog: The official SQL Server AlwaysOn Team Blog

See Also

Overview of AlwaysOn Availability Groups (SQL Server)

Client Connectivity and Application Failover (AlwaysOn Availability Groups)

[Using Statistics to Improve Query Performance]()

Configure Read-Only Access on an Availability Replica

By default both read-write and read-intent access are allowed to the primary replica and no connections are allowed to secondary replicas of an AlwaysOn availability group. This topic describes how to configure connection access on an availability replica of an AlwaysOn availability group in SQL Server 2012 by using SQL Server Management Studio, Transact-SQL, or PowerShell.

For information about the implications of enabling read-only access for a secondary replica and for an introduction to connection access, see [Client Connection Access to Availability Replicas (SQL Server)]() and [Readable Secondary Replicas]().

- **Before you begin:**

  Prerequisites and Restrictions

  Security

- **To configure access on an availability replica, using:**

  SQL Server Management Studio

  Transact-SQL

  PowerShell

- **Follow Up:** After Configuring Read-Only Access for an Availability Replica
- Related Tasks
- Related Content

Before You Begin

Prerequisites and Restrictions

- To configure different connection access, you must be connected to the server instance that hosts the primary replica.

Security

Permissions

Task	**Permissions**
To configure replicas when creating an availability group	Requires membership in the **sysadmin** fixed server role and either CREATE AVAILABILITY GROUP server permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.
To modify an availability replica	Requires ALTER AVAILABILITY GROUP

Task	Permissions
	permission on the availability group, CONTROL AVAILABILITY GROUP permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.

Using SQL Server Management Studio

To configure access on an availability replica

1. In Object Explorer, connect to the server instance that hosts the primary replica, and expand the server tree.
2. Expand the **AlwaysOn High Availability** node and the **Availability Groups** node.
3. Click the availability group whose replica you want to change.
4. Right-click the availability replica, and click **Properties**.
5. In the **Availability Replica Properties** dialog box, you can change the connection access for the primary role and for the secondary role, as follows:
   - For the secondary role, select a new value from the **Readable secondary** drop list, as follows:

     No

     No user connections are allowed to secondary databases of this replica. They are not available for read access. This is the default setting.

     Read-intent only

     Only read-only connections are allowed to secondary databases of this replica. The secondary database(s) are all available for read access.

     Yes

     All connections are allowed to secondary databases of this replica, but only for read access. The secondary database(s) are all available for read access.
   - For the primary role, select a new value from the **Connections in primary role** drop list, as follows:

     Allow all connections

     All connections are allowed to the databases in the primary replica. This is the default setting.

     Allow read/write connections

     When the Application Intent property is set to **ReadWrite** or the Application Intent connection property is not set, the connection is allowed. Connections where the Application Intent connection property is set to **ReadOnly** are not allowed. This can help prevent customers from connecting a read-intent work load to the primary replica by mistake. For more information about Application Intent connection property, see Using Connection String Keywords with SQL Server Native Client.

Using Transact-SQL

To configure access on an availability replica

Note
For an example of this procedure, see Example (Transact-SQL), later in this section.

1. Connect to the server instance that hosts the primary replica.
2. If you are specifying a replica for a new availability group, use the CREATE AVAILABILITY GROUP Transact-SQL statement. If you are adding or modifying a replica of an existing availability group, use the ALTER AVAILABILITY GROUP Transact-SQL statement.
   - To configure connection access for the secondary role, in the ADD REPLICA or MODIFY REPLICA WITH clause, specify the SECONDARY_ROLE option, as follows:

     SECONDARY_ROLE **(** ALLOW_CONNECTIONS **=** { NO | READ_ONLY | ALL } **)**

     where,

     NO

     No direct connections are allowed to secondary databases of this replica. They are not available for read access. This is the default setting.

     READ_ONLY

     Only read-only connections are allowed to secondary databases of this replica. The secondary database(s) are all available for read access.

     ALL

     All connections are allowed to secondary databases of this replica, but only for read access. The secondary database(s) are all available for read access.
3. To configure connection access for the primary role, in the ADD REPLICA or MODIFY REPLICA WITH clause, specify the PRIMARY_ROLE option, as follows:

   PRIMARY_ROLE **(** ALLOW_CONNECTIONS **=** { READ_WRITE | ALL } **)**

   where,

   READ_WRITE

   Connections where the Application Intent connection property is set to **ReadOnly** are disallowed. When the Application Intent property is set to **ReadWrite** or the Application Intent connection property is not set, the connection is allowed. For more information about Application Intent connection property, see Using Connection String Keywords with SQL Server Native Client.

   ALL

   All connections are allowed to the databases in the primary replica. This is the default setting.

Example (Transact-SQL)

The following example adds a secondary replica to an availability group named *AG2*. A stand-alone server instance, *COMPUTER03\HADR_INSTANCE*, is specified to host the new availability replica. This replica configured to allow only read-write connections for the primary role and to allow only read-intent connections for secondary role.

```
ALTER AVAILABILITY GROUP AG2

   ADD REPLICA ON

      'COMPUTER03\HADR_INSTANCE' WITH

         (

         ENDPOINT_URL = 'TCP://COMPUTER03:7022',

         PRIMARY_ROLE ( ALLOW_CONNECTIONS = READ_WRITE ),

         SECONDARY_ROLE (ALLOW_CONNECTIONS = READ_ONLY )

         );

GO
```

Using PowerShell

To configure access on an availability replica

Note

For a code example, see Example (PowerShell), later in this section.

1. Change directory (**cd**) to the server instance that hosts the primary replica.
2. When adding an availability replica to an availability group, use the **New-SqlAvailabilityReplica** cmdlet. When modifying an existing availability replica, use the **Set-SqlAvailabilityReplica** cmdlet. The relevant parameters are as follows:
   - To configure connection access for the secondary role, specify the **ConnectionModeInSecondaryRole** secondary_role_keyword parameter, where secondary_role_keyword equals one of the following values:

     AllowNoConnections

     No direct connections are allowed to the databases in the secondary replica and the databases are not available for read access. This is the default setting.

     AllowReadIntentConnectionsOnly

     Connections are allowed only to the databases in the secondary replica where the Application Intent property is set to **ReadOnly**. For more information about this property, see Using Connection String Keywords with SQL Server Native Client.

     AllowAllConnections

     All connections are allowed to the databases in the secondary replica for read-only access.

- To configure connection access for the primary role, specify **ConnectionModeInPrimaryRole** primary_role_keyword, where primary_role_keyword equals one of the following values:

  AllowReadWriteConnections

  Connections where the Application Intent connection property is set to ReadOnly are disallowed. When the Application Intent property is set to ReadWrite or the Application Intent connection property is not set, the connection is allowed. For more information about Application Intent connection property, see Using Connection String Keywords with SQL Server Native Client.

  AllowAllConnections

  All connections are allowed to the databases in the primary replica. This is the default setting.

Note

To view the syntax of a cmdlet, use the **Get-Help** cmdlet in the SQL Server 2012 PowerShell environment. For more information, see Get Help SQL Server PowerShell.

To set up and use the SQL Server PowerShell provider

- SQL Server PowerShell Provider

Example (PowerShell)

The following example, sets the both the **ConnectionModeInSecondaryRole** and **ConnectionModeInPrimaryRole** parameters to **AllowAllConnections**.

```
Set-Location SQLSERVER:\SQL\PrimaryServer\default\AvailabilityGroups\MyAg

$primaryReplica = Get-Item "AvailabilityReplicas\PrimaryServer"

Set-SqlAvailabilityReplica -ConnectionModeInSecondaryRole
"AllowAllConnections" `

-InputObject $primaryReplica

Set-SqlAvailabilityReplica -ConnectionModeInPrimaryRole "AllowAllConnections"
`

-InputObject $primaryReplica
```

Follow Up: After Configuring Read-Only Access for an Availability Replica

Read-only access to a readable secondary replica

- When using the bcp Utility or sqlcmd Utility, you can specify read-only access to any secondary replica that is enabled for read-only access by specifying the **-K ReadOnly** switch.
- To enable client applications to connect to readable secondary replicas:

	Prerequisite	Link
☐	Ensure that the availability group has a listener.	Create or Configure an Availability Group Listener (SQL Server)
☐	Configure read-only routing for the availability group.	Configure Read-Only Routing on an Availability Group (SQL Server)

Factors that might affect triggers and jobs after a failover

If you have triggers and jobs that will fail when running on a non-readable secondary database or on a readable secondary database, you need to script the triggers and jobs to check on a given replica to determine whether the database is a primary database or is a readable secondary database. To obtain this information, use the DATABASEPROPERTYEX function to return the **Updatability** property of the database. To identify a read-only database, specify READ_ONLY as the value, as follows:

```
DATABASEPROPERTYEX([db name],'Updatability') = N'READ_ONLY'
```

To identify a read-write database, specify READ_WRITE as the value.

Related Tasks

- Configure Read-Only Routing on an Availability Group (SQL Server)
- Create or Configure an Availability Group Listener (SQL Server)

Related Content

- Microsoft SQL Server AlwaysOn Solutions Guide for High Availability and Disaster Recovery
- SQL Server AlwaysOn Team Blog: The official SQL Server AlwaysOn Team Blog

See Also

Overview of AlwaysOn Availability Groups (SQL Server)

Readable Secondary replicas

Client Connection Access to Availability Replicas (SQL Server)

Availability Group Listeners, Client Connectivity, and Application Failover

This topic contains information about considerations for AlwaysOn Availability Groups client connectivity and application-failover functionality.

In This Topic:

- Availability Group Listeners
- Using a Listener to Connect to the Primary Replica
- Using a Listener to Connect to a Read-Only Secondary Replica (Read-Only Routing)
  - To Configure Availability Replicas for Read-Only Routing
  - Read-Only Application Intent and Read-Only Routing
- Bypassing Availability Group Listeners
- Behavior of Client Connections on Failover
- Supporting Availability Group Multi-Subnet Failovers
- Availability Group Listeners and SSL Certificates
- Availability Group Listeners and Server Principal Names (SPNs)
- Related Tasks
- Related Content

Availability Group Listeners

You can provide client connectivity to the database of a given availability group by creating an availability group listener. An availability group listener is a virtual network name (VNN) to which clients can connect in order to access a database in a primary or secondary replica of an AlwaysOn availability group. An availability group listener enables a client to connect to an availability replica without knowing the name of the physical instance of SQL Server to which the client is connecting. The client connection string does not need to be modified to connect to the current location of the current primary replica.

An availability group listener consists of a Domain Name System (DNS) listener name, listener port designation, and one or more IP addresses. Only the TCP protocol is supported by availability group listener. The DNS name of the listener must also be unique in the domain and in NetBIOS. When you create a new availability group listener it becomes a resource in a cluster with an associated virtual network name (VNN), virtual IP (VIP), and availability group dependency. A client uses DNS to resolve the VNN into multiple IP addresses and then tries to connect to each address, until a connection request succeeds or until the connection requests time out.

If read-only routing is configured for one or more readable secondary replicas, read-intent client connections to the primary replica are redirected to a readable secondary replica. Also, if the primary replica goes offline on one instance of SQL Server, and a new primary replica comes online on another instance of SQL Server, the availability group listener enables clients to connect to the new primary replica.

For essential information about availability group listeners, see Prerequisites, Restrictions, and Recommendations for AlwaysOn Client Connectivity (SQL Server).

In This Section:

- Availability Group Listener Configuration
- Selecting an Availability Group Listener Port

Availability Group Listener Configuration

An availability group listener is defined by the following:

- The Virtual Network Name (VNN)
- The listener port (listens for incoming requests against the listener name)
- One or more Virtual IPs (VIPs) that are configured for one or more subnets to which the availability group can failover
- Configured to use either DHCP or a static IP

For the majority of the common listener configurations, you can create the first availability group listener simply by using Transact-SQL statements or PowerShell cmdlets. You can configure an availability group listener to use the Dynamic Host Configuration Protocol (DHCP) if an availability group resides on a single subnet. DHCP offers an easy setup for an availability group that does not require disaster recovery to a remote site on a separate subnet.

However in a case where your availability groups extend across subnets in a multi-subnet domain, an availability group listener must use static IP addresses, not DHCP.

Important
We do not recommend using DHCP in conjunction with an availability group listener in a production environment. In the event of down time, if the DHCP IP lease expires, it will take additional time to re-register the new DHCP IP address associated with the listener DNS name.

Hybrid network configurations and DHCP across subnets are not supported for availability group listeners. This is because when a failover happens, a dynamic IP might be expired or released, which jeopardizes overall high availability.

Selecting an Availability Group Listener Port

When configuring an availability group listener, you must designate a port. You can configure the default port to 1433 in order to allow for simplicity of the client connection strings. If using 1433, you do not need to designate a port number in a connection string. Also, since each availability group listener will have a separate virtual network name, each availability group listener configured on a single WSFC can be configured to reference the same default port of 1433.

You can also designate a non-standard listener port; however this means that you will also need to explicitly specify a target port in your connection string whenever connecting to the availability group listener. You will also need to open permission on the firewall for the non-standard port.

If you use the default port of 1433 for availability group listener VNNs, you will still need to ensure that no other services on the cluster node are using this port; otherwise this would cause a port conflict.

If one of the instances of SQL Server is already listening on TCP port 1433 via the instance listener and there are no other services (including additional instances of SQL Server) on the computer listening on port 1433, this will not cause a port conflict with the availability group listener. This is because the availability group listener can share the same TCP port inside the same service process. However multiple instances of SQL Server (side-by-side)should not be configured to listen on the same port.

Using a Listener to Connect to the Primary Replica

To use an availability group listener to connect to the primary replica for read-write access, the connection string specifies the availability group listener DNS name. If an availability group primary replica changes to a new replica, existing connections that use an availability group listener's network name are disconnected. New connections to the availability group listener are then directed to the new primary replica. An example of a basic connection string for the ADO.NET provider (System.Data.SqlClient) is as follows:

```
Server=tcp: AGListener,1433;Database=MyDB;IntegratedSecurity=SSPI
```

You can still choose to directly reference the instance of SQL Server name of the primary or secondary replicas instead of using the availability group listener server name, however if you choose to do so you will lose the benefit of new connections being directed automatically to the current primary replica. You will also lose the benefit of read-only routing.

Using a Listener to Connect to a Read-Only Secondary Replica (Read-Only Routing)

Read-only routing refers to the ability of SQL Server to route incoming connections to an availability group listener to a secondary replica that is configured to allow read-only workloads. An incoming connection referencing an availability group listener name can automatically be routed to a read-only replica if the following are true:

- At least one secondary replica is set to read-only access, and each read-only secondary replica and the primary replica are configured to support read-only routing. For more information, see To Configure Availability Replicas for Read-Only Routing, later in this section.
- The connection string references an availability group listener, and the application intent of the incoming connection is set to read-only (for example, by using the **Application Intent=ReadOnly** keyword in the ODBC or OLEDB connection strings or connection attributes or properties). For more information, see Read-Only Application Intent and Read-Only Routing, later in this section.

To Configure Availability Replicas for Read-Only Routing

A database administrator must configure the availability replicas as follows:

1. For each availability replica that you want to configure as a readable secondary replica, a database administrator must configure the following settings, which take effect only under the secondary role:
    - Connection access must be set to "all" or "read only".
    - The read-only routing URL must be specified.

2. For each of these replicas, a read-only routing list must be specified for the primary role. Specify one or more server names as routing targets.

Related Tasks

- Configure Connection Access on an Availability Replica (SQL Server)
- Configure an Availability Group for Read-Only Routing (SQL Server)

Read-Only Application Intent and Read-Only Routing

The application intent connection string property expresses the client application's request to be directed either to a read-write or read-only version of an availability group database. To use read-only routing, a client must use an application intent of read-only in the connection string when connecting to the availability group listener. Without the read-only application intent, connections to the availability group listener are directed to the database on the primary replica.

The application intent attribute is stored in the client's session during login and the instance of SQL Server will then process this intent and determine what to do according to the configuration of the availability group and the current read-write state of the target database in the secondary replica.

An example of a connection string for the ADO.NET provider (System.Data.SqlClient) that designates read-only application intent is as follows:

```
Server=tcp:AGListener,1433;Database=AdventureWorks;IntegratedSecurity=SSPI;Ap
plicationIntent=ReadOnly
```

In this connection string example, the client is attempting to connect to an availability group listener named `AGListener` on port 1433 (you may also omit the port if the availability group listener is listening on 1433). The connection string has the **ApplicationIntent** property set to **ReadOnly**, making this a *read-intent connection string*. Without this setting, the server would not have attempted a read-only routing of the connection.

The primary database of the availability group processes the incoming read-only routing request and attempts to locate an online, read-only replica that is joined to the primary replica and is configured for read-only routing. The client receives back connection information from the primary replica server and connects to the identified read-only replica.

Note that the application intent can be sent from a client driver to a down-level instance of SQL Server. In this case, application intent of read-only is ignored and the connection proceeds as normal.

You can bypass read-only routing by not setting the application intent connection property to **ReadOnly** (when not designated, the default is **ReadWrite** during login) or by connecting directly to the primary replica instance of SQL Server instead of using the availability group listener name. Read-only routing will also not occur if you connect directly to a read-only replica.

Related Tasks

- SQL Server Native Client Support for High Availability, Disaster Recovery
- Using Connection String Keywords with SQL Server Native Client

Bypassing Availability Group Listeners

While availability group listeners enable support for failover redirection and read-only routing, client connections are not required to use them. A client connection can also directly reference the instance of SQL Server instead of connecting to the availability group listener.

To the instance of SQL Server it is irrelevant whether a connection logs in using the availability group listener or using another instance endpoint. The instance of SQL Server will verify the state of the targeted database and either allow or disallow connectivity based on the configuration of the availability group and the current state of the database on the instance. For example, if a client application connects directly to a instance of SQL Server port and connects to a target database hosted in an availability group, and the target database is in primary state and online, then connectivity will succeed. If the target database is offline or in a transitional state, connectivity to the database will fail.

Alternatively, while migrating from database mirroring to AlwaysOn Availability Groups, applications can specify the database mirroring connection string as long as only one secondary replica exists and it disallows user connections. For more information, see Using Database-Mirroring Connection Strings with Availability Groups, later in this section.

Using Database-Mirroring Connection Strings with Availability Groups

If an availability group possesses only one secondary replica and is not configured to allow read-access to the secondary replica, clients can connect to the primary replica by using a database mirroring connection string. This approach can be useful while migrating an existing application from database mirroring to an availability group, as long as you limit the availability group to two availability replicas (a primary replica and one secondary replica). If you add additional secondary replicas, you will need to create an availability group listener for the availability group and update your applications to use the availability group listener DNS name.

When using database mirroring connection strings, the client can use either SQL Server Native Client or .NET Framework Data Provider for SQL Server. The connection string provided by a client must minimally supply the name of one server instance, the *initial partner name*, to identify the server instance that initially hosts the availability replica to which you intend to connect. Optionally, the connection string can also supply the name of another server instance, the *failover partner name*, to identify the server instance that initially hosts the secondary replica as the failover partner name.

For more information about database mirroring connection strings, see Connect Clients to a Database Mirroring Session (SQL Server).

Behavior of Client Connections on Failover

When an availability group failover occurs, existing persistent connections to the availability group are terminated and the client must establish a new connection in order to continue working with the same primary database or read-only secondary database. While a failover is occurring on the server side, connectivity to the availability group may fail, forcing the client application to retry connecting until the primary is brought fully back online.

If the availability group comes back online during a client application's connection attempt but before the connect timeout period, the client driver may successfully connect during one of its internal retry attempts and no error will be surfaced to the application in this case.

Supporting Availability Group Multi-Subnet Failovers

If you are using client libraries that support the MultiSubnetFailover connection option in the connection string, you can optimize availability group failover to a different subnet by setting MultiSubnetFailover to "True" or "Yes", depending on the syntax of the provider you are using.

Note

We recommend this setting for both single and multi-subnet connections to availability groups listeners and to SQL Server Failover Cluster Instance names. Enabling this option adds additional optimizations, even for single-subnet scenarios.

The **MultiSubnetFailover** connection option only works with the TCP network protocol and is only supported when connecting to an availability group listener and for any virtual network name connecting to SQL Server 2012.

An example of a for the ADO.NET provider (System.Data.SqlClient) connection string that enables multi-subnet failover is as follows:

```
Server=tcp:AGListener,1433;Database=AdventureWorks;IntegratedSecurity=SSPI;
MultiSubnetFailover=True
```

The **MultiSubnetFailover** connection option should be set to **True** even if the availability group only spans a single subnet. This allows you to preconfigure new clients to support future spanning of subnets without any need for future client connection string changes and also optimizes failover performance for single subnet failovers. While the **MultiSubnetFailover** connection option is not required, it does provide the benefit of a faster subnet failover. This is because the client driver will attempt to open up a TCP socket for each IP address in parallel associated with the availability group. The client driver will wait for the first IP to respond with success and once it does, will then use it for the connection.

Availability Group Listeners and SSL Certificates

When connecting to an availability group listener, if the participating instances of SQL Server use SSL certificates in conjunction with session encryption, the connecting client driver will need to support the Subject Alternate Name in the SSL certificate in order to force encryption. SQL Server driver support for certificate Subject Alternative Name is planned for ADO.NET (SqlClient), Microsoft JDBC and SQL Native Client (SNAC).

A X.509 certificate must be configured for each participating server node in the failover cluster with a list of all availability group listeners set in the Subject Alternate Name of the certificate.

For example, if the WSFC has three availability group listeners with the names `AG1_listener.Adventure-Works.com`, `AG2_listener.Adventure-Works.com`, and `AG3_listener.Adventure-Works.com`, the Subject Alternative Name for the certificate should be set as follows:

```
CN = ServerFQDN
```

```
SAN = ServerFQDN,AG1_listener.Adventure-Works.com, AG2_listener.Adventure-
Works.com, AG3_listener.Adventure-Works.com
```

Availability Group Listeners and Server Principal Names (SPNs)

A Server Principal Name (SPN) must be configured in Active Directory by a domain administrator for each availability group listener name in order to enable Kerberos for the client connection to the availability group listener. When registering the SPN, you must use the service account of the server instance that hosts the availability replica . For the SPN to work across all replicas, the same service account must be used for all instances in the WSFC cluster that hosts the availability group.

Use the **setspn** Windows command line tool to configure the SPN. For example to configure an SPN for an availability group named `AG1listener.Adventure-Works.com` hosted on a set of instances of SQL Server all configured to run under the domain account `corp/svclogin2`:

```
setspn -A MSSQLSvc/AG1listener.Adventure-Works.com:1433 corp/svclogin2
```

For more information about manual registration of a SPN for SQL Server, see Register a Service Principal Name for Kerberos Connections.

Related Tasks

- Prerequisites, Restrictions, and Recommendations for AlwaysOn Client Connectivity (SQL Server)
- Create or Configure an Availability Group Listener (SQL Server)
- View Availability Group Listener Properties (SQL Server)
- Remove an Availability Group Listener (SQL Server)
- Configure Connection Access on an Availability Replica (SQL Server)
- Configure Read-Only Routing on an Availability Group (SQL Server)

Related Content

- Microsoft SQL Server AlwaysOn Solutions Guide for High Availability and Disaster Recovery
- Introduction to the Availability Group Listener (a SQL Server AlwaysOn team blog)
- SQL Server AlwaysOn Team Blog: The official SQL Server AlwaysOn Team Blog

See Also

Overview of AlwaysOn Availability Groups

Prerequisites, Restrictions, and Recommendations for AlwaysOn Client Connectivity (SQL Server)

Connection Access on Availability Replicas (SQL Server)

Readable Secondary Replicas

Connect Clients to a Database Mirroring Session (SQL Server)

Prerequisites, Restrictions, and Recommendations for AlwaysOn Client Connectivity

This topic describes considerations for client connectivity to AlwaysOn Availability Groups, including prerequisites, restrictions, and recommendations for client configurations and settings.

In this Topic:

- Limitations and Recommendations
- Windows Hotfixes that Support Availability Group Listeners
- SQL Server Instance Prerequisites
- Troubleshoot Failure to Create an Availability Group Listener Because of Active Directory Quotas
- Permissions
- Related Tasks
- Related Content

Limitations and Recommendations

- **Availability group listeners support only the TCP/IP protocol.** To connect to an availability group listener, a client must use a TCP connection string.
- **To avoid potential NetBIOS conflicts, we recommend that you use a unique 15-character prefix for every availability group listener name.** If two WSFC clusters are controlled by the same Active Directory and you try to create availability group listeners in both clusters using names with more than 15 characters and with an identical 15 character prefix, you will get an error reporting that the Virtual Network Name resource could not be brought online.

  For information about prefix naming rules for DNS names, see Assigning Domain Names, in the Windows Server 2008 and Windows Server 2008 R2 documentation.

Windows Hotfixes that Support Availability Group Listeners

Depending on your cluster topology, several additional Windows Server 2008 Service Pack 2 (SP2) or Windows Server 2008 R2 hotfixes might be applicable for supporting connections to availability group listeners. The following table identifies these hotfixes. The hotfixes can be installed in any order.

Note

For information about all the hotfixes that support AlwaysOn Availability Groups, see Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups (SQL Server).

	Applies to Win 2008 SP2	Applies to Win 2008 R2 SP1	To Support...	Hotfix	Link
☐	√	√	**Internet Protocol Security (IPsec)**	If your environment uses IPsec connections, you could experience a long time delay (about two or three minutes) when a client computer reestablishes the IPsec connection to a virtual network name (in this context, to connect to the availability group listener). If you use IPsec connections, we recommend that you review the specific scenarios detailed in Knowledge Base article (KB 980915).	KB 980915: A long time delay occurs when you reconnect an IPSec connection from a computer that is running Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, or Windows Server 2008 R2
☐	√	√	**IPv6**	If your Windows Server topology uses IP version 6 (IPv6), the WSFC Cluster service requires about 30 seconds to fail over the IPv6 IP	• KB 2578103 (Windows Server 2008): The Cluster service takes about 30 seconds to fail over IPv6 IP

	Applies to Win 2008 SP2	Applies to Win 2008 R2 SP1	To Support…	Hotfix	Link
				address. This causes clients to wait for about 30 seconds to reconnect to the IPv6 IP address. If you use IPv6, we recommend that you review the specific scenarios detailed in Knowledge Base article 2578103 or 2578113, depending on your Windows Server operating system.	addresses in Windows Server 2008 • KB 2578113 (Windows Server 2008 R2): **Windows Server 2008 R2:** The Cluster service takes about 30 seconds to fail over IPv6 IP addresses in Windows Server 2008 R2
☐	√	√	**No Router Between cluster and application server**	If no router exists between the failover cluster and the application server, the Cluster service fails over network-related resources slowly. This delays client reconnections after an availability group fails over. In the absence of a router, we recommend that you review the specific scenarios detailed in	KB 2582281: Slow failover operation if no router exists between the cluster and an application server

	Applies to Win 2008 SP2	Applies to Win 2008 R2 SP1	To Support...	Hotfix	Link
				Knowledge Base article 2582281 and install the hotfix, if applicable to your environment.	
☐		√	**Faster failover to local replicas**	If a WSFC node is running Windows Server 2008 R2 Service Pack 1 (SP1), ensure that the hotfix described in Knowledge Base article 2687741 is installed. This hotfix improves the performance of AlwaysOn Availability Groups failover to local replicas.	KB 2687741: A hotfix that improves the performance of the "AlwaysOn Availability Group" feature in SQL Server 2012 is available for Windows Server 2008 R2

Server Instance Prerequisites for Supporting Client Connections to Availability Group Listeners (Server Instance)

Note
For information about all the server-instance prerequisites and requirements for using AlwaysOn Availability Groups, see Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups (SQL Server).

	To Support...	Prerequisite	Links
☐	**Keberos**	If you want an availability	Register a Service Principal

	To Support...	Prerequisite	Links
		group to work with Kerberos: • All server instances that host an availability replica for the availability group must use the same SQL Server service account. • The domain administrator needs to manually register a Service Principal Name (SPN) with Active Directory on the SQL Server service account for the virtual network name (VNN) of the availability group listener. If the SPN is registered on an account other than the SQL Server service account, authentication will fail. **Important** If you change the SQL Server service account, the domain administrator will need to manually re-register the SPN.	Name for Kerberos Connections **Brief explanation:** Kerberos and SPNs enforce mutual authentication. The SPN maps to the Windows account that starts the SQL Server services. If the SPN is not registered correctly or if it fails, the Windows security layer cannot determine the account associated with the SPN, and Kerberos authentication cannot be used. **Note** NTLM does not have this requirement.

Troubleshoot Failure to Create an Availability Group Listener Because of Active Directory Quotas

The creation of a new availability group listener may fail upon creation because you have reached an Active Directory quota for the participating cluster node machine account. For more information, see the following articles:

- How to Troubleshoot the Cluster Service Account When It Modifies Computer Objects
- Active Directory Quotas

Permissions

For information about the permissions required for creating or modifying an availability group listener, see Create or Configure an Availability Group Listener (SQL Server).

Related Tasks

- Creation and Configuration of Availability Groups (SQL Server)
- Create or Configure an Availability Group Listener (SQL Server)
- View Availability Group Listener Properties (SQL Server)
- Remove an Availability Group Listener (SQL Server)

Related Content

- Microsoft SQL Server AlwaysOn Solutions Guide for High Availability and Disaster Recovery
- SQL Server AlwaysOn Team Blog: The official SQL Server AlwaysOn Team Blog
- Failover Cluster Step-by-Step Guide: Configuring Accounts in Active Directory
- A long time delay occurs when you reconnect an IPSec connection from a computer that is running Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, or Windows Server 2008 R2
- The Cluster service takes about 30 seconds to fail over IPv6 IP addresses in Windows Server 2008 R2
- Slow failover operation if no router exists between the cluster and an application server
- How to troubleshoot the Cluster service account when it modifies computer objects
- Active Directory Quotas
- Assigning Domain Names

See Also

Overview of AlwaysOn Availability Groups (SQL Server)

Failover Clustering and AlwaysOn Availability Groups (SQL Server)

Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups (SQL Server)

Client Connectivity and Application Failover (AlwaysOn Availability Groups)

Client Connection Access to Availability Replicas (SQL Server)

Create or Configure an Availability Group Listener

This topic describes how to create or configure a single *availability group listener* for an AlwaysOn availability group by using SQL Server Management Studio, Transact-SQL, or PowerShell in SQL Server 2012.

- **Before you begin:**

  Does a Listener Exist for this Availability Group Already?

  Limitations and Restrictions

  Recommendations

  Prerequisites

  Requirements for the DNS Name of an Availability Group Listener

  Windows Permissions

  SQL Server Permissions
- **To create or configure an availability group listener, using:**

  SQL Server Management Studio

  Transact-SQL

  PowerShell
- **Follow Up:**

  After Creating an Availability Group Listener

  To Create An Additional Listener for an Availability Group (Optional)

Before You Begin

Does a Listener Exist for this Availability Group Already?

To determine whether a listener already exists for the availability group

- View Availability Group Listener Properties (SQL Server)

Limitations and Restrictions

- You can create only one listener per availability group through SQL Server. Typically, each availability group requires only one listener. However, some customer scenarios require multiple listeners for one availability group. After creating a listener through SQL Server, you can use Windows PowerShell for failover clusters or the WSFC Failover Cluster Manager to create additional listeners. For more information, see To Create An Additional Listener for an Availability Group (Optional), later in this topic.

Recommendations

Using a static IP address is recommended, although not required, for multiple subnet configurations.

Prerequisites

- You must be connected to the server instance that hosts the primary replica.
- If you are setting up an availability group listener across multiple subnets and plan to use static IP addresses, you need to get the static IP address of every subnet that hosts an

availability replica for the availability group for which you are creating the listener. Usually, you will need to ask your network administrators for the static IP addresses.

Important
Before you create your first listener, we strongly recommend that you read Prerequisites, Restrictions, and Recommendations for AlwaysOn Client Connectivity (SQL Server).

Requirements for the DNS Name of an Availability Group Listener

Each availability group listener requires a DNS host name that is unique in the domain and in NetBIOS. The DNS name is a string value. This name can contain only alphanumeric characters, dashes (-), and hyphens (_), in any order. DNS host names are case insensitive. The maximum length is 63 characters, however, in SQL Server Management Studio, the maximum length you can specify is 15 characters.

We recommend that you specify a meaningful string. For example, for an availability group named `AG1`, a meaningful DNS host name would be `ag1-listener`.

Important
NetBIOS recognizes only the first 15 chars in the dns_name. If you have two WSFC clusters that are controlled by the same Active Directory and you try to create availability group listeners in both of clusters using names with more than 15 characters and an identical 15 character prefix, you will get an error reporting that the Virtual Network Name resource could not be brought online. For information about prefix naming rules for DNS names, see Assigning Domain Names.

Windows Permissions

Permissions	**Link**
The cluster object name (CNO) of WSFC cluster that is hosting the availability group must have **Create Computer objects** permission: • If the Create Cluster wizard created this WSFC cluster while running in a domain user account with **Create Computer objects** permission, cluster object name (CNO) of this WSFC cluster already has this permission. • If the account that ran the Create Cluster wizard did not have **Create Computer objects** permission, give the WSFC cluster name to your domain administrators and ask them to grant this permission to the cluster object	• *Steps for configuring the account for the person who installs the cluster* in Failover Cluster Step-by-Step Guide: Configuring Accounts in Active Directory • *Steps for prestaging the cluster name account* in Failover Cluster Step-by-Step Guide: Configuring Accounts in Active Directory

Permissions	**Link**
name (CNO) of the WSFC cluster. **Note** In some organizations, the security policy prohibits granting **Create Computer objects** permission to individual user accounts.	
If your organization requires that you prestage the computer account for a listener virtual network name, you will need membership in the **Account Operator** group or your domain administrator's assistance. **Tip** Generally, it is simplest not to prestage the computer account for a listener virtual network name. If you can, let the account to be created and configured automatically when you run the WSFC High Availability wizard.	*Steps for prestaging an account for a clustered service or application* in Failover Cluster Step-by-Step Guide: Configuring Accounts in Active Directory.

SQL Server Permissions

Task	**Permissions**
To create an availability group listener	Requires membership in the **sysadmin** fixed server role and either CREATE AVAILABILITY GROUP server permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.
To modify an existing availability group listener	Requires ALTER AVAILABILITY GROUP permission on the availability group, CONTROL AVAILABILITY GROUP permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER

Task	Permissions
	permission.

Using SQL Server Management Studio

Tip
The New Availability Group wizard supports creation of the listener for a new availability group.

To create or configure an availability group listener

1. In Object Explorer, connect to the server instance that hosts the primary replica of the availability group, and click the server name to expand the server tree.
2. Expand the **AlwaysOn High Availability** node and the **Availability Groups** node.
3. Click the availability group whose listener you want to configure, and choose one of the following alternatives:
   - To create a listener, right-click the **Availability group Listeners** node, and select the **New Listener** command. This opens the **New Availability Group Listener** dialog box. For more information, see Add Availability Group Listener (Dialog Box), later in this topic.
   - To change the port number of an existing listener, expand the **Availability group Listeners** node, right-click the listener, and select the **Properties** command. Enter the new port number into the **Port** field, and click **OK**.

New Availability Group Listener (Dialog Box)

Listener DNS Name

Specifies the DNS host name of the availability group listener. The DNS name is a string must be unique in the domain and in NetBIOS. This name can contain only alphanumeric characters, dashes (-), and hyphens (_), in any order. DNS host names are case insensitive. The maximum length is 15 characters.

For more information, see Requirements for the DNS Name of an Availability Group Listener, earlier in this topic.

Port

The TPC port used by this listener.

Network Mode

Indicates the TCP protocol used by the listener, one of:

DHCP

The listener will us a dynamic IP address that is assigned by a server running the Dynamic Host Configuration Protocol (DHCP). DHCP is limited to a single subnet.

Important

We do not recommend DHCP in production environment. If there is a down time and the DHCP IP lease expires, extra time is required to register the new DHCP network IP address that is associated with the listener DNS name and impact the client connectivity. However, DHCP is good for setting up your development and testing environment to verify basic functions of availability groups and for integration with your applications.

Static IP

The listener will use one or more static IP addresses. Additional IP addresses are optional. To create an availability group listener across multiple subnets, for each subnet you must specify a static IP address in the listener configuration. Contact your network administrator to get these static IP addresses.

If you select **Static IP** a subnet grid appears below the **Network Mode** field. This grid displays information about each subnet that can be accessed by this availability group listener. This grid is empty until you add a static IP address by clicking **Add**.

The columns are as follows:

Subnet

Displays the identifier of each subnet that you add to the availability group listener.

IP Address

Displays the IP address of a given subnet. For a given subnet, the IP address is either an IPv4 address or an IPv6 address.

Add

Click to add to add a static IP address to a selected subnet or to another subnet for this listener. This opens the **Add IP Address** dialog box. For more information, see the Add IP Address Dialog Box (SQL Server Management Studio) help topic.

Remove

Click to remove the selected subnet from this listener.

OK

Click to create the specified availability group listener.

Using Transact-SQL

To create or configure an availability group listener

1. Connect to the server instance that hosts the primary replica.
2. Use the LISTENER option of the CREATE AVAILABILITY GROUP statement or the ADD LISTENER option of the ALTER AVAILABILITY GROUP statement.

   The following example adds an availability group listener to an existing availability group named `MyAg2`. A unique DNS name, `MyAg2ListenerIvP6`, is specified for this listener. The two replicas are on different subnets, so , as recommended, the listener uses static IP addresses. For each of the two availability replicas, the WITH IP clause specifies a static IP address, `2001:4898:f0:f00f::cf3c and 2001:4898:e0:f213::4ce2`, which use the IPv6 format. This example also specifies uses the optional PORT argument to specify port `60173` as the listener port.

```
ALTER AVAILABILITY GROUP MyAg2
      ADD LISTENER ‘MyAg2ListenerIvP6’ ( WITH IP (
('2001:db88:f0:f00f::cf3c'),('2001:4898:e0:f213::4ce2') ) , PORT =
60173 );
GO
```

Using PowerShell

To create or configure an availability group listener

1. Change directory (**cd**) to the server instance that hosts the primary replica.
2. To create or modify an availability group listener use one of the following cmdlets:

New-SqlAvailabilityGroupListener

Creates a new availability group listener and attaches it to an existing availability group.

For example, the following **New-SqlAvailabilityGroupListener** command creates an availability group listener named `MyListener` for the availability group `MyAg`. This listener will use the IPv4 address passed to the **-StaticIp** parameter as its virtual IP address.

```
New-SqlAvailabilityGroupListener -Name MyListener `
-StaticIp '192.168.3.1/255.255.252.0' `
-Path SQLSERVER:\Sql\Computer\Instance\AvailabilityGroups\MyAg
```

Set-SqlAvailabilityGroupListener

Modifies the port setting on an existing availability group listener.

For example, the following **Set-SqlAvailabilityGroupListener** command sets the port number for the availability group listener named `MyListener` to `1535`. This port is used to listen for connections to the listener.

```
Set-SqlAvailabilityGroupListener -Port 1535 `
-Path
SQLSERVER:\Sql\PrimaryServer\InstanceName\AvailabilityGroups\MyAg\AGListe
ners\MyListener
```

Add-SqlAGListenerstaticIp

Adds a static IP address to an existing availability group listener configuration. The IP address can be an IPv4 address with subnet, or an IPv6 address.

For example, the following **Add-SqlAGListenerstaticIp** command adds a static IPv4 address to the availability group listener `MyListener` on the availability group `MyAg`. This IPv6 address serves as the virtual IP address of the listener on the subnet `255.255.252.0`. If the availability group spans multiple subnets, you should add a static IP address for each subnet to the listener.

```
$path =
"SQLSERVER:\SQL\PrimaryServer\InstanceName\AvailabilityGroups\MyAg\AGList
eners\ MyListener" `
Add-SqlAGListenerstaticIp -Path $path `
-StaticIp "2001:0db8:85a3:0000:0000:8a2e:0370:7334"
```

Note

To view the syntax of a cmdlet, use the **Get-Help** cmdlet in the SQL Server PowerShell environment. For more information, see Get Help SQL Server PowerShell.

To set up and use the SQL Server PowerShell provider

- SQL Server PowerShell Provider

Follow Up: After Creating an Availability Group Listener

Recommendations: After you define an availability group listener, we strongly recommend that you do the following:

- Ask your network administrator to reserve the listener's IP address for its exclusive use.
- Give the listener's DNS host name to application developers to use in connection strings when requesting client connections to this availability group.

Validate a Configuration Wizard issues an incorrect warning: 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:<network_name>' is set to 1 For the current cluster configuration this value should be set to 0."

Please ignore this message.

To Create An Additional Listener for an Availability Group (Optional)

After you create one listener through SQL Server, you can add an additional listener, as follows:

1. Create the listener using either of the following tools:
   - **Using WSFC Failover Cluster Manager:**
     i. Add a client access point and configure the IP address.
     ii. Bring the listener online.
     iii. Add a dependency to the WSFC availability group resource.

For information about the dialog boxes and tabs of the Failover Cluster Manager, see User Interface: The Failover Cluster Manager Snap-In.

- **Using Windows PowerShell for failover clusters:**
    1. Use Add-ClusterResource to create a network name and the IP address resources.
    2. Use Start-ClusterResource to start the network name resource.
    3. Use Add-ClusterResourceDependency to set the dependency between the network name and the existing SQL Server Availability Group resource.

    For information about using Windows PowerShell for failover clusters, see Overview of Server Manager Commands.

2. Start SQL Server listening on the new listener. After creating the additional listener, connect to the instance of SQL Server that hosts the primary replica of the availability group and use SQL Server Management Studio, Transact-SQL, or PowerShell to modify the listener port.

For more information, see How to create multiple listeners for same availability group (a SQL Server AlwaysOn team blog).

Related Tasks

- View Availability Group Listener Properties (SQL Server)
- Remove an Availability Group Listener (SQL Server)

Related Content

- Microsoft SQL Server AlwaysOn Solutions Guide for High Availability and Disaster Recovery
- SQL Server AlwaysOn Team Blog: The official SQL Server AlwaysOn team blog

See Also

AlwaysOn Availability Groups

Client Connectivity and Application Failover (AlwaysOn Availability Groups)

Add IP Address Dialog Box (SQL Server Management Studio)

This F1 help topic describes the options of the **Add IP Address** dialog box. This dialog box accessed from the **New Availability Group Listener** dialog box and the **Listener** tab of the **Specify Replicas** page of the New Availability Group Wizard or the Add Replica to Availability Group Wizard of SQL Server 2012.

Prerequisites

Before you begin to add subnets to an availability group listener, ensure that know the IP address for each subnet and, for an IPv4 address, the subnet mask.

Add IP Address Options

Subnet

Use the drop list to select an address for the subnet that you are adding to the availability group listener. By default a subnet possesses both an IPv4 address and an IPv6 address. The first time you use the **Add IP Address** dialog, the **Subnet** drop list displays both subnet addresses for each subnet that hosts a replica for the availability group. To add a given subnet to the listener, select one of its subnet addresses.

After you complete the **Add IP Address** dialog box and click **OK** to add a selected subnet address to the listener, the **Subnet** drop list filters out that subnet address. All unselected subnet addresses remain on the drop list. Be sure that you add one and only one subnet address per subnet to the listener, or listener creation will fail.

Addresses

Use this field to enter a static IP address for the selected subnet address. Contact your network administrator for this IP address. Ensure that you enter a valid address for the selected subnet address, or listener creation will fail.

IPv4 Address

If you selected the IPv4 subnet address of a subnet, enter a valid IPv4 static address here.

Subnet Mask

For an IPv4 address, this read-only field displays the subnet mask of the selected subnet.

IPv6 Address

If you selected the IPv6 subnet address of a subnet, enter a valid IPv6 static address here.

OK

Click to create add the subnet whose address you selected, along with the static IP address that you specified. A row containing these values will be added to the subnet grid of the **New Availability Group Listener** or **Specify Replicas** dialog box.

Important
The **Add IP Address** dialog does not verify the IP address. Also the dialog does not prevent you from adding the second subnet address for a subnet that you have already added to the availability group listener.

Cancel

Click to cancel your selections, and return to the **New Availability Group Listener** dialog box or **Listener** tab without adding a static IP address for any subnet.

Related Tasks

- Create or Configure an Availability Group Listener (SQL Server)
- Use the New Availability Group Wizard
- Use the Add Replica to Availability Group Wizard

See Also

AlwaysOn Availability Groups

Client Connectivity and Application Failover (AlwaysOn Availability Groups)

Prerequisites, Restrictions, and Recommendations for AlwaysOn Client Connectivity (SQL Server)

Configure Read-Only Routing for an Availability Group

To configure an AlwaysOn availability group to support read-only routing in SQL Server 2012, you can use either Transact-SQL or PowerShell. *Read-only routing* refers to the ability of SQL Server to route qualifying read-only connection requests to an available AlwaysOn readable secondary replica (that is, a replica that is configured to allow read-only workloads when running under the secondary role). To support read-only routing, the availability group must possess an availability group listener. Read-only clients must direct their connection requests to this listener, and the client's connection strings must specify the application intent as "read-only." That is, they must be *read-intent connection requests*.

Note
For information about how to configure a readable secondary replica, see Configure Read-Only Access on an Availability Replica (SQL Server).

- **Before you begin:**

  Prerequisites

  What Replica Properties Do you Need to Configure to Support Read-Only Routing?

  Security

- **To Configure read-only routing, using:**

  Transact-SQL

  PowerShell

  Note
  Configuring read-only routing is not supported by SQL Server Management Studio.

- **Follow Up:** After Configuring Read-Only Routing
- Related Tasks
- Related Content

Before You Begin

Prerequisites

- The availability group must possess an availability group listener. For more information, see Create or Configure an Availability Group Listener.
- One or more availability replicas must be configured to accept read-only in the secondary role (that is, to be readable secondary replicas). For more information, see Configure Connection Access on an Availability Replica (SQL Server).
- You must be connected to the server instance that hosts the current primary replica.

What Replica Properties Do you Need to Configure to Support Read-Only Routing?

- For each readable secondary replica that is to support read-only routing, you need to specify a *read-only routing URL*. This URL takes effect only when the local replica is running under the secondary role. The read-only routing URL must be specified on a replica-by-replica basis, as needed. Each read-only routing URL is used for routing read-intent connection requests to a specific readable secondary replica. Typically, every readable secondary replica is assigned a read-only routing URL.
- For each availability replica that you want to support read-only routing when it is the primary replica, you need to specify a *read-only routing list*. A given read-only routing list takes effect only when the local replica is running under the primary role. This list must be specified on a replica-by-replica basis, as needed. Typically, each read-only routing list would contain every read-only routing URL, with the URL of the local replica at the end of the list.

  Note
  Read-intent connection requests are routed to the first available readable secondary on the read-only routing list of the current primary replica. There is no load balancing.

Note
For information about availability group listeners and more information about read-only routing, see Availability Group Listeners, Client Connectivity, and Application Failover (AlwaysOn Availability Groups).

Security

Permissions

Task	Permissions
To configure replicas when creating an availability group	Requires membership in the **sysadmin** fixed server role and either CREATE AVAILABILITY GROUP server permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.
To modify an availability replica	Requires ALTER AVAILABILITY GROUP permission on the availability group, CONTROL AVAILABILITY GROUP permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.

Using Transact-SQL

To Configure read-only routing

Note
For a code example, see Example (Transact-SQL), later in this section.

1. Connect to the server instance that hosts the primary replica.
2. If you are specifying a replica for a new availability group, use the CREATE AVAILABILITY GROUP Transact-SQL statement. If you are adding or modifying a replica for an existing availability group, use the ALTER AVAILABILITY GROUP Transact-SQL statement.
   - To configure read-only routing for the secondary role, in the ADD REPLICA or MODIFY REPLICA WITH clause, specify the SECONDARY_ROLE option, as follows:

     SECONDARY_ROLE **(** READ_ONLY_ROUTING_URL **=** 'TCP**://**system-address**:**port' **)**

     The parameters of the read-only routing URL are as follows:

     system-address

     Is a string, such as a system name, a fully qualified domain name, or an IP address, that unambiguously identifies the destination computer system.

     port

     Is a port number that is used by the Database Engine of the SQL Server instance.

     For example: `SECONDARY_ROLE (READ_ONLY_ROUTING_URL = N'TCP://COMPUTER01.contoso.com:1433')`

     In a MODIFY REPLICA clause the ALLOW_CONNECTIONS is optional if the replica is already configured to allow read-only connections.
   - To configure read-only routing for the primary role, in the ADD REPLICA or MODIFY REPLICA WITH clause, specify the PRIMARY_ROLE option, as follows:

     PRIMARY_ROLE **(** READ_ONLY_ROUTING_LIST **=** **(** 'server' [**,**...n] **))**

     where, server identifies a server instance that hosts a read-only secondary replica in the availability group.

     For example: `PRIMARY_ROLE (READ_ONLY_ROUTING_LIST=('`*Server1*`','`*Server2*`'))`

     Note
     You must set the read-only routing URL before configuring the read-only routing list.

Example (Transact-SQL)

The following example modifies two availability replicas of an existing availability group, `AG1` to support read-only routing if one of these replicas currently owns the primary role. To identify the server instances that host the availability replica, this example specifies the instance names—`COMPUTER01` and `COMPUTER02`.

```
ALTER AVAILABILITY GROUP [AG1]

 MODIFY REPLICA ON
```

```
N'COMPUTER01' WITH
(SECONDARY_ROLE (ALLOW_CONNECTIONS = READ_ONLY));
ALTER AVAILABILITY GROUP [AG1]
 MODIFY REPLICA ON
N'COMPUTER01' WITH
(SECONDARY_ROLE (READ_ONLY_ROUTING_URL =
N'TCP://COMPUTER01.contoso.com:1433'));

ALTER AVAILABILITY GROUP [AG1]
 MODIFY REPLICA ON
N'COMPUTER02' WITH
(SECONDARY_ROLE (ALLOW_CONNECTIONS = READ_ONLY));
ALTER AVAILABILITY GROUP [AG1]
 MODIFY REPLICA ON
N'COMPUTER02' WITH
(SECONDARY_ROLE (READ_ONLY_ROUTING_URL =
N'TCP://COMPUTER02.contoso.com:1433'));

ALTER AVAILABILITY GROUP [AG1]
MODIFY REPLICA ON
N'COMPUTER01' WITH
(PRIMARY_ROLE (READ_ONLY_ROUTING_LIST=('COMPUTER02','COMPUTER01')));

ALTER AVAILABILITY GROUP [AG1]
MODIFY REPLICA ON
N'COMPUTER02' WITH
(PRIMARY_ROLE (READ_ONLY_ROUTING_LIST=('COMPUTER01','COMPUTER02')));
GO
```

Using PowerShell

To Configure read-only routing

Note
For a code example, see Example (PowerShell), later in this section.

1. Set default (**cd**) to the server instance that hosts the primary replica.

2. When adding an availability replica to an availability group, use the **New-SqlAvailabilityReplica** cmdlet. When modifying an existing availability replica, use the **Set-SqlAvailabilityReplica** cmdlet. The relevant parameters are as follows:

   - To configure read-only routing for the secondary role, specify the **ReadonlyRoutingConnectionUrl "**url**"** parameter.

     where, url is the connectivity fully-qualified domain name (FQDN) and port to use when routing to the replica for read-only connections. For example: `-ReadonlyRoutingConnectionUrl "TCP://DBSERVER8.manufacturing.Adventure-Works.com:7024"`

   - To configure connection access for the primary role, specify **ReadonlyRoutingList "**server**"** [**,**...n], where server identifies a server instance that hosts a read-only secondary replica in the availability group. For example: `-ReadOnlyRoutingList "SecondaryServer","PrimaryServer"`

     Note
     You must set the read-only routing URL of a replica before configuring its read-only routing list.

   Note
   To view the syntax of a cmdlet, use the **Get-Help** cmdlet in the SQL Server PowerShell environment. For more information, see SQL Server PowerShell Help.

To set up and use the SQL Server PowerShell provider

- Using the SQL Server PowerShell Provider
- SQL Server PowerShell Help

Example (PowerShell)

The following example configures the primary replica and one secondary replica in an availability group for read-only routing. First, the example assigns a read-only routing URL to each replica. Then it sets the read-only routing list on the primary replica. Connections with the "ReadOnly" property set in the connection string will be redirected to the secondary replica. If this secondary replica is not readable (as determined by the **ConnectionModeInSecondaryRole** setting), the connection will be directed back to the primary replica.

```
Set-Location SQLSERVER:\SQL\PrimaryServer\default\AvailabilityGroups\MyAg
$primaryReplica = Get-Item "AvailabilityReplicas\PrimaryServer"
$secondaryReplica = Get-Item "AvailabilityReplicas\SecondaryServer"


Set-SqlAvailabilityReplica -ReadOnlyRoutingConnectionUrl
"TCP://PrimaryServer.domain.com:1433" -InputObject $primaryReplica
Set-SqlAvailabilityReplica -ReadOnlyRoutingConnectionUrl
"TCP://SecondaryServer.domain.com:1433" -InputObject $secondaryReplica
```

```
Set-SqlAvailabilityReplica -ReadOnlyRoutingList
"SecondaryServer","PrimaryServer" -InputObject $primaryReplica
```

Follow Up: After Configuring Read-Only Routing

Once the current primary replica and the readable secondary replicas are configured to support read-only routing in both roles, the readable secondary replicas can receive read read-intent connection requests from clients that connect via the availability group listener.

Tip
When using the bcp Utility or sqlcmd Utility, you can specify read-only access to any secondary replica that is enabled for read-only access by specifying the **-K ReadOnly** switch.

Requirements and Recommendations for Client Connection-Strings

For a client application to use read-only routing, its connection string must satisfy the following requirements:

- Use the TCP protocol.
- Set the application intent attribute/property to readonly.
- Reference the listener of an availability group that is configured to support read-only routing.
- Reference a database in that availability group.

In addition, we recommend that connection strings enable multi-subnet failover, which supports a parallel client thread for each replica on each subnet. This minimizes client reconnection time after a failover.

The syntax for a connection string depends on the SQL Server provider an application is using. The following example connection string for the .NET Framework Data Provider 4.0.2 for SQL Server illustrates the parts of a connection string that are required and recommended to work for read-only routing.

```
Server=tcp:MyAgListener,1433;Database=Db1;IntegratedSecurity=SSPI;Application
Intent=ReadOnly;MultiSubnetFailover=True
```

For more information about read-only application intent and read-only routing, see Availability Group Listeners, Client Connectivity, and Application Failover (AlwaysOn Availability Groups).

Related Tasks

To view read-only routing configurations

- sys.availability_read_only_routing_lists (Transact-SQL)
- sys.availability_replicas (Transact-SQL) (**read_only_routing_url** column)

To configure client connection access

- Create or Configure an Availability Group Listener
- Configure Connection Access on an Availability Replica (SQL Server)

To use connection strings in applications

- SQL Server Native Client Support for High Availability, Disaster Recovery
- Using Connection String Keywords with SQL Server Native Client

Related Content

- Microsoft SQL Server AlwaysOn Solutions Guide for High Availability and Disaster Recovery
- SQL Server AlwaysOn Team Blog: The official SQL Server AlwaysOn Team Blog

See Also

AlwaysOn Availability Groups (SQL Server)

Overview of AlwaysOn Availability Groups (SQL Server)

Readable Secondary Availability Replicas

Client Connection Access to Availability Replicas (SQL Server)

Availability Group Listeners, Client Connectivity, and Application Failover (AlwaysOn Availability Groups)

Overview of Transact-SQL Statements for AlwaysOn Availability Groups

This topic introduces the Transact-SQL statements that support deploying AlwaysOn Availability Groups and creating and managing an given availability group, availability replica and availability database.

In This Topic:

- CREATE ENDPOINT
- CREATE AVAILABILITY GROUP
- ALTER AVAILABILITY GROUP
- ALTER DATABASE SET HADR Options
- DROP AVAILABILITY GROUP
- Restrictions on the AVAILABILITY GROUP Transact-SQL statements

CREATE ENDPOINT

CREATE ENDPOINT ... FOR DATABASE_MIRRORING creates a database mirroring endpoint, if none exists on the server instance. Every server instance on which you intend to deploy AlwaysOn Availability Groups or database mirroring requires a database mirroring endpoint.

Execute this statement on the server instance on which you are creating the endpoint. You can create only one database mirroring endpoint on a given server instance. For more information, see Database Mirroring Endpoint.

CREATE AVAILABILITY GROUP

CREATE AVAILABILITY GROUP creates a new availability group and optionally an availability group listener. Minimally, you must specify your local server instance, which will become the initial primary replica. Optionally, you can also specify up to four secondary replicas.

Execute CREATE AVAILABILITY GROUP on the instance of SQL Server that you want to host the initial primary replica of your new availability group. This server instance must reside on a node of a Windows Server Failover Cluster (WSFC) (for more information, see Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups (SQL Server).

ALTER AVAILABILITY GROUP

ALTER AVAILABILITY GROUP supports changing an existing availability group or availability group listener and for failing over an availability group.

Execute ALTER AVAILABILITY GROUP on the instance of SQL Server that hosts the current primary replica.

ALTER DATABASE … SET HADR …

The options of the SET HADR clause of the ALTER DATABASE statement enables you to join a secondary database to the availability group of the corresponding primary database, remove a joined database, and suspend data synchronization on a joined database, and resume data synchronization.

DROP AVAILABILITY GROUP

DROP AVAILABILITY GROUP removes a specified availability group and all of its replicas. DROP AVAILABILITY GROUP can be run from any AlwaysOn Availability Groups node in the WSFC failover cluster.

Restrictions on the AVAILABILITY GROUP Transact-SQL Statements

The CREATE AVAILABILITY GROUP, ALTER AVAILABILITY GROUP, and DROP AVAILABILITY GROUP Transact-SQL statements have the following limitations:

- With the exception of DROP AVAILABILITY GROUP, executing these statements requires that the HADR service is enabled on the instance of SQL Server. For more information, see Enable and Disable AlwaysOn Availability Groups (SQL Server).
- These statements cannot be executed within transactions or batches.
- Though they make a best effort to clean up after a failure, these statements do not guarantee that they will roll back all changes on failure. However, systems should be able cleanly handle and then ignore partial failures.
- These statements do not support expressions or variables.
- If a Transact-SQL statement is executed while another availability group action or recovery is in process, the statement returns an error. Wait for the action or recovery to complete, and retry the statement, if necessary.

See Also

Overview of AlwaysOn Availability Groups (SQL Server)

Overview of PowerShell Cmdlets for AlwaysOn Availability Groups

Microsoft PowerShell is a task-based command-line shell and scripting language designed especially for system administration. AlwaysOn Availability Groups provides a set of PowerShell cmdlets in SQL Server 2012 that enable you to deploy, manage, and monitor availability groups, availability replicas, and availability databases.

Note
A PowerShell cmdlet can complete by successfully initiating an action. This does not indicate that the intended work, such as the fail over of an availability group, has completed. When scripting a sequence of actions, you might have to check the status of actions, and wait for them to complete.

This topic introduces the cmdlets for the following sets of tasks:

- Configuring a server instance for AlwaysOn Availability Groups
- Backing up and restoring databases and transaction logs
- Creating and managing an availability group
- Creating and managing an availability group listener
- Creating and managing an availability replica
- Adding and managing an availability database
- Monitoring availability group health

Note
For a list of topics in SQL Server 2012 Books Online that describe how to use cmdlets to perform AlwaysOn Availability Groups tasks, see the "Related Tasks" section of Overview of AlwaysOn Availability Groups (SQL Server).

Configuring a Server Instance for AlwaysOn Availability Groups

Cmdlets	Description	Supported on
Disable-SqlAlwaysOn	Disables the AlwaysOn Availability Groups feature on a server instance.	The server instance that is specified by the **Path**, **InputObject**, or **Name** parameter. (Must be an edition of SQL Server 2012 that supports AlwaysOn Availability Groups.)
Enable-SqlAlwaysOn	Enables AlwaysOn Availability Groups on an instance of SQL Server 2012 that supports the AlwaysOn Availability Groups feature. For information about	Any edition of SQL Server 2012 that supports AlwaysOn Availability Groups.

Cmdlets	Description	Supported on
	support for AlwaysOn Availability Groups, see Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups (SQL Server).	
New-SqlHadrEndPoint	Creates a new database mirroring endpoint on a server instance. This endpoint is required for data movement between primary and secondary databases.	Any instance of SQL Server
Set-SqlHadrEndpoint	Changes the properties of an existing database mirroring endpoint, such as the name, state, or authentication properties.	A server instance that supports AlwaysOn Availability Groups and lacks a database mirroring endpoint

Backing Up and Restoring Databases and Transaction Logs

Cmdlets	Description	Supported on
Backup-SqlDatabase	Creates a data or log backup.	Any online database (for AlwaysOn Availability Groups, a database on the server instance that hosts the primary replica)
Restore-SqlDatabase	Restores a backup.	Any instance of SQL Server (for AlwaysOn Availability Groups, a server instance that hosts a secondary replica) **Important** When preparing a secondary database, you must use the **-NoRecovery** parameter in every **Restore-SqlDatabase** command.

For information about using these cmdlets to prepare a secondary database, see Manually Prepare a Secondary Database for an Availability Group (SQL Server).

Creating and Managing an Availability Group

Cmdlets	Description	Supported on
New-SqlAvailabilityGroup	Creates a new availability group.	Server instance to host primary replica
Remove-SqlAvailabilityGroup	Deletes availability group.	HADR-enabled server instance
Set-SqlAvailabilityGroup	Sets the properties of an availability group; take an availability group online/offline	Server instance that hosts primary replica
Switch-SqlAvailabilityGroup	Initiates one of the following forms of failover: • A forced failover of an availability group (with possible data loss). • A manual failover of an availability group.	Server instance that hosts target secondary replica

Creating and Managing an Availability Group Listener

Cmdlet	Description	Supported on
New-SqlAvailabilityGroupListener	Creates a new availability group listener and attaches it to an existing availability group.	Server instance that hosts primary replica
Set-SqlAvailabilityGroupListener	Modifies the port setting on an existing availability group listener.	Server instance that hosts primary replica
Add-SqlAvailabilityGroupListenerStaticIp	Adds a static IP address to an existing availability	Server instance that hosts primary replica

Cmdlet	Description	Supported on
	group listener configuration. The IP address can be an IPv4 address with subnet, or an IPv6 address.	

Creating and Managing an Availability Replica

Cmdlets	Description	Supported on
New-SqlAvailabilityReplica	Creates a new availability replica. You can Use the **-AsTemplate** parameter to create an in-memory availability-replica object for each new availability replica.	Server instance that hosts primary replica
Join-SqlAvailabilityGroup	Joins a secondary replica to the availability group.	Server instance that hosts secondary replica
Remove-SqlAvailabilityReplica	Deletes an availability replica.	Server instance that hosts primary replica
Set-SqlAvailabilityReplica	Sets the properties of an availability replica.	Server instance that hosts primary replica

Adding and Managing an Availability Database

Cmdlets	Description	Supported on
Add-SqlAvailabilityDatabase	• On the primary replica, adds a database to an availability group. • On a secondary replica, joins a secondary database to an availability group.	Any server instance that hosts an availability replica (behavior differs for primary and secondary replicas)

Cmdlets	Description	Supported on
Remove-SqlAvailabilityDatabase	• On the primary replica, removes the database from the availability group. • On a secondary replica, removes the local secondary database from the local secondary replica.	Any server instance that hosts an availability replica (behavior differs for primary and secondary replicas)
Resume-SqlAvailabilityDatabase	Resumes the data movement for a suspended availability database.	The server instance on which the database was suspended.
Suspend-SqlAvailabilityDatabase	Suspends the data movement for an availability database.	Any server instance that hosts an availability replica.

Monitoring Availability Group Health

The following SQL Server cmdlets enable you to monitor the health of an availability group and its replicas and databases.

noteDXDOC112778PADS Security Note

You must have CONNECT, VIEW SERVER STATE, and VIEW ANY DEFINITION permissions to execute these cmdlets.

Cmdlet	Description	Supported on
Test-SqlAvailabilityGroup	Assesses the health of an availability group by evaluating SQL Server policy based management (PBM) policies.	Any server instance that hosts an availability replica.*
Test-SqlAvailabilityReplica	Assesses the health of availability replicas by evaluating SQL Server policy based management (PBM) policies.	Any server instance that hosts an availability replica.*

Cmdlet	Description	Supported on
Test-SqlDatabaseReplicaState	Assesses the health of an availability database on all joined availability replicas by evaluating SQL Server policy based management (PBM) policies.	Any server instance that hosts an availability replica.*

* To view information about all of the availability replicas in an availability group, use to the server instance that hosts the primary replica.

For more information, see Use Policy-Based Management to Monitor an Availability Group (SQL Server).

See Also

Overview of AlwaysOn Availability Groups (SQL Server)

SQL Server PowerShell Help

Configuration of a Server Instance for AlwaysOn Availability Groups

This topic contains information about the requirements for configuring an instance of SQL Server to support SQL Server 2012.

Important
For essential information about AlwaysOn Availability Groups prerequisites and restrictions for Windows Server Failover Clustering (WSFC) nodes and for instances of SQL Server, see Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups (SQL Server).

In this Topic:

- Terms and Definitions
- To Configure a Server Instance to Support AlwaysOn Availability Groups
- Related Tasks
- Related Content

Terms and Definitions

AlwaysOn Availability Groups

A high-availability and disaster-recovery solution that provides an enterprise-level replacement for database mirroring. An *availability group* supports a failover environment for a discrete set of user databases, known as *availability databases*, that fail over together.

availability replica

An instantiation of an availability group that is hosted by a specific instance of SQL Server and that maintains a local copy of each availability database that belongs to the availability group. Two types of availability replicas exist: a single *primary replica* and one to four *secondary replicas*. The server instances that host the availability replicas for a given availability group must reside on different nodes of a single Windows Server Failover Clustering (WSFC) cluster.

database mirroring endpoint

An endpoint is a SQL Server object that enables SQL Server to communicate over the network. To participate in database mirroring and/or AlwaysOn Availability Groups a server instance requires a special, dedicated endpoint. All mirroring and availability group connections on a server instance use the same database mirroring endpoint. This endpoint is a special-purpose endpoint used exclusively to receive these connections from other server instances.

To Configure a Server Instance to Support AlwaysOn Availability Groups

To support AlwaysOn Availability Groups, a server instance must reside on a node in the WSFC failover cluster that hosts the availability group, be AlwaysOn Availability Groups enabled, and possess a database mirroring endpoint.

1. Enable the AlwaysOn Availability Groups feature on every server instance that is to participate in one or more availability groups. A given server instance can host only a single availability replica for a given availability group.
2. Ensure that the server instance possesses a database mirroring endpoint.

Related Tasks

To enable AlwaysOn Availability Groups

- Enable and Disable the AlwaysOn Availability Groups Feature (SQL Server)

To determine whether a database mirroring endpoint exists

- sys.database_mirroring_endpoints (Transact-SQL)

To create a database mirroring endpoint

- Create a Database Mirroring Endpoint for AlwaysOn Availability Groups (SQL Server PowerShell)
- Create a Mirroring Endpoint for Windows Authentication (Transact-SQL)
- Allow Database Mirroring to Use Certificates for Outbound Connections (Transact-SQL)

Related Content

- Microsoft SQL Server AlwaysOn Solutions Guide for High Availability and Disaster Recovery
- SQL Server AlwaysOn Team Blog: The official SQL Server AlwaysOn Team Blog

See Also

Overview of AlwaysOn Availability Groups

Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups (SQL Server)

The Database Mirroring Endpoint (SQL Server)

AlwaysOn Availability Groups: Interoperability and Coexistence (SQL Server)

Failover Clustering and AlwaysOn Availability Groups (SQL Server)

Windows Server Failover Clustering (WSFC) with SQL Server

AlwaysOn Failover Cluster Instances (FCI)

Enable and Disable AlwaysOn Availability Groups

Enabling AlwaysOn Availability Groups is a prerequisite for a server instance to use availability groups. Before you can create and configure any availability group, the AlwaysOn Availability Groups feature must have been enabled on the each instance of SQL Server that will host an availability replica for one or more availability groups.

Important
If you delete and re-create a WSFC cluster, you must disable and re-enable the AlwaysOn Availability Groups feature on each instance of SQL Server that hosted an availability replica on the original WSFC cluster.

- **Before you begin:**

  Prerequisites

  Security
- **How To:**
  - Determine Whether AlwaysOn Availability Groups is Enabled
  - Enable AlwaysOn Availability Groups
  - Disable AlwaysOn Availability Groups

Before You Begin

Prerequisites for Enabling AlwaysOn Availability Groups

- The server instance must reside on a Windows Server Failover Clustering (WSFC) node.
- The server instance must be running an edition of SQL Server that supports AlwaysOn Availability Groups. For more information, see Features Supported by the Editions of SQL Server "Denali".

- Enable AlwaysOn Availability Groups on only one server instance at a time. After enabling AlwaysOn Availability Groups, wait until the SQL Server service has restarted before you proceed to another server instance.

For information about additional prerequisites for creating and configuring availability groups, see Prerequisites, Restrictions, and Recommendations (AlwaysOn Availability Groups).

Security

While AlwaysOn Availability Groups is enabled on an instance of SQL Server, the server instance has full control on the WSFC cluster.

Permissions

Requires membership in the **Administrator** group on the local computer and full control on the WSFC cluster. When enabling AlwaysOn by using PowerShell, open the Command Prompt window using the **Run as administrator** option.

Requires Active Directory Create Objects and Manage Objects permissions.

Determine Whether AlwaysOn Availability Groups is Enabled

- SQL Server Management Studio
- Transact-SQL
- PowerShell

Using SQL Server Management Studio

To determine whether AlwaysOn Availability Groups is enabled

1. In Object Explorer, right-click the server instance, and click **Properties**.
2. In the **Server Properties** dialog box, click the **General** page. The **Is HADR Enabled** property displays one of the following values:
    - **True**, if AlwaysOn Availability Groups is enabled
    - **False**, if AlwaysOn Availability Groups is disabled.

Using Transact-SQL

To determine whether AlwaysOn Availability Groups is enabled

1. Use the following SERVERPROPERTY statement:

    ```
    SELECT SERVERPROPERTY ('IsHadrEnabled');
    ```

    The setting of the **IsHadrEnabled** server property indicates whether an instance of SQL Server is enabled for AlwaysOn Availability Groups, as follows:

    - If **IsHadrEnabled** = 1, AlwaysOn Availability Groups is enabled.
    - If **IsHadrEnabled** = 0, AlwaysOn Availability Groups is disabled.

    Note
    For more information about the **IsHadrEnabled** server property, see SERVERPROPERTY (Transact-SQL).

Using PowerShell

To determine whether AlwaysOn Availability Groups is enabled

1. Set default (**cd**) to the server instance on which you want to determine whether AlwaysOn Availability Groups is enabled.
2. Enter the following PowerShell **Get-Item** command:

```
PS SQLSERVER:\SQL\NODE1\DEFAULT> get-item . | select IsHadrEnabled
```

Note
To view the syntax of a cmdlet, use the **Get-Help** cmdlet in the SQL Server PowerShell environment. For more information, see SQL Server PowerShell Help.

To set up and use the SQL Server PowerShell provider

- SQL Server PowerShell Help

Enable AlwaysOn Availability Groups

To enable AlwaysOn, using:

- SQL Server Configuration Manager
- PowerShell

Using SQL Server Configuration Manager

To enable AlwaysOn Availability Groups

1. Connect to the Windows Server Failover Clustering (WSFC) node that hosts the SQL Server instance where you want to enable AlwaysOn Availability Groups.
2. On the **Start** menu, point to **All Programs**, point to , point to **Configuration Tools**, and click **SQL Server Configuration Manager**.
3. In **SQL Server Configuration Manager**, click **SQL Server Services**, right-click SQL Server (**<**instance name**>)**, where **<**instance name**>** is the name of a local server instance for which you want to enable AlwaysOn Availability Groups, and click **Properties.**
4. Select the **AlwaysOn High Availability** tab.
5. Verify that **Windows failover cluster name** field contains the name of the local failover cluster node. If this field is blank, this server instance currently does not support AlwaysOn Availability Groups. Either the local computer is not a cluster node, the WSFC cluster has been shut down, or this edition of SQL Server 2012 that does not support AlwaysOn Availability Groups.
6. Select the **Enable AlwaysOn Availability Groups** check box, and click **OK**.

   SQL Server Configuration Manager saves your change. Then, you must manually restart the SQL Server service. This enables you to choose a restart time that is best for your business requirements. When the SQL Server service restarts, AlwaysOn will be enabled, and the **IsHadrEnabled** server property will be set to 1.

Using SQL Server PowerShell

To enable AlwaysOn

1. Change directory (**cd**) to a server instance that you want to enable for AlwaysOn Availability Groups.
2. Use the **Enable-SqlAlwaysOn** cmdlet to enable AlwaysOn Availability Groups.

   To view the syntax of a cmdlet, use the **Get-Help** cmdlet in the SQL Server PowerShell environment. For more information, see Get Help SQL Server PowerShell.

   Note
   For information about how to control whether the **Enable-SqlAlwaysOn** cmdlet restarts the SQL Server service, see When Does a Cmdlet Restart the SQL Server Service?, later in this topic.

To set up and use the SQL Server PowerShell provider

- SQL Server PowerShell Provider

Example: Enable-SqlAlwaysOn

The following PowerShell command enables AlwaysOn Availability Groups on an instance of SQL Server (*Computer*\Instance).

```
Enable-SqlAlwaysOn -Path SQLSERVER:\SQL\Computer\Instance
```

Disable AlwaysOn Availability Groups

- **Before you disable AlwaysOn:**

  Recommendations
- **To disable AlwaysOn, using:**
  - SQL Server Configuration Manager
  - PowerShell
- **Follow Up:** After Disabling AlwaysOn

Important
Disable AlwaysOn on only one server instance at a time. After disabling AlwaysOn Availability Groups, wait until the SQL Server service has restarted before you proceed to another server instance.

Recommendations

Before you disable AlwaysOn on a server instance, we recommend that you do the following:

1. If the server instance is currently hosting the primary replica of an availability group that you want to keep, we recommend that you manually fail over the availability group to a synchronized secondary replica, if possible. For more information, see Perform a Planned Manual Failover of an Availability Group (SQL Server).

2. Remove all local secondary replicas. For more information, see Remove a Secondary Replica from an Availability Group (SQL Server).

Using SQL Server Configuration Manager

To disable AlwaysOn

1. Connect to the Windows Server Failover Clustering (WSFC) node that hosts the SQL Server instance where you want to disable AlwaysOn Availability Groups.
2. On the **Start** menu, point to **All Programs**, point to , point to **Configuration Tools**, and click **SQL Server Configuration Manager**.
3. In **SQL Server Configuration Manager**, click **SQL Server Services**, right-click SQL Server (**<**instance name**>)**, where **<**instance name**>** is the name of a local server instance for which you want to disable AlwaysOn Availability Groups, and click **Properties**.
4. On the **AlwaysOn High Availability** tab, deselect the **Enable AlwaysOn Availability Groups** check box, and click **OK**.

   SQL Server Configuration Manager saves your change and restarts the SQL Server service. When the SQL Server service restarts, AlwaysOn will be disabled, and the **IsHadrEnabled** server property will be set to 0, to indicate that AlwaysOn Availability Groups is disabled.
5. We recommend that you read the information in Follow Up: After Disabling AlwaysOn, later in this topic.

Using SQL Server PowerShell

To disable AlwaysOn

1. Change directory (**cd**) to a currently-enabled server instance that that you want to disenable for AlwaysOn Availability Groups.
2. Use the **Disable-SqlAlwaysOn** cmdlet to enable AlwaysOn Availability Groups.

   For example, the following command disables AlwaysOn Availability Groups on an instance of SQL Server (*Computer\Instance*). This command requires restarting the instance, and you will be prompted to confirm this restart.

   ```
   Disable-SqlAlwaysOn -Path SQLSERVER:\SQL\Computer\Instance
   ```

   Important
   For information about how to control whether the **Disable-SqlAlwaysOn** cmdlet restarts the SQL Server service, see When Does a Cmdlet Restart the SQL Server Service?, later in this topic.

   To view the syntax of a cmdlet, use the **Get-Help** cmdlet in the SQL Server PowerShell environment. For more information, see Get Help SQL Server PowerShell.

To set up and use the SQL Server PowerShell provider

- SQL Server PowerShell Provider

Follow Up: After Disabling AlwaysOn

After you disable AlwaysOn Availability Groups, the instance of SQL Server must be restarted. SQL Configuration Manager restarts the server instance automatically. However, if you used the **Disable-SqlAlwaysOn** cmdlet, you will need to restart the server instance manually. For more information, see sqlservr Application.

On the restarted server instance:

- Availability databases do not start up at SQL Server startup, making them inaccessible.
- The only supported AlwaysOn Transact-SQL statement is DROP AVAILABILITY GROUP. CREATE AVAILABILITY GROUP, ALTER AVAILABILITY GROUP, and the SET HADR options of ALTER DATABASE are not supported.
- SQL Server metadata and AlwaysOn Availability Groups configuration data in WSFC are unaffected by disabling AlwaysOn Availability Groups.

If you permanently disable AlwaysOn Availability Groups on every server instance that hosts an availability replica for one or more availability groups, we recommend that you complete the following steps:

1. If you did not remove the local availability replicas before disabling AlwaysOn, delete (drop) each availability group for which the server instance is hosting an availability replica. For information about deleting an availability group, see Remove an Availability Group (AlwaysOn Availability Groups).
2. To remove the metadata left behind, delete (drop) each affected availability group on a server instance that is part of the original WSFC cluster.
3. Any primary databases continue to be accessible to all connections but the data synchronization between the primary and secondary databases stops.
4. The secondary databases enter the RESTORING state. You can delete them, or you can restore them by using RESTORE WITH RECOVERY. However, restored databases are no longer participating in availability-group data synchronization.

When Does a Cmdlet Restart the SQL Server Service?

On a server instance that is currently running, using **Enable-SqlAlwaysOn** or **Disable-SqlAlwaysOn** to change the current AlwaysOn setting could cause the SQL Server service to restart. The restart behavior on depends on the following conditions:

-NoServiceRestart parameter specified	-Force parameter specified	Is the SQL Server service restarted?
No	No	By default. But the cmdlet prompts you as follows: **To complete this action, we must restart the SQL Server service for server instance '<instance_name>'. Do you want to continue?**

-NoServiceRestart parameter specified	-Force parameter specified	Is the SQL Server service restarted?
		[Y] Yes [N] No [S] Suspend [?] Help (default is "Y"): If you specify **N** or **S**, the service is not restarted.
No	Yes	Service is restarted.
Yes	No	Service is not restarted.
Yes	Yes	Service is not restarted.

See Also

AlwaysOn Availability Groups (SQL Server)

SERVERPROPERTY (Transact-SQL)

Create a Database Mirroring Endpoint for AlwaysOn Availability Groups (SQL Server PowerShell)

This topic describes how to create a database mirroring endpoint for use by AlwaysOn Availability Groups in SQL Server 2012 by using PowerShell.

In This Topic

- **Before you begin:** Security
- **To create a database mirroring endpoint, using:** PowerShell

Before You Begin

Security

noteDXDOC112778PADS Security Note

The RC4 algorithm is deprecated. This feature will be removed in a future version of Microsoft SQL Server. Do not use this feature in new development work, and modify applications that currently use this feature as soon as possible. We recommend that you use AES.

Permissions

Requires CREATE ENDPOINT permission, or membership in the sysadmin fixed server role. For more information, see GRANT Endpoint Permissions (Transact-SQL).

Using PowerShell

To create a database mirroring endpoint

1. Change directory (**cd**) to the server instance for which you want to create the database mirroring endpoint.
2. Use the **New-SqlHadrEndpoint** cmdlet to create the endpoint and then use the **Set-SqlHadrEndpoint** to start the endpoint.

Example (PowerShell)

The following PowerShell commands create a database mirroring endpoint on an instance of SQL Server (*Machine**Instance*). The endpoint uses port 5022.

Important
This example works only on a server instance that currently lack a database mirroring endpoint.

```
# Create the endpoint.
$endpoint = New-SqlHadrEndpoint MyMirroringEndpoint -Port 5022 -Path
SQLSERVER:\SQL\Machine\Instance


# Start the endpoint
Set-SqlHadrEndpoint -InputObject $endpoint -State "Started"
```

Related Tasks

To Configure a Database Mirroring Endpoint

- Create a Mirroring Endpoint for Windows Authentication (Transact-SQL)
- Use Certificates for a Database Mirroring Endpoint
  - Allow Database Mirroring to Use Certificates for Outbound Connections (Transact-SQL)
  - Allow a Database Mirroring Endpoint to Use Certificates for Inbound Connections (Transact-SQL)
- Specify a Server Network Address (Database Mirroring)
- Specify the Endpoint URL When Adding or Modifying an Availability Replica (SQL Server)

To View Information About the Database Mirroring Endpoint

- sys.database_mirroring_endpoints (Transact-SQL)

See Also

Create an Availability Group (Transact-SQL)

AlwaysOn Availability Groups

Troubleshoot AlwaysOn Availability Groups Configuration

This topic provides information to help you troubleshoot typical problems with configuring server instances for AlwaysOn Availability Groups. Typical configuration problems include AlwaysOn Availability Groups is disabled, accounts are incorrectly configured, the database mirroring endpoint does not exist, the endpoint is inaccessible (SQL Server Error 1418), network access does not exist, and a join database command fails (SQL Server Error 35250).

Note

Ensure that you are meeting the AlwaysOn Availability Groups prerequisites. For more information, see Prerequisites, Restrictions, and Recommendations (AlwaysOn Availability Groups).

Issue	Summary
AlwaysOn Availability Groups Is Not Enabled	If an instance of SQL Server is not enabled for AlwaysOn Availability Groups, the instance does not support availability group creation and cannot host any availability replicas.
Accounts	Discusses requirements for correctly configuring the accounts under which SQL Server is running.
Endpoints	Discusses how to diagnose issues with the database mirroring endpoint of a server instance.
System name	Summarizes the alternatives for specifying the system name of a server instance in an endpoint URL.
Network access	Documents the requirement that each server instance that is hosting an availability replica must be able to access the port of each of the other server instances over TCP.
Endpoint Access (SQL Server Error 1418)	Contains information about this SQL Server error message.
Join Database Fails (SQL Server Error 35250)	Discusses the possible causes and resolution of a failure to join secondary databases to an availability group because the connection to the primary replica is not

Issue	Summary
	active.

AlwaysOn Availability Groups Is Not Enabled

The AlwaysOn Availability Groups feature must be enabled on each of the instances of SQL Server 2012. For more information, see Enabling and Disabling the AlwaysOn Availability Groups Feature (SQL Server).

Accounts

The accounts under which SQL Server is running must be correctly configured.

1. Do the accounts have the correct permissions?
   a. If the partners run as the same domain user account, the correct user logins exist automatically in both **master** databases. This simplifies the security configuration the database and is recommended.
   b. If two server instances run as different accounts, the login each account must be created in **master** on the remote server instance, and that login must be granted CONNECT permissions to connect to the database mirroring endpoint of that server instance. For more information, see Setting Up Login Accounts for Database Mirroring.
2. If SQL Server is running as a built-in account, such as Local System, Local Service, or Network Service, or a nondomain account, you must use certificates for endpoint authentication. If your service accounts are using domain accounts in the same domain, you can choose to grant CONNECT access for each service account on all the replica locations or you can use certificates. For more information, see Using Certificates for Database Mirroring.

Endpoints

Endpoints must be correctly configured.

1. Make sure that each instance of SQL Server that is going to host an availability replica (each *replica location*) has a database mirroring endpoint. To determine whether a database mirroring endpoint exists on a given server instance, use the sys.database_mirroring_endpoints catalog view. For more information, see either How to: Create a Mirroring Endpoint for Windows Authentication (Transact-SQL) or How to: Allow Database Mirroring to Use Certificates for Outbound Connections (Transact-SQL).
2. Check that the port numbers are correct.

   To identify the port currently associated with database mirroring endpoint of a server instance, use the following Transact-SQL statement:

```
SELECT type_desc, port FROM sys.tcp_endpoints;
GO
```

3. For AlwaysOn Availability Groups setup issues that are difficult to explain, we recommend that you inspect each server instance to determine whether it is listening on the correct ports. For information about verifying port availability, see MSSQLSERVER_1418.
4. Make sure that the endpoints are started (STATE=STARTED). On each server instance, use the following Transact-SQL statement:

   ```
   SELECT state_desc FROM sys.database_mirroring_endpoints
   ```

   For more information about the **state_desc** column, see sys.endpoints (Transact-SQL).

   To start an endpoint, use the following Transact-SQL statement:

   ```
   ALTER ENDPOINT Endpoint_Mirroring
   STATE = STARTED
   AS TCP (LISTENER_PORT = <port_number>)
   FOR database_mirroring (ROLE = ALL);
   GO
   ```

   For more information, see ALTER ENDPOINT (Transact-SQL).
5. Make sure that the login from the other server has CONNECT permission. To determine who has CONNECT permission for an endpoint, on each server instance use the following Transact-SQL statement:

   ```
   SELECT 'Metadata Check';
   SELECT EP.name, SP.STATE,
      CONVERT(nvarchar(38), suser_name(SP.grantor_principal_id))
         AS GRANTOR,
      SP.TYPE AS PERMISSION,
      CONVERT(nvarchar(46),suser_name(SP.grantee_principal_id))
         AS GRANTEE
      FROM sys.server_permissions SP , sys.endpoints EP
      WHERE SP.major_id = EP.endpoint_id
      ORDER BY Permission,grantor, grantee;
   GO
   ```

System Name

For the system name of a server instance in an endpoint URL, you can use any name that unambiguously identifies the system. The server address can be a system name (if the systems are in the same domain), a fully qualified domain name, or an IP address (preferably, a static IP address). Using the fully qualified domain name is guaranteed to work. For more information, see Specifying the Endpoint URL When Adding or Modifying a "HADR" Availability Replica (SQL Server).

Network Access

Each server instance that is hosting an availability replica must be able to access the port of each of the other server instance over TCP. This is especially important if the server instances are in different domains that do not trust each other (untrusted domains).

Endpoint Access (SQL Server Error 1418)

This SQL Server message indicates that the server network address specified in the endpoint URL cannot be reached or does not exist, and it suggests that you verify the network address name and reissue the command. For more information, see MSSQLSERVER_1418.

Join Database Fails (SQL Server Error 35250)

This section discusses the possible causes and resolution of a failure to join secondary databases to the availability group because the connection to the primary replica is not active.

Resolution:

1. Check the firewall setting to see if whether allows the endpoint port communication between the server instances that host primary replica and the secondary replica (port 5022 by default).
2. Check whether the network service account has connect permission to the endpoint.

See Also

Database Mirroring Transport Security

Create a Mirroring Endpoint for Windows Authentication (Transact-SQL)

Specifying the Endpoint URL for an Availability Replica (SQL Server)

Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups (SQL Server)

Creation and Configuration of Availability Groups (SQL Server)

Manually Prepare a Secondary Database for an Availability Group (SQL Server)

Troubleshooting a Failed Add-File Operation (AlwaysOn Availability Groups)

Creation and Configuration of Availability Groups

The topics in this section explain how to deploy a AlwaysOn Availability Groups implementation on instances of SQL Server 2012 that reside on different Windows Server Failover Clustering (WSFC) nodes within a single WSFC failover cluster.

Before you create your first availability group, we strongly recommend that you familiarize yourself with the information in the following topics:

Prerequisites, Restrictions, and Recommendations (AlwaysOn Availability Groups)

This topic describes the prerequisites, restrictions, and recommendations for computers;

WSFC nodes; instances of SQL Server; availability groups, replicas, and databases. This topic also contains information about security considerations.

Getting Started with AlwaysOn Availability Groups (SQL Server)

Contains information about the steps for configuring a server instance, creating an availability group, configuring the availability group for client connections, managing availability groups, and monitoring availability groups.

In this Topic:

- Related Tasks
- Related Content

Related Tasks

To configure a server instance for AlwaysOn Availability Groups

- Enable and Disable AlwaysOn Availability Groups (SQL Server)
- Create a Database Mirroring Endpoint for AlwaysOn Availability Groups (SQL Server PowerShell)
- Create a Database Mirroring Endpoint for Windows Authentication (Transact-SQL)
- Allow a Database Mirroring Endpoint to Use Certificates for Outbound Connections (Transact-SQL)

To get started with configuring AlwaysOn Availability Groups

- Getting Started with AlwaysOn Availability Groups (SQL Server)

To create and configure a new availability group

- Create and Configure an Availability Group (New Availability Group Wizard)
- Create and Configure an Availability Group (Transact-SQL)
- Create and Configure an Availability Group (SQL Server PowerShell)
- Use the New Availability Group Dialog Box (SQL Server Management Studio)
- Specify the Endpoint URL When Adding or Modifying an Availability Replica (AlwaysOn Availability Groups)
- Create or Configure an Availability Group Listener (SQL Server)
- Configure the Flexible Failover Policy to Control Conditions for Automatic Failover (AlwaysOn Availability Groups)
- Configure Backup on Availability Replicas (SQL Server)
- Configure Connection Access on an Availability Replica (SQL Server)
- Configure Read-Only Routing on an Availability Group (SQL Server)
- Join a Secondary Replica to an Availability Group (SQL Server)
- Manually Start Data Synchronization on an AlwaysOn Secondary Database (SQL Server)
- Prepare a Secondary Database for an Availability Group (SQL Server)
- Join a Secondary Database to an Availability Group (SQL Server)

- Management of Logins and Jobs for the Databases of an Availability Group (SQL Server)

To troubleshoot

- Troubleshoot AlwaysOn Availability Groups Configuration (SQL Server)
- Troubleshoot a Failed Add-File Operation (AlwaysOn Availability Groups)

Related Content

- Microsoft SQL Server AlwaysOn Solutions Guide for High Availability and Disaster Recovery
- SQL Server AlwaysOn Team Blog: The official SQL Server AlwaysOn Team Blog

See Also

Overview of AlwaysOn Availability Groups

Administration of an Availability Group (SQL Server)

Policy-Based Management of Operational Issues with AlwaysOn Availability Groups (SQL Server)

Monitoring of Availability Groups (SQL Server)

AlwaysOn Availability Groups: Interoperability (SQL Server)

Use the New Availability Group Wizard (SQL Server Management Studio)

This topic describes how to use the New Availability Group Wizard (in SQL Server Management Studio) to create and configure an AlwaysOn availability group in SQL Server 2012. An *availability group* defines a set of user databases that will fail over as a single unit and a set of failover partners, known as *availability replicas*, that support failover.

Note

For an introduction to availability groups, see Overview of AlwaysOn Availability Groups (SQL Server).

- **Before you begin:**

  Prerequisites, Restrictions, and Recommendations

  Considerations for Using the New Availability Group Wizard (SQL Server Management Studio)

  Security

- **To create and configure an availability group, using:** New Availability Group Wizard (SQL Server Management Studio)

Note

As an alternative to using the New Availability Group Wizard, you can use Transact-SQL or SQL Server PowerShell cmdlets. For more information, see Creating and Configuring an Availability Group (Transact-SQL) or Creating and Configuring an Availability Group (SQL Server PowerShell).

Before You Begin

We strongly recommend that you read this section before attempting to create your first availability group.

Prerequisites, Restrictions, and Recommendations

In most cases, you can use the New Availability Group Wizard to complete all of the tasks require to create and configure an availability group. However, you might need to complete some of the tasks manually.

- Before creating an availability group, verify that the instances of SQL Server that host availability replicas reside on different Windows Server Failover Clustering (WSFC) node within the same WSFC failover cluster. Also, verify that each of the server instance meets all other AlwaysOn Availability Groups prerequisites. For more information, we strongly recommend that you read Prerequisites, Restrictions, and Recommendations (AlwaysOn Availability Groups).
- If a server instance that you select to host an availability replica does not yet have a database mirroring endpoint, the wizard can create the endpoint and grant CONNECT permission to the server instance service account if the server instance is running under a domain user account. However, if the SQL Server service is running as a built-in account, such as Local System, Local Service, or Network Service, or a nondomain account, you must use certificates for endpoint authentication, and the wizard will be unable to create a database mirroring endpoint on the server instance. In this case, we recommend that you create the database mirroring endpoints manually before you launch the New Availability Group Wizard. For more information, see Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups (SQL Server).
- SQL Server Failover Cluster Instances (FCIs) do not support automatic failover by availability groups, so any availability replica that is hosted by an FCI can only be configured for manual failover.
- If a database is encrypted or even contains a Database Encryption Key (DEK), you cannot use the New Availability Group Wizard or Add Database to Availability Group Wizard to add the database to an availability group. Even if an encrypted database has been decrypted, its log backups might contain encrypted data. In this case, full initial data synchronization could fail on the database. This is because the restore log operation might require the certificate that was used by the database encryption keys (DEKs), and that certificate might be unavailable.

  To make a decrypted database eligible to add to an availability group using the wizard:

  a. Create a log backup of the primary database.

  b. Create a full database backup of the primary database.

  c. Restore the database backup on the server instance that hosts the secondary replica.

  d. Create a new log backup from primary database.

  e. Restore this log backup on the secondary database.

- **Prerequisites for the wizard to perform full initial data synchronization**
  - All the database-file paths must be identical on every server instance that hosts a replica for the availability group.
  - No primary database name can exist on any server instance that hosts a secondary replica. This means that none of the new secondary databases can exist yet.
  - You will need to specify a network share in order for the wizard to create and access backups. For the primary replica, the account used to start the Database Engine must have read and write file-system permissions on a network share. For secondary replicas, the account must have read permission on the network share.

    Important
    The log backups will be part of your log backup chain. Store the log backup files appropriately.

  If you are unable to use the wizard to perform full initial data synchronization, you need to prepare your secondary databases manually. You can do this before or after running the wizard. For more information, see Manually Prepare a Secondary Database for an Availability Group (SQL Server).

Security

Permissions

Requires membership in the **sysadmin** fixed server role and either CREATE AVAILABILITY GROUP server permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.

Also requires CONTROL ON ENDPOINT permission if you want to allow Availability Group Wizard to manage the database mirroring endpoint.

Using the New Availability Group Wizard

1. In Object Explorer, connect to the server instance that hosts the primary replica.
2. Expand the **AlwaysOn High Availability** node and the **Availability Groups** node.
3. To launch the New Availability Group Wizard, select the **New Availability Group Wizard** command.
4. The first time you run this wizard, an **Introduction** page appears. To bypass this page in the future, you can click **Do not show this page again**. After reading this page, click **Next**.
5. On the **Specify Availability Group Name** page, enter the name of the new availability group in the **Availability group name** field. This name must be a valid SQL Server identifier that is unique on the WSFC failover cluster and in your domain as a whole. The maximum length for an availability group name is 128 characters.
6. On the **Select Databases** page, the grid lists user databases on the connected server instance that are eligible to become the *availability databases*. Select one or more of the listed databases to participate in the new availability group. These databases will initially be the initial *primary databases*.

For each listed database, the **Size** column displays the database size, if known. The **Status** column indicates whether a given database meets the prerequisites for availability databases. It the prerequisites are not met, a brief status description indicates the reason that the database is ineligible; for example, if it does not use the full recovery model. For more information, click the status description.

If you change a database to make it eligible, click **Refresh** to update the databases grid.

7. On the **Specify Replicas** page, specify and configure one or more replicas for the new availability group. This page contains four tabs. The following table introduces these tabs. For more information, see the Specify Replicas Page (New Availability Group Wizard/Add Replica Wizard) topic.

Tab	**Brief Description**
Replicas	Use this tab to specify each instance of SQL Server that will host a secondary replica. Note that the server instance to which you are currently connected must host the primary replica.
Endpoints	Use this tab to verify any existing database mirroring endpoints and also, if this endpoint is lacking on a server instance whose service accounts use Windows Authentication, to create the endpoint automatically. **Note** If any server instance is running under a non-domain user account, you need to do make a manual change to your server instance before you can proceed in the wizard.
Backup Preferences	Use this tab to specify your backup preference for the availability group as a whole and your backup priorities for the individual availability replicas.
Listener	Use this tab to create an availability group listener. By default, the wizard does not create a listener.

8. On the **Select Initial Data Synchronization** page, choose how you want your new secondary databases to be created and joined to the availability group. Choose one of the following options:

   - **Full**

     Select this option if your environment meets the requirements for automatically starting initial data synchronization (for more information, see Prerequisites, Restrictions, and Recommendations , earlier in this topic).

     If you select **Full**, after creating the availability group, the wizard will back up every primary database and its transaction log to a network share and restore the backups on every server instance that hosts an secondary replica. The wizard will then join every secondary database to the availability group.

     In the **Specify a shared network location accessible by all replicas:** field, specify a backup share to which all of the server instance that host replicas have read-write access. For more information, see Prerequisites, earlier in this topic.

   - **Join only**

     If you have manually prepared secondary databases on the server instances that will host the secondary replicas, you can select this option. The wizard will join the existing secondary databases to the availability group.

   - **Skip initial data synchronization**

     Select this option if you want to use your own database and log backups of your primary databases. For more information, see Manually Start Data Synchronization on an AlwaysOn Secondary Database (SQL Server).

9. The **Validation** page verifies whether the values you specified in this Wizard meet the requirements of the New Availability Group Wizard. To make a change, click **Previous** to return to an earlier wizard page to change one or more values. The click **Next** to return to the **Validation** page, and click **Re-run Validation**.

10. On the **Summary** page, review your choices for the new availability group. To make a change, click **Previous** to return to the relevant page. After making the change, click **Next** to return to the **Summary** page.

    noteDXDOC112778PADS Security Note

    When the SQL Server service account of a server instance that will host a new availability replica does not already exist as a login, the New Availability Group Wizard needs to create the login. On the **Summary** page, the wizard displays the information for the login that is to be created. If you click **Finish**, the wizard creates this login for the SQL Server service account and grants the login CONNECT permission.

    If you are satisfied with your selections, optionally click **Script** to create a script of the steps the wizard will execute. Then, to create and configure the new availability group, click **Finish**.

11. The **Progress** page displays the progress of the steps for creating the availability group (configuring endpoints, creating the availability group, and joining the secondary replica to the group).
12. When these steps complete, the **Results** page displays the result of each step. If all these steps succeed, the new availability group is completely configured. If any of the steps result in an error, you might need to manually complete the configuration or use a wizard for the failed step. For information about the cause of a given error, click the associated "Error" link in the **Result** column.

    When the wizard completes, click **Close** to exit.

Related Tasks

To complete availability group configuration

- Join a Secondary Replica to an Availability Group (SQL Server)
- Manually Prepare a Secondary Database for an Availability Group (SQL Server)
- Join a Secondary Database to an Availability Group (SQL Server)
- Create or Configure an Availability Group Listener (SQL Server)

Alternative ways to create an availability group

- Use the New Availability Group Dialog Box (SQL Server Management Studio)
- Create an Availability Group (Transact-SQL)
- Create and Configure an Availability Group (SQL Server PowerShell)

To enable AlwaysOn Availability Groups

- Enable and Disable the AlwaysOn Availability Groups Feature (SQL Server)

To configure a database mirroring endpoint

- Create a Database Mirroring Endpoint for AlwaysOn Availability Groups (SQL Server PowerShell)
- Create a Mirroring Endpoint for Windows Authentication (Transact-SQL)
- Use Certificates for a Database Mirroring Endpoint
- Specify the Endpoint URL When Adding or Modifying an Availability Replica (SQL Server)

To troubleshoot AlwaysOn Availability Groups configuration

- Troubleshoot AlwaysOn Availability Groups Configuration (SQL Server)
- Troubleshoot a Failed Add-File Operation (AlwaysOn Availability Groups)

Related Content

- Microsoft SQL Server AlwaysOn Solutions Guide for High Availability and Disaster Recovery

See Also

Database Mirroring Endpoint

Overview of AlwaysOn Availability Groups

Prerequisites, Restrictions, and Recommendations (AlwaysOn Availability Groups)

Specify Availability Group Name Page (New Availability Group Wizard/Add Database Wizard)

This topic describes the options of the **Specify Availability Group Name** page. This topic is used by both the New Availability Group Wizard and Add Database to Availability Group Wizard of SQL Server 2012.

Specify Availability Group Name Options

Specify an availability group name

Specify the name of the availability group. For a new availability group, specify a valid SQL Server identifier that is unique across all availability groups in the WSFC cluster. The maximum length for an availability group name is 128 characters.

Related Tasks

- Use the New Availability Group Wizard
- Use the Add Database to Availability Group Wizard

See Also

AlwaysOn Availability Groups

Select Databases Page (New Availability Group Wizard/Add Database Wizard)

This help topic describes the options of the **Specify Databases** page. This topic applies to the New Availability Group Wizard and Add Database to Availability Group Wizard of SQL Server 2012.

Select Databases Options

The **User databases on this instance of SQL Server** grid lists every local user database. The columns are as follows:

Name

Displays the name of a local user database.

Size

Displays the database size, if the size is available to the wizard.

Status

Displays a hyperlink whose text that indicates whether a given database meets the prerequisites for being added to an availability group. If the status is "**Meets prerequisites**" you can add the database to the availability group. If a database does not meet all of the prerequisites, the **Status** hyperlink provides a brief explanation of why the database is ineligible. For more information, click the hyperlink.

You can leave the wizard on the **Select Database** page while you take action on a database to meet a prerequisite. When you return to the **Select Databases** page, click **Refresh** to update the grid.

Refresh

Click to refresh the grid. This is useful after you take action on a database to meet a prerequisite.

Related Tasks

- Use the New Availability Group Wizard
- Use the Add Database to Availability Group Wizard

See Also

Overview of AlwaysOn Availability Groups

Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups (SQL Server)

Specify Replicas Page (New Availability Group Wizard/Add Replica Wizard)

This topic describes the options of the **Specify Replicas** page. This page applies to the New Availability Group Wizard and the Add Replica to Availability Group Wizard of SQL Server 2012. Use the **Specify Replicas** page to specify and configure one or more availability replicas to add the availability group. This page contains four tabs, which are introduced in the following table. Click the name of a tab in the table to go to the corresponding section, later in this topic.

Tab	Brief Description
Replicas	Use this tab to specify each instance of SQL Server that will host or currently hosts a secondary replica. Note that the server instance to which you are currently connected must host the primary replica. **Tip** Finish specifying all the replicas on the Replicas tab before starting the other tabs.
Endpoints	Use this tab to verify any existing database mirroring endpoints and also, if this endpoint is lacking on a server instance whose service accounts use Windows Authentication, to create the endpoint

Tab	Brief Description
	automatically.
Backup Preferences	Use this tab to specify your backup preference for the availability group as a whole and your backup priorities for the individual availability replicas.
Listener	Use this tab, if available, to create an availability group listener. By default, a listener is not created. **Note** This tab is available only if you are running the New Availability Group Wizard.

Replicas Tab

Server Instance

Displays the name of the server instance that will host the availability replica.

If a server instance that you to use to host a secondary replica is not listed by the **Availability Replicas** grid, click the **Add Replica** button.

Initial Role

Indicates the role that the new replica will initially perform: **Primary** or **Secondary**.

Automatic Failover (Up to 2)

Select this checkbox only if you want this availability replica to be an automatic-failover partner. To configure automatic failover, you must choose this option for the initial primary replica and for one secondary replica. Both of these replicas will use the synchronous-commit availability mode. Only two replicas can support automatic failover.

For information about the synchronous-commit availability mode, see Availability Modes . For information about automatic failover, see Failover Modes (AlwaysOn Availability Groups).

Synchronous Commit (Up to 3)

If you selected **Automatic Failover (Up to 2)** for the replica, **Synchronous Commit (Up to 3)** is also selected. If the check box is blank, select it only if you want this replica to use synchronous-commit mode with only planned manual failover. Only three replicas can use synchronous-commit mode.

If you want this replica to use asynchronous-commit availability mode, leave this checkbox blank. The replica will support only forced manual failover (with possible data loss). For information about the asynchronous-commit availability mode, see Availability Modes .

For information about planned manual failover and forced manual failover, see Failover Modes (AlwaysOn Availability Groups).

Readable Secondary Role

Select a value from the **Readable secondary** drop list, as follows:

No

No direct connections are allowed to secondary databases of this replica. They are not available for read access. This is the default setting.

Read-intent only

Only direct read-only connections are allowed to secondary databases of this replica. The secondary database(s) are all available for read access.

Yes

All connections are allowed to secondary databases of this replica, but only for read access. The secondary database(s) are all available for read access.

Add Replica

Click to add a secondary replica to the availability group.

Remove Replica

Click to remove the selected secondary replica from the availability group.

Endpoints Tab

For each server instance that will host an availability replica, the **Endpoints** tab displays actual values of the existing database mirroring endpoint, if any, or suggested values for a potential new endpoint that would use Windows Authentication. For both existing and potential endpoints, the Endpoint values grid displays the following information:

Server Name

Displays the name of a server instance that will host an availability replica.

Endpoint URL

Displays the actual or proposed URL of the database mirroring endpoint. For a proposed new endpoint, you can change this value. For information the format of these URLs, see Specifying the Endpoint URL for an Availability Replica (SQL Server).

Port Number

Displays the actual or proposed port number of the endpoint. For a proposed new endpoint, you can change this value.

Endpoint Name

Displays the actual or proposed name of the endpoint. For a proposed new endpoint, you can change this value.

Encrypt Data

Indicates whether data sent over this endpoint is encrypted. For a proposed new endpoint, you can change this setting.

SQL Server Service Account

Username of the SQL Server service account.

For a server instance to use an endpoint that uses Windows Authentication, its SQL Server service account must be a domain account.

This requirement determines your next configuration step, as follows:

- If every server instance is running under a domain service account, that is, if the **SQL Server Service Account** column displays a domain service account for every server instance, click **Next**.
- If any server instance is running under a non-domain service account, you need to do make a manual change to your server instance before you can proceed in the wizard. In this case, clicking **Next** brings up a warning dialog box; you should click **No**, which returns you to the**Endpoints** tab. While leaving the wizard on the **Specify Replicas** page, make one of the following changes to each server instance for which the **SQL Server Service Account** column displays a nondomain service account, either:
    - Use the SQL Server Configuration Manager to change the SQL Server service account to a domain account. For more information, see How to: Change the Service Startup Account for SQL Server (SQL Server Configuration Manager).
    - Use Transact-SQL or PowerShell to manually create a database mirroring endpoint that uses a certificate. For more information, see CREATE ENDPOINT (Transact-SQL) or Create a Database Mirroring Endpoint for AlwaysOn Availability Groups (SQL Server PowerShell).

  If you leave the **Specify Availability Replicas** page open while you configure endpoints, return to the **Endpoints** tab and click **Refresh** to update the **Endpoint values** grid.

Backup Preferences Tab

To specify where backups should occur, choose one of the following options:

Prefer Secondary

Specifies that backups should occur on a secondary replica except when the primary replica is the only replica online. In that case, the backup should occur on the primary replica. This is the default option.

Secondary only

Specifies that backups should never be performed on the primary replica. If the primary replica is the only replica online, the backup should not occur.

Primary

Specifies that the backups should always occur on the primary replica. This option is useful if you need backup features, such as creating differential backups, that are not supported when backup is run on a secondary replica.

Any Replica

Specifies that you prefer that backup jobs ignore the role of the availability replicas when choosing the replica to perform backups. Note backup jobs might evaluate other factors such as backup priority of each availability replica in combination with its operational state and connected state.

Important

There is no enforcement of the backup-preference setting. The interpretation of this preference depends on the logic, if any, that you script into back jobs for the databases in a given availability group. For more information, see Backup on Secondary Replicas (AlwaysOn Availability Groups).

Replica backup priorities grid

Use the **Replica backup priorities** grid to specify your backup priorities for each of replicas of the availability group. This grid contains the following columns:

Server Instance

Displays the name of the instance of SQL Server that hosts the availability replica.

Backup Priority (Lowest=1, Highest=100)

Assign the priority for backups being performed on this replica relative to the other replicas in the same availability group. The default value is 50. You can select any other integer in the range of 0..100. 1 indicates the lowest priority, and 100 indicates the highest priority. If you set **Backup Priority** to 1, the availability replica will be choosen for performing backups only if no higher priority availability replica is currently available.

Exclude Replica

To prevent this availability replica from ever being be chosen for performing backups. This is useful, for example, for a remote availability replica to which you never want backups to fail over.

Listener Tab

Specify your preference for an availability group listener that will provide a client connection point, one of:

Do not create an availability group listener now.

Select to skip this step. You can create a listener later. For more information, see Create or Configure an Availability Group Listener (SQL Server).

Create an availability group listener.

Specify your listener preferences for this availability group, as follows:

Listener DNS Name

Specify the network name of the listener. This name must be unique on the domain and can contain only alphanumeric characters, dashes (**-**), and hyphens (_), in any order. When specified by using the **Listener** tab, the DNS name can up to 15 characters long.

Important

If you enter an invalid DNS listener name (or port number) on the **Listener** tab, the **Next** button is disabled on the **Specify Replicas** page.

Port

Specify the TPC port used by this listener.

Note

If you enter an invalid port number (or DNS listener name) on the **Listener** tab, the **Next** button is disabled on the **Specify Replicas** page.

Network Mode

Use the drop list to select the network mode to be used by this listener, one of:

Static IP

Select if you want the listener to listen on more than one subnet. To use the static IP network mode, an availability group listener must listen on every subnet that hosts an availability replica for the availability group. For each subnet, click **Add** to select a subnet address and to specify an IP address.

If **Static IP** is selected as the network mode (this is the default selection), a grid displays the **Subnet** and **IP Address** columns, and the associated **Add** and **Remove** buttons are displayed. Note that the grid is empty until you add the first subnet.

Subnet column

Displays the subnet address that you selected for each subnet you have added for the listener.

IP Address column

Displays the IPv4 or IPv6 address that you specified for a given subnet.

Add

Click to add a subnet to this listener. This opens the **Add IP Address** dialog box. For more information, see the Add IP Address Dialog Box (SQL Server Management Studio) help topic.

Remove

Click to remove the subnet that is currently selected in the grid.

DHCP

Select if you want the listener to listen on a single subnet and to use a dynamic IPv4 address that is assigned by a server running the Dynamic Host Configuration Protocol (DHCP). DHCP is limited to a single subnet that is common to every server instance that host an availability replica for the availability group.

Important

We do not recommend DHCP in production environment. If there is a down time and the DHCP IP lease expires, extra time is required to register the new DHCP network IP address that is associated with the listener DNS name and impact the client connectivity. However, DHCP is good for setting up your development and testing environment to verify basic functions of availability groups and for integration with your applications.

When **DHCP** is selected, the **Subnet** field is displayed.

Subnet

If you selected **DHCP** as the network mode, use the **Subnet** drop list to select an address for the subnet that hosts the availability replicas of the availability group.

Important

- After you define an availability group listener, we strongly recommend that you do the following:

Related Tasks

- Use the New Availability Group Wizard (SQL Server Management Studio)
- Use the Add Replica to Availability Group Wizard
- Create or Configure an Availability Group Listener (SQL Server)
- Use Certificates for a Database Mirroring Endpoint (SQL Server)
- CREATE ENDPOINT (Transact-SQL)
- Create a Database Mirroring Endpoint for AlwaysOn Availability Groups (SQL Server PowerShell)

See Also

Overview of AlwaysOn Availability Groups

CREATE AVAILABILITY GROUP (Transact-SQL)

Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups (SQL Server)

Select Initial Data Synchronization Page (AlwaysOn Availability Group Wizards)

Use the AlwaysOn **Select Initial Data Synchronization** page to indicate your preference for initial data synchronization of new secondary databases. This page is shared by three wizards—the New Availability Group Wizard, the Add Replica to Availability Group Wizard, and the Add Database to Availability Group Wizard.

The possible choices include **Full**, **Join only**, or **Skip initial data synchronization**. Before you select **Full** or **Join only** ensure that your environment meets the prerequisites.

In this topic:

- Recommendations
- Full
- Join only
- Skip initial data synchronization
- To Prepare Secondary Databases Manually
- Related Tasks

Recommendations

- Suspend log backup tasks for the primary databases during initial data synchronization.
- For large database, full backup and restore operations can take extensive time and resources. In such cases, we recommend that you prepare secondary databases yourself. For more information, see To Prepare Secondary Databases Manually, later in this topic.
- Full initial data synchronization requires you to specify a network share. Before you use a wizard to perform full initial data synchronization, we recommend that you implement a security plan for the access permissions on the network share folder. This precaution is important because potentially sensitive data in the backup file can be accessed by anyone who has a READ permission on the folder. Also, to protect your backup and restore operations, we recommend that you secure the network channels between every server instance that hosts an availability replica and the network share folder.

  If your backup and restore operations must be highly secured, we recommend that you select either the **Join only** or **Skip initial data synchronization** option.

Full

For each primary database, the **Full** option performs several operations in one workflow: create a full and log backup of the primary database, create the corresponding secondary databases by restoring these backups on every server instance that is hosting a secondary replica, and join each secondary database to availability group.

Select this option only if your environment meets the following prerequisites for using full initial data synchronization, and you want the wizard to automatically start data synchronization.

Prerequisites for using full initial data synchronization

- All the database-file paths must be identical on every server instance that hosts a replica for the availability group.

  Note
  If the backup and restore file paths differ between the server instance where you run the wizard and any server instance that is to host a secondary replica. The backup and restore operations must be performed manually using the WITH MOVE option.

For more information, see To Prepare Secondary Databases Manually, later in this topic.

- No primary database name can exist on any server instance that hosts a secondary replica. This means that none of the new secondary databases can exist yet.
- You will need to specify a network share in order for the wizard to create and access backups. For the primary replica, the account used to start the Database Engine must have read and write file-system permissions on a network share. For secondary replicas, the account must have read permission on the network share.

Important
The log backups will be part of your log backup chain. Store the log backup files appropriately.

If prerequisites are not met

The wizard cannot create the secondary databases for this availability group. For information on how to prepare them, see To Prepare Secondary Databases Manually, later in this topic.

If prerequisites are met

If these prerequisites are all met and you want the wizard to perform full initial data synchronization, select the **Full** option and specify a network share. This will cause the wizard to create full database and log backups of every selected database and to place these backups on the network share that you specify. Then, on every server instance that hosts one of the new secondary replicas, the wizard will create the secondary databases by restoring backups using RESTORE WITH NORECOVERY. After creating each of the secondary databases, the wizard will join the new secondary database to the availability group. As soon as a secondary database is joined, data synchronizations starts on that database.

Specify a shared network location accessible by all replicas

To create and restore backups, the wizard requires that you specify a network share. The account used to start the Database Engine on each server instance that will host an availability replica must have read and write file-system permissions on the network share.

Important
The log backups will be part of your log backup chain. Store their backup files appropriately.

Join only

Select this option only if the new secondary databases already exist on each server instance that hosts a secondary replica for the availability group. For information about preparing secondary databases, see To Prepare Secondary Databases Manually, later in this section.

If you select **Join only**, the wizard will attempt to join each existing secondary database to the availability group.

Skip initial data synchronization

Select this option if you want to perform your own database and log backups of every primary database, restore them to every server instance that hosts a secondary replica. After you exit the wizard, you will then need to join every secondary database on every secondary replica.

Note
For more information, see Manually Start Data Synchronization on an AlwaysOn Secondary Database (SQL Server).

To Prepare Secondary Databases Manually

To prepare secondary databases independently of any AlwaysOn Availability Groups wizard, you can use either of the following approaches:

- Manually restore a recent database backup of the primary database using RESTORE WITH NORECOVERY, and then restore each subsequent log backup using RESTORE WITH NORECOVERY. If the primary and secondary databases have different file paths, you must use the WITH MOVE option. Perform this restore sequence on every server instance that hosts a secondary replica for the availability group. You can use Transact-SQL or PowerShell to perform these backup and restore operations.

  For more information:

  Manually Prepare a Secondary Database for an Availability Group (SQL Server)

- If you are adding one or more log shipping primary databases to an availability group, you might be able to migrate one or more of the corresponding secondary databases from log shipping to AlwaysOn Availability Groups. For more information, see Prerequisites for Migrating from Log Shipping to AlwaysOn Availability Groups (SQL Server).

  Note
  After you have created all the secondary databases for the availability group, if you want to perform backups on secondary replicas, you will need to re-configure the automated backup preference of the availability group.

  For more information:

  Prerequisites for Migrating from Log Shipping to AlwaysOn Availability Groups (SQL Server)

  Configure Backup on Availability Replicas (SQL Server)

After creating a secondary database, apply all current log backups to the new secondary database.

Optionally, you can prepare all the secondary databases before you run the wizard. Then, on the wizard's **Specify Initial Data Synchronization** page, select **Join only** to automatically join your new secondary databases to the availability group.

Related Tasks

- Use the New Availability Group Wizard
- Use the Add Replica to Availability Group Wizard

- Use the Add Database to Availability Group Wizard
- Use the Fail Over Availability Group Wizard
- Manually Start Data Synchronization on an AlwaysOn Secondary Database (SQL Server)
- Join a Secondary Database to an Availability Group (SQL Server)
- Use the New Availability Group Dialog Box (SQL Server Management Studio)

See Also

Overview of AlwaysOn Availability Groups

Validation Page (AlwaysOn Availability Group Wizards)

This help topic describes the options of the **Validation** page. This topic applies to the New Availability Group Wizard, Add Replica to Availability Group Wizard, and Add Database to Availability Group Wizard of SQL Server 2012. Use this page to validate that your environment supports all the configuration choices you made on previous pages of the wizard.

Validation Page Options

Results of availability group validation.

This grid displays the results of each completed validation step. The grid columns are as follows:

Name

Displays a phrase that describes a specific step.

Result

Displays one of the following hyperlink texts. For more information about the result of given validation step, click the hyperlink.

Result	Description
Error	Indicates that the validation step failed. Click the link to view the error message.
Skipped	Indicates that the validation step was skipped because it is not required by your selections. Click the link to view the reason that a step was skipped.
Success	Indicates that the validation step completed successfully
Warning	Indicates a potential issue with the availability group configuration. Click the link to view the warning message.

Re-run Validation

Click to repeat the validation steps if you make a change outside of the wizard in response to a validation error.

Related Tasks

- Use the New Availability Group Wizard
- Use the Add Replica to Availability Group Wizard
- Use the Add Database to Availability Group Wizard

See Also

AlwaysOn Availability Groups

Summary Page (AlwaysOn Availability Group Wizards)

This help topic describes the options of the **Summary** page. This topic applies to the New Availability Group Wizard, Add Replica to Availability Group Wizard, Add Database to Availability Group Wizard and Fail Over Availability Group Wizard of SQL Server 2012. Use the grid on this page to review your choices for the new availability group. To make one or more changes, click **Previous** to return to the relevant page or pages. When you are ready, click **Next** to return to the **Summary** page. Once you are satisfied with your choices, click **Finish**.

Summary Page Options

Script

Click to generate a Transact-SQL script for the actions listed in the summary grid. You will be prompted to specify a destination for the script.

Previous

Click to return to the page immediately preceding the current page. You can use the **Previous** button to navigate backward to any of the preceding pages and, optionally, change any of your specified values.

Finish

Once you are satisfied with your choices, click to make the wizard proceed with creating the availability group.

Cancel

Click to cancel the wizard. On the **Summary** page, cancelling the wizard causes it to exit without performing any actions.

Related Tasks

- Use the New Availability Group Wizard
- Use the Add Replica to Availability Group Wizard
- Use the Add Database to Availability Group Wizard
- Use the Fail Over Availability Group Wizard

See Also

AlwaysOn Availability Groups

Progress Page (AlwaysOn Availability Group Wizards)

Use this dialog box to view the progress of a AlwaysOn Availability Groups wizard that you are running in SQL Server 2012. The progress bar indicates the relative progress of the steps that the wizard is performing.

UI Element List

More details

Click the down arrow to display a progress grid that lists any completed steps, in order, followed by the current in-progress operation. The grid contains the following columns:

Name

Displays a phrase that describes a specific step.

Status

Indicates the outcome of completed steps and the percentage of completion of the current step, as follows:

Result	Description
Error	Indicates the operation for this step experienced an error. Click the link to display a message dialog box that describes the error.
In Progress (*percentage-completed*)	Indicates that the operation is occurring now and estimates the percentage of this step that has completed.
Success	Indicates the operation for this step completed successfully.

Fewer Details

Click to hide the progress grid.

Cancel

Click to skip any remaining operations and then exit the wizard.

Related Tasks

- Use the New Availability Group Wizard
- Use the Add Replica to Availability Group Wizard
- Use the Add Database to Availability Group Wizard
- Use the Fail Over Availability Group Wizard

See Also

Overview of AlwaysOn Availability Groups (SQL Server)

Results Page (AlwaysOn Availability Group Wizards)

This help topic describes the options of the **Results** page. This topic applies to the New Availability Group Wizard, Add Replica to Availability Group Wizard, Add Database to Availability Group Wizard, and Fail Over Availability Group Wizard of SQL Server 2012. Use this page to view the results of the wizard.

Results Page Options

The **Summary** grid contains the following columns:

Name

Displays a phrase that describes a specific operation.

Result

For each completed step, displays one of the following hyperlink texts.

Result	Description
Error	Indicates that the validation step failed. Click the link to view the error message.
Success	Indicates that the validation step completed successfully.

Related Tasks

- Use the New Availability Group Wizard
- Use the Add Replica to Availability Group Wizard
- Use the Add Database to Availability Group Wizard
- Use the Fail Over Availability Group Wizard

See Also

AlwaysOn Availability Groups

Use the New Availability Group Dialog Box (SQL Server Management Studio)

This topic contains information about how to use the **New Availability Group** dialog box of SQL Server Management Studio to create an AlwaysOn availability group on instances of SQL Server 2012 that are enabled for AlwaysOn Availability Groups. An *availability group* defines a set of user databases that will fail over as a single unit and a set of failover partners, known as *availability replicas*, that support failover.

Note
For an introduction to availability groups, see AlwaysOn Availability Groups.

- **Before you begin:**

  Prerequisites

  Limitations

  Security

  Summary of Tasks and Corresponding Transact-SQL Statements

- **To create an availability group, using:** The New Availability Group Dialog Box
- **Follow up:** After Using the New Availability Group Dialog Box to Create an Availability Group

Note
For information about alternative ways to create an availability group, see Related Tasks, later in this topic.

Before You Begin

We strongly recommend that you read this section before attempting to create your first availability group.

Prerequisites

- Before creating an availability group, verify that the instances of SQL Server that host availability replicas reside on different Windows Server Failover Clustering (WSFC) node within the same WSFC failover cluster. Also, verify that each of the server instance is enabled for AlwaysOn Availability Groups and meets all other AlwaysOn Availability Groups prerequisites. For more information, we strongly recommend that you read Prerequisites, Restrictions, and Recommendations (AlwaysOn Availability Groups).
- Before you create an availability group, ensure that every server instance that will host an availability replica has a fully functioning database mirroring endpoint. For more information, see Database Mirroring Endpoint.
- To use the **New Availability Group** dialog box, you need to know the names of the server instances that will host availability replicas. Also, you need know the names of any databases

that you intend to add to your new availability group, and you need to ensure that these databases meet the availability database prerequisites and restrictions described in Prerequisites, Restrictions, and Recommendations (AlwaysOn Availability Groups). If you enter invalid values, the new availability group will not work.

Limitations

The **New Availability Group** dialog box does not:

- Create an availability group listener.
- Join secondary replicas to the availability group.
- Perform initial data synchronization.

For information about these configuration tasks, see Follow Up: After Creating an Availability Group, later in this topic.

Security

Permissions

Requires membership in the **sysadmin** fixed server role and either CREATE AVAILABILITY GROUP server permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.

Using the New Availability Group Dialog Box (SQL Server Management Studio)

To create an availability group

1. In Object Explorer, connect to the server instance that hosts the primary replica, and click the server name.
2. Expand the **AlwaysOn High Availability** node.
3. Right-click the **Availability Groups** node, and select the **New Availability Group** command.
4. This command opens up the **New Availability Group** dialog box.
5. On the **General** page, use the **Availability group name** field to enter a name for the new availability group. This name must be a valid SQL Server identifier that is unique across all availability groups in the WSFC cluster. The maximum length for an availability group name is 128 characters.
6. In the **Availability Databases** grid, click **Add** and enter the name of a local database that you want to belong to this availability group. Repeat this for every database to be added. When you click **OK**, the dialog will verify whether your specified database meet the prerequisites for belonging to an availability group. For information about these prerequisites, see Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups.
7. In the **Availability Databases** grid, click **Add** and enter the name of a server instance to host a secondary replica. The dialog will not attempt to connect to these instances. If you specify an incorrect server name, a secondary replica will be added but you will be unable to connect to that replica.

Tip

If you have added a replica and cannot connect to the host server instance, you can remove the replica and add a new one. For more information, see Remove a Secondary Replica from an Availability Group (SQL Server) and Add a Secondary Replica to an Availability Group (SQL Server).

8. On the **Select a page** pane of the dialog box, click **Backup Preferences**. Then, on the **Backup Preferences** page, specify where backups should occur based on replica role and assign backup priorities to each server instances that will host an availability replica for this availability group. For more information, see Availability Group Properties/New Availability Group (Backup Preferences Page).
9. To create the availability group, click **OK**. This causes the dialog to verify whether that specified databases meet the prerequisites.

   To exit the dialog box without creating the availability group, click **Cancel**.

Follow Up: After Using the New Availability Group Dialog Box to Create an Availability Group

- You will need to connect, in turn, to each server instance that is hosting a secondary replica for the availability group and complete the following steps:
  a. Join the secondary replica to the availability group. For more information, see Join a Secondary Replica to an Availability Group (SQL Server).
  b. Restore current backups of each primary database and its transaction log (using RESTORE WITH NORECOVERY). For more information, see Prepare a Secondary Database for an Availability Group (SQL Server).
  c. Immediately join each newly prepared secondary database to the availability group. For more information, see Join a Secondary Database to an Availability Group (SQL Server).
- We recommend that you create an availability group listener for the new availability group. This requires that you be connected to the server instance that hosts the current primary replica. For more information, see Create or Configure an Availability Group Listener (SQL Server).

Related Tasks

To configure availability group and replica properties

- Set the Availability Mode of an Availability Replica (SQL Server)
- Set the Failover Mode of an Availability Replica (SQL Server)
- Create or Configure an Availability Group Listener (SQL Server)
- Configure the Flexible Failover Policy to Control Conditions for Automatic Failover (AlwaysOn Availability Groups)

- Specify the Endpoint URL When Adding or Modifying an Availability Replica (AlwaysOn Availability Groups)
- Configure Backup on Availability Replicas (SQL Server)
- Configure Connection Access on an Availability Replica (SQL Server)
- Configure Read-Only Routing on an Availability Group (SQL Server)
- Change the Session-Timeout Period for an Availability Replica (SQL Server)

To complete availability group configuration

- Join a Secondary Replica to an Availability Group (SQL Server)
- Manually Prepare a Secondary Database for an Availability Group (SQL Server)
- Join a Secondary Database to an Availability Group (SQL Server)
- Create or Configure an Availability Group Listener (SQL Server)

Alternative ways to create an availability group

- Use the New Availability Group Wizard (SQL Server Management Studio)
- Create an Availability Group (Transact-SQL)
- Create and Configure an Availability Group (SQL Server PowerShell)

To enable AlwaysOn Availability Groups

- Enable and Disable the AlwaysOn Availability Groups Feature (SQL Server)

To configure a database mirroring endpoint

- Create a Database Mirroring Endpoint for AlwaysOn Availability Groups (SQL Server PowerShell)
- Create a Mirroring Endpoint for Windows Authentication (Transact-SQL)
- Use Certificates for a Database Mirroring Endpoint
- Specify the Endpoint URL When Adding or Modifying an Availability Replica (SQL Server)

To troubleshoot AlwaysOn Availability Groups configuration

- Troubleshoot AlwaysOn Availability Groups Configuration (SQL Server)
- Troubleshoot a Failed Add-File Operation (AlwaysOn Availability Groups)

Related Content

- Microsoft SQL Server AlwaysOn Solutions Guide for High Availability and Disaster Recovery

See Also

AlwaysOn Availability Groups (SQL Server)

Database Mirroring Endpoint

Client Connectivity and Application Failover (AlwaysOn Availability Groups)

Prerequisites, Restrictions, and Recommendations (AlwaysOn Availability Groups)

Create an Availability Group (Transact-SQL)

This topic describes how to use Transact-SQL to create and configure an availability group on instances of SQL Server 2012 on which the AlwaysOn Availability Groups feature is enabled. An *availability group* defines a set of user databases that will fail over as a single unit and a set of failover partners, known as *availability replicas*, that support failover.

Note
For an introduction to availability groups, see Overview (AlwaysOn Availability Groups).

- **Before you begin:**

  Prerequisites

  Security

  Summary of Tasks and Corresponding Transact-SQL Statements
- **To create and configure an availability group, using:** Transact-SQL
- **Example:** Configuring an Availability Group that Uses Windows Authentication
- Related Tasks
- Related Content

Note
As an alternative to using Transact-SQL, you can use the Create Availability Group wizard or SQL Server PowerShell cmdlets. For more information, see Use the New Availability Group Wizard (SQL Server Management Studio), Use the New Availability Group Dialog Box (SQL Server Management Studio), or Create an Availability Group (SQL Server PowerShell).

Before You Begin

We strongly recommend that you read this section before attempting to create your first availability group.

Prerequisites, Restrictions, and Recommendations

- Before creating an availability group, verify that the instances of SQL Server that host availability replicas reside on different Windows Server Failover Clustering (WSFC) node within the same WSFC failover cluster. Also, verify that each of the server instance meets all other AlwaysOn Availability Groups prerequisites. For more information, we strongly recommend that you read Prerequisites, Restrictions, and Recommendations (AlwaysOn Availability Groups).

Security

Permissions

Requires membership in the **sysadmin** fixed server role and either CREATE AVAILABILITY GROUP server permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.

Summary of Tasks and Corresponding Transact-SQL Statements

The following table lists the basic tasks involved in creating and configuring an availability group and indicates which Transact-SQL statements to use for these tasks. The AlwaysOn Availability Groups tasks must be performed in the sequence in which they are presented in the table.

Task	Transact-SQL Statement(s)	Where to Perform Task[*]
Create database mirroring endpoint (once per SQL Server instance)	CREATE ENDPOINT endpointName … FOR DATABASE_MIRRORING	Execute on each server instance that lacks database mirroring endpoint.
Create availability group	CREATE AVAILABILITY GROUP	Execute on the server instance that is to host the initial primary replica.
Join secondary replica to availability group	ALTER AVAILABILITY GROUP group_name JOIN	Execute on each server instance that hosts a secondary replica.
Prepare the secondary database	BACKUP and RESTORE.	Create backups on the server instance that hosts the primary replica. Restore backups on each server instance that hosts a secondary replica, using RESTORE WITH NORECOVERY.
Start data synchronization by joining each secondary database to availability group	ALTER DATABASE database_name SET HADR AVAILABILITY GROUP = group_name	Execute on each server instance that hosts a secondary replica.

[*] To perform a given task, connect to the indicated server instance or instances.

Using Transact-SQL to Create and Configure an Availability Group

Note

For a sample configuration procedure containing code examples of each these Transact-SQL statements, see Example: Configuring an Availability Group that Uses Windows Authentication.

1. Connect to the server instance that is to host the primary replica.
2. Create the availability group by using the CREATE AVAILABILITY GROUP Transact-SQL statement.

3. Join the new secondary replica to the availability group. For more information, see Joining a Secondary Replica to an Availability Group (SQL Server).
4. For each database in the availability group, create a secondary database by restoring recent backups of the primary database, using RESTORE WITH NORECOVERY. For more information, see Example: Setting Up an Availability Group Using Windows Authentication (Transact-SQL), starting with the step that restores the database backup.
5. Join every new secondary database to the availability group. For more information, see Joining a Secondary Replica to an Availability Group (SQL Server).

Example: Configuring an Availability Group that Uses Windows Authentication

This example creates a sample AlwaysOn Availability Groups configuration procedure that uses Transact-SQL to set up database mirroring endpoints that use Windows Authentication and to create and configure an availability group and its secondary databases.

This example contains the following sections:

- Prerequisites for Using the Sample Configuration Procedure
- Sample Configuration Procedure
- Complete Code Example for Sample Configuration Procedure

Prerequisites for Using the Sample Configuration Procedure

This sample procedure has the following requirements:

- The server instances must support AlwaysOn Availability Groups. For more information, see "HADR" Prerequisites and Restrictions.
- Two sample databases, *MyDb1* and *MyDb2*, must exist on the server instance that will host the primary replica. The following code examples create and configure these two databases and create a full backup of each. Execute these code examples on the server instance on which you intend to create the sample availability group. This server instance will host the initial primary replica of the sample availability group.
    a. The following Transact-SQL example creates these databases and alters them to use the full recovery model:

```
-- Create sample databases:
CREATE DATABASE MyDb1;
GO
ALTER DATABASE MyDb1 SET RECOVERY FULL;
GO

CREATE DATABASE MyDb2;
GO
ALTER DATABASE MyDb2 SET RECOVERY FULL;
```

```
GO
```

b. The following code example creates a full database backup of *MyDb1* and *MyDb2*. This code example uses a fictional backup share, \\*FILESERVER**SQLbackups*.

```
-- Backup sample databases:
BACKUP DATABASE MyDb1
TO DISK = N'\\FILESERVER\SQLbackups\MyDb1.bak'
    WITH FORMAT
GO

BACKUP DATABASE MyDb2
TO DISK = N'\\FILESERVER\SQLbackups\MyDb2.bak'
    WITH FORMAT
GO
```

[TopOfExample]

Sample Configuration Procedure

In this sample configuration, the availability replica will be created on two stand-alone server instances whose service accounts run under different, but trusted, domains (`DOMAIN1` and `DOMAIN2`).

The following table summarizes the values used in this sample configuration.

Initial role	**System**	**Host SQL Server Instance**
Primary	`COMPUTER01`	`AgHostInstance`
Secondary	`COMPUTER02`	Default instance.

1. Create a database mirroring endpoint named *dbm_endpoint* on the server instance on which you plan to create the availability group (this is an instance named `AgHostInstance` on `COMPUTER01`). This endpoint uses port 7022. Note that the server instance on which you create the availability group will host the primary replica.

```
-- Create endpoint on server instance that hosts the primary replica:
CREATE ENDPOINT dbm_endpoint
    STATE=STARTED
    AS TCP (LISTENER_PORT=7022)
    FOR DATABASE_MIRRORING (ROLE=ALL)
```

```
GO
```

2. Create an endpoint *dbm_endpoint* on the server instance that will host the secondary replica (this is the default server instance on `COMPUTER02`). This endpoint uses port 5022.

    ```
    -- Create endpoint on server instance that hosts the secondary
    replica:
    CREATE ENDPOINT dbm_endpoint
        STATE=STARTED
        AS TCP (LISTENER_PORT=5022)
        FOR DATABASE_MIRRORING (ROLE=ALL)
    GO
    ```

3.

    Note
    If the service accounts of the server instances that are to host your availability replicas run under the same domain account this step is unnecessary. Skip it and go directly to the next step.

    If the service accounts of the server instances run under different domain users, on each server instance, create a login for the other server instance and grant this login permission to access the local database mirroring endpoint.

    The following code example shows the Transact-SQL statements for creating a login and granting it permission on an endpoint. The domain account of the remote server instance is represented here as *domain_name\user_name*.

    ```
    -- If necessary, create a login for the service account,
    domain_name\user_name
    -- of the server instance that will host the other replica:
    USE master;
    GO
    CREATE LOGIN [domain_name\user_name] FROM WINDOWS;
    GO
    -- And Grant this login connect permissions on the endpoint:
    GRANT CONNECT ON ENDPOINT::dbm_endpoint
       TO [domain_name\user_name];
    GO
    ```

4. On the server instance where the user databases reside, create the availability group.

   The following code example creates an availability group named *MyAG* on the server instance on which the sample databases, *MyDb1* and *MyDb2*, were created. The local server instance, `AgHostInstance`, on *COMPUTER01* is specified first. This instance will host the initial primary replica. A remote server instance, the default server instance on *COMPUTER02*, is specified to host a secondary replica. Both availability replica are configured to use asynchronous-commit mode with manual failover (for asynchronous-commit replicas manual failover means forced failover with possible data loss).

   ```
   -- Create the availability group, MyAG:
   CREATE AVAILABILITY GROUP MyAG
      FOR
         DATABASE MyDB1, MyDB2
      REPLICA ON
         'COMPUTER01\AgHostInstance' WITH
            (
            ENDPOINT_URL = 'TCP://COMPUTER01.Adventure-Works.com:7022',
            AVAILABILITY_MODE = ASYNCHRONOUS_COMMIT,
            FAILOVER_MODE = MANUAL
            ),
         'COMPUTER02' WITH
            (
            ENDPOINT_URL = 'TCP://COMPUTER02.Adventure-Works.com:5022',
            AVAILABILITY_MODE = ASYNCHRONOUS_COMMIT,
            FAILOVER_MODE = MANUAL
            );
   GO
   ```

   For additional Transact-SQL code examples of creating an availability group, see CREATE AVAILABILITY GROUP (Transact-SQL).

5. On the server instance that hosts the secondary replica, join the secondary replica to the availability group.

   The following code example joins the secondary replica on `COMPUTER02` to the `MyAG` availability group.

   ```
   -- On the server instance that hosts the secondary replica,
   -- join the secondary replica to the availability group:
   ALTER AVAILABILITY GROUP MyAG JOIN;
   GO
   ```

6. On the server instance that hosts the secondary replica, create the secondary databases.

   The following code example creates the *MyDb1* and *MyDb2* secondary databases by restoring database backups using RESTORE WITH NORECOVERY.

```
-- On the server instance that hosts the secondary replica,
-- Restore database backups using the WITH NORECOVERY option:
RESTORE DATABASE MyDb1
    FROM DISK = N'\\FILESERVER\SQLbackups\MyDb1.bak'
    WITH NORECOVERY
GO

RESTORE DATABASE MyDb2
    FROM DISK = N'\\FILESERVER\SQLbackups\MyDb2.bak'
    WITH NORECOVERY
GO
```

7. On the server instance that hosts the primary replica, back up the transaction log on each of the primary databases.

   Important
   When you are configuring a real availability group, we recommend that, before taking this log backup, you suspend log backup tasks for your primary databases until you have joined the corresponding secondary databases to the availability group.

   The following code example creates a transaction log backup on MyDb1 and on MyDb2.

```
-- On the server instance that hosts the primary replica,
-- Backup the transaction log on each primary database:
BACKUP LOG MyDb1
TO DISK = N'\\FILESERVER\SQLbackups\MyDb1.bak'
    WITH NOFORMAT
GO

BACKUP LOG MyDb2
TO DISK = N'\\FILESERVER\SQLbackups\MyDb2.bak'
    WITHNOFORMAT
GO
```

Tip

Typically, a log backup must be taken on each primary database and then restored on the corresponding secondary database (using WITH NORECOVERY). However, this log backup might be unnecessary if the database has just been created and no log backup has been taken yet or the recovery model has just been changed from SIMPLE to FULL.

8. On the server instance that hosts the secondary replica, apply log backups to the secondary databases.

   The following code example applies backups to *MyDb1* and *MyDb2* secondary databases by restoring database backups using RESTORE WITH NORECOVERY.

   Important

   When you are preparing a real secondary database, you need to apply every log backup taken since the database backup from which you created the secondary database, starting with the earliest and always using RESTORE WITH NORECOVERY. Of course, if you restore both full and differential database backups, you would only need to apply the log backups taken after the differential backup.

```
-- Restore the transaction log on each secondary database,
-- using the WITH NORECOVERY option:
RESTORE LOG MyDb1
    FROM DISK = N'\\FILESERVER\SQLbackups\MyDb1.bak'
    WITH FILE=1, NORECOVERY
GO
RESTORE LOG MyDb2
    FROM DISK = N'\\FILESERVER\SQLbackups\MyDb2.bak'
    WITH FILE=1, NORECOVERY
GO
```

9. On the server instance that hosts the secondary replica, join the new secondary databases to the availability group.

   The following code example, joins the *MyDb1* secondary database and then the *MyDb2* secondary databases to the *MyAG* availability group.

```
-- On the server instance that hosts the secondary replica,
-- join each secondary database to the availability group:
ALTER DATABASE MyDb1 SET HADR AVAILABILITY GROUP = MyAG;
GO
```

```
ALTER DATABASE MyDb2 SET HADR AVAILABILITY GROUP = MyAG;
GO
```

[TopOfExample]

Complete Code Example for Sample Configuration Procedure

The following example merges the code examples from all the steps of the sample configuration procedure. The following table summarized the placeholder values used in this code example. For more information about the steps in this code example, see Prerequisites for Using the Sample Configuration Procedure and Sample Configuration Procedure, earlier in this topic.

Placeholder	**Description**
\\FILESERVER\SQLbackups	Fictional backup share.
\\FILESERVER\SQLbackups\MyDb1.bak	Backup file for MyDb1.
\\FILESERVER\SQLbackups\MyDb2.bak	Backup file for MyDb2.
7022	Port number assigned to each database mirroring endpoint.
COMPUTER01\AgHostInstance	Server instance that hosts the initial primary replica.
COMPUTER02	Server instance that hosts the initial secondary replica. This is the default server instance on `COMPUTER02`.
dbm_endpoint	Name specified for each database mirroring endpoint.
MyAG	Name of sample availability group.
MyDb1	Name of first sample database.
MyDb2	Name of second sample database.
DOMAIN1\user1	Service account of the server instance that is to host the initial primary replica.
DOMAIN2\user2	Service account of the server instance that is to host the initial secondary replica.
TCP://*COMPUTER01.Adventure-Works.com*:*7022*	Endpoint URL of the AgHostInstance instance of SQL Server on COMPUTER01.
TCP://*COMPUTER02.Adventure-Works.com*:*5022*	Endpoint URL of the default instance of SQL Server on COMPUTER02.

Note

For additional Transact-SQL code examples of creating an availability group, see CREATE AVAILABILITY GROUP (Transact-SQL).

```
-- on the server instance that will host the primary replica,
-- create sample databases:
CREATE DATABASE MyDb1;
GO
ALTER DATABASE MyDb1 SET RECOVERY FULL;
GO

CREATE DATABASE MyDb2;
GO
ALTER DATABASE MyDb2 SET RECOVERY FULL;
GO

-- Backup sample databases:
BACKUP DATABASE MyDb1
TO DISK = N'\\FILESERVER\SQLbackups\MyDb1.bak'
    WITH FORMAT
GO

BACKUP DATABASE MyDb2
TO DISK = N'\\FILESERVER\SQLbackups\MyDb2.bak'
    WITH FORMAT
GO

-- Create the endpoint on the server instance that will host the primary
replica:
CREATE ENDPOINT dbm_endpoint
    STATE=STARTED
    AS TCP (LISTENER_PORT=7022)
    FOR DATABASE_MIRRORING (ROLE=ALL)
GO
```

```
-- Create the endpoint on the server instance that will host the secondary
replica:
CREATE ENDPOINT dbm_endpoint
    STATE=STARTED
    AS TCP (LISTENER_PORT=7022)
    FOR DATABASE_MIRRORING (ROLE=ALL)
GO

-- If both service accounts run under the same domain account, skip this
step. Otherwise,
-- On the server instance that will host the primary replica,
-- create a login for the service account
-- of the server instance that will host the secondary replica,
DOMAIN2\user2,
-- and grant this login connect permissions on the endpoint:
USE master;
GO
CREATE LOGIN [DOMAIN2\user2] FROM WINDOWS;
GO
GRANT CONNECT ON ENDPOINT::dbm_endpoint
   TO [DOMAIN2\user2];
GO

-- If both service accounts run under the same domain account, skip this
step. Otherwise,
-- On the server instance that will host the secondary replica,
-- create a login for the service account
-- of the server instance that will host the primary replica, DOMAIN1\user1,
-- and grant this login connect permissions on the endpoint:
USE master;
GO

CREATE LOGIN [DOMAIN1\user1] FROM WINDOWS;
```

```
GO
GRANT CONNECT ON ENDPOINT::dbm_endpoint
   TO [DOMAIN1\user1];
GO

-- On the server instance that will host the primary replica,
-- create the availability group, MyAG:
CREATE AVAILABILITY GROUP MyAG
   FOR
      DATABASE MyDB1, MyDB2
   REPLICA ON
      'COMPUTER01\AgHostInstance' WITH
         (
         ENDPOINT_URL = 'TCP://COMPUTER01.Adventure-Works.com:7022',
         AVAILABILITY_MODE = SYNCHRONOUS_COMMIT,
         FAILOVER_MODE = AUTOMATIC
         ),
      'COMPUTER02' WITH
         (
         ENDPOINT_URL = 'TCP://COMPUTER02.Adventure-Works.com:7022',
         AVAILABILITY_MODE = SYNCHRONOUS_COMMIT,
         FAILOVER_MODE = AUTOMATIC
         );
GO

-- On the server instance that hosts the secondary replica,
-- join the secondary replica to the availability group:
ALTER AVAILABILITY GROUP MyAG JOIN;
GO

-- Restore database backups onto this server instance, using RESTORE WITH
NORECOVERY:
RESTORE DATABASE MyDb1
    FROM DISK = N'\\FILESERVER\SQLbackups\MyDb1.bak'
    WITH NORECOVERY
GO

RESTORE DATABASE MyDb2
    FROM DISK = N'\\FILESERVER\SQLbackups\MyDb2.bak'
    WITH NORECOVERY
GO

-- Back up the transaction log on each primary database:
BACKUP LOG MyDb1
TO DISK = N'\\FILESERVER\SQLbackups\MyDb1.bak'
    WITH NOFORMAT
GO

BACKUP LOG MyDb2
TO DISK = N'\\FILESERVER\SQLbackups\MyDb2.bak'
    WITHNOFORMAT
GO

-- Restore the transaction log on each secondary database,
-- using the WITH NORECOVERY option:
RESTORE LOG MyDb1
    FROM DISK = N'\\FILESERVER\SQLbackups\MyDb1.bak'
    WITH FILE=1, NORECOVERY
GO
RESTORE LOG MyDb2
    FROM DISK = N'\\FILESERVER\SQLbackups\MyDb2.bak'
    WITH FILE=1, NORECOVERY
GO

-- On the server instance that hosts the secondary replica,
-- join each secondary database to the availability group:
ALTER DATABASE MyDb1 SET HADR AVAILABILITY GROUP = MyAG;
```

```
GO

ALTER DATABASE MyDb2 SET HADR AVAILABILITY GROUP = MyAG;
GO
```

[TopOfExample]

Related Tasks

To configure availability group and replica properties

- Set the Availability Mode of an Availability Replica (SQL Server)
- Set the Failover Mode of an Availability Replica (SQL Server)
- Create or Configure an Availability Group Listener (SQL Server)
- Configure the Flexible Failover Policy to Control Conditions for Automatic Failover (AlwaysOn Availability Groups)
- Specify the Endpoint URL When Adding or Modifying an Availability Replica (AlwaysOn Availability Groups)
- Configure Backup on Availability Replicas (SQL Server)
- Configure Connection Access on an Availability Replica (SQL Server)
- Configure Read-Only Routing on an Availability Group (SQL Server)
- Change the Session-Timeout Period for an Availability Replica (SQL Server)

To complete availability group configuration

- Join a Secondary Replica to an Availability Group (SQL Server)
- Manually Prepare a Secondary Database for an Availability Group (SQL Server)
- Join a Secondary Database to an Availability Group (SQL Server)
- Create or Configure an Availability Group Listener (SQL Server)

Alternative ways to create an availability group

- Use the New Availability Group Wizard (SQL Server Management Studio)
- Use the New Availability Group Dialog Box (SQL Server Management Studio)
- Create and Configure an Availability Group (SQL Server PowerShell)

To enable AlwaysOn Availability Groups

- Enable and Disable the AlwaysOn Availability Groups Feature (SQL Server)

To configure a database mirroring endpoint

- Create a Database Mirroring Endpoint for AlwaysOn Availability Groups (SQL Server PowerShell)
- Create a Mirroring Endpoint for Windows Authentication (Transact-SQL)
- Use Certificates for a Database Mirroring Endpoint
- Specify the Endpoint URL When Adding or Modifying an Availability Replica (SQL Server)

To troubleshoot AlwaysOn Availability Groups configuration

- Troubleshoot AlwaysOn Availability Groups Configuration (SQL Server)
- Troubleshoot a Failed Add-File Operation (AlwaysOn Availability Groups)

Related Content

- Microsoft SQL Server AlwaysOn Solutions Guide for High Availability and Disaster Recovery
- SQL Server AlwaysOn Team Blog: The official SQL Server AlwaysOn Team Blog

See Also

Database Mirroring Endpoint

Overview of AlwaysOn Availability Groups

Client Connectivity and Application Failover (AlwaysOn Availability Groups)

Prerequisites, Restrictions, and Recommendations (AlwaysOn Availability Groups)

Create an Availability Group (SQL Server PowerShell)

This topic describes how to use PowerShell cmdlets to create and configure an AlwaysOn availability group by using PowerShell in SQL Server 2012. An *availability group* defines a set of user databases that will fail over as a single unit and a set of failover partners, known as *availability replicas*, which support failover.

Note
For an introduction to availability groups, see Overview of AlwaysOn Availability Groups (SQL Server).

- **Before you begin:**

  Prerequisites, Restrictions, and Recommendations

  Security

  Summary of Tasks and Corresponding PowerShell Cmdlets

  To Set Up and Use the SQL Server PowerShell Provider
- **To create and configure an availability group, using:** Using PowerShell to Create and Configure an Availability Group
- **Examples:** Using PowerShell to Create an Availability Group
- Related Tasks
- Related Content

Note
As an alternative to using PowerShell cmdlets, you can use the Create Availability Group wizard or Transact-SQL. For more information, see Using the New Availability Group

Wizard (SQL Server Management Studio) or Creating and Configuring an Availability Group (Transact-SQL).

Before You Begin

We strongly recommend that you read this section before attempting to create your first availability group.

Prerequisites, Restrictions, and Recommendations

- Before creating an availability group, verify that the host instances of SQL Server each resides on a different Windows Server Failover Clustering (WSFC) node of a single WSFC failover cluster. Also, verify that your server instances met the other server-instance prerequisites and that all of the other AlwaysOn Availability Groups requirements are meet and that you are aware of the recommendations. For more information, we strongly recommend that you read Prerequisites, Restrictions, and Recommendations (AlwaysOn Availability Groups).

Security

Permissions

Requires membership in the **sysadmin** fixed server role and either CREATE AVAILABILITY GROUP server permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.

Summary of Tasks and Corresponding PowerShell Cmdlets

The following table lists the basic tasks involved in configuring an availability group and indicates those that are supported by PowerShell cmdlets. The AlwaysOn Availability Groups tasks must be performed in the sequence in which they are presented in the table.

Task	PowerShell Cmdlets (if Available) or Transact-SQL Statement	Where to Perform Task*
Create database mirroring endpoint (once per SQL Server instance)	**New-SqlHadrEndPoint**	Execute on each server instance that lacks database mirroring endpoint. **Note** To alter an existing database mirroring endpoint, use **Set-SqlHadrEndpoint**.
Create availability group	First, use the **New-SqlAvailabilityReplica** cmdlet with the **-AsTemplate**	Execute on the server instance that is to host the initial primary replica.

Task	PowerShell Cmdlets (if Available) or Transact-SQL Statement	Where to Perform Task[*]
	parameter to create an in-memory availability-replica object for each of the two availability replicas that you plan to include in the availability group. Then, create the availability group by using the **New-SqlAvailabilityGroup** cmdlet and referencing your availability-replica objects.	
Join secondary replica to availability group	**Join-SqlAvailabilityGroup**	Execute on each server instance that is hosts a secondary replica.
Prepare the secondary database	**Backup-SqlDatabase** and **Restore-SqlDatabase**	Create backups on the server instance that hosts the primary replica. Restore backups on each server instance that hosts a secondary replica, using the **NoRecovery** restore parameter. If the file paths differ between the computers that host the primary replica and the target secondary replica, also use the **RelocateFile** restore parameter.
Start data synchronization by joining each secondary database to availability group	**Add-SqlAvailabilityDatabase**	Execute on each server instance that hosts a secondary replica.

[*] To perform a given task, change directory (**cd**) to the indicated server instance or instances.

To Set Up and Use the SQL Server PowerShell Provider

- SQL Server PowerShell Provider
- Get Help SQL Server PowerShell

Using PowerShell to Create and Configure an Availability Group

Note
To view the syntax and an example of a given cmdlet, use the **Get-Help** cmdlet in the SQL Server PowerShell environment. For more information, see Get Help SQL Server PowerShell.

1. Change directory (**cd**) to the server instance that is to host the primary replica.
2. Create an in-memory availability-replica object for the primary replica.
3. Create an in-memory availability-replica object for each of the secondary replicas.
4. Create the availability group.

   Note
   The maximum length for an availability group name is 128 characters.

5. Join the new secondary replica to the availability group. For more information, see Joining a Secondary Replica to an Availability Group (SQL Server).
6. For each database in the availability group, create a secondary database by restoring recent backups of the primary database, using RESTORE WITH NORECOVERY.
7. Join every new secondary database to the availability group. For more information, see Joining a Secondary Replica to an Availability Group (SQL Server).
8. Optionally, use the Windows **dir** command to verify the contents of the new availability group.

Example: Using PowerShell to Create an Availability Group

The following PowerShell example creates and configures a simple availability group named `MyAG` with two availability replicas and one availability database. The example:

1. Backs up `MyDatabase` and its transaction log.
2. Restores `MyDatabase` and its transaction log, using the **-NoRecovery** option.
3. Creates an in-memory representation of the primary replica, which will be hosted by the local instance of SQL Server (named `PrimaryComputer\Instance`).
4. Creates an in-memory representation of the secondary replica, which will be hosted by an instance of SQL Server (named `SecondaryComputer\Instance`).
5. Creates an availability group named `MyAG`.
6. Joins the secondary replica to the availability group.
7. Joins the secondary database to the availability group.

```
# Backup my database and its log on the primary

Backup-SqlDatabase `

    -Database "MyDatabase" `

    -BackupFile "\\share\backups\MyDatabase.bak" `
    -ServerInstance "PrimaryComputer\Instance"

Backup-SqlDatabase `
    -Database "MyDatabase" `
    -BackupFile "\\share\backups\MyDatabase.log" `
    -ServerInstance "PrimaryComputer\Instance" `
    -BackupAction Log

# Restore the database and log on the secondary (using NO RECOVERY)
Restore-SqlDatabase `
    -Database "MyDatabase" `
    -BackupFile "\\share\backups\MyDatabase.bak" `
    -ServerInstance "SecondaryComputer\Instance" `
    -NoRecovery

Restore-SqlDatabase `
    -Database "MyDatabase" `
    -BackupFile "\\share\backups\MyDatabase.log" `
    -ServerInstance "SecondaryComputer\Instance" `
    -RestoreAction Log `
    -NoRecovery

# Create an in-memory representation of the primary replica.
$primaryReplica = New-SqlAvailabilityReplica `
    -Name "PrimaryComputer\Instance" `
    -EndpointURL "TCP://PrimaryComputer.domain.com:5022" `
    -AvailabilityMode "SynchronousCommit" `
    -FailoverMode "Automatic" `
    -Version 11 `
    -AsTemplate

# Create an in-memory representation of the secondary replica.
$secondaryReplica = New-SqlAvailabilityReplica `
```

```
    -Name "SecondaryComputer\Instance" `
    -EndpointURL "TCP://SecondaryComputer.domain.com:5022" `
    -AvailabilityMode "SynchronousCommit" `
    -FailoverMode "Automatic" `
    -Version 11 `
    -AsTemplate

# Create the availability group
New-SqlAvailabilityGroup `
    -Name "MyAG" `
    -Path "SQLSERVER:\SQL\PrimaryComputer\Instance" `
    -AvailabilityReplica @($primaryReplica,$secondaryReplica) `
    -Database "MyDatabase"

# Join the secondary replica to the availability group.
Join-SqlAvailabilityGroup -Path "SQLSERVER:\SQL\SecondaryComputer\Instance" -
Name "MyAG"

# Join the secondary database to the availability group.
Add-SqlAvailabilityDatabase -Path
"SQLSERVER:\SQL\SecondaryComputer\Instance\AvailabilityGroups\MyAG" -Database
"MyDatabase"
```

Related Tasks

To configure a server instance for AlwaysOn Availability Groups

- Enable and Disable the AlwaysOn Availability Groups Feature (SQL Server)
- Create a Database Mirroring Endpoint for AlwaysOn Availability Groups (SQL Server PowerShell)

To configure availability group and replica properties

- Set the Availability Mode of an Availability Replica (SQL Server)
- Set the Failover Mode of an Availability Replica (SQL Server)
- Create or Configure an Availability Group Listener (SQL Server)
- Configure the Flexible Failover Policy to Control Conditions for Automatic Failover (AlwaysOn Availability Groups)
- Specify the Endpoint URL When Adding or Modifying an Availability Replica (AlwaysOn Availability Groups)

- Configure Backup on Availability Replicas (SQL Server)
- Configure Connection Access on an Availability Replica (SQL Server)
- Configure Read-Only Routing on an Availability Group (SQL Server)
- Change the Session-Timeout Period for an Availability Replica (SQL Server)

To complete availability group configuration

- Join a Secondary Replica to an Availability Group (SQL Server)
- Manually Prepare a Secondary Database for an Availability Group (SQL Server)
- Join a Secondary Database to an Availability Group (SQL Server)
- Create or Configure an Availability Group Listener (SQL Server)

Alternative ways to create an availability group

- Use the New Availability Group Wizard (SQL Server Management Studio)
- Use the New Availability Group Dialog Box (SQL Server Management Studio)
- Create and Configure an Availability Group (Transact-SQL)

To troubleshoot AlwaysOn Availability Groups configuration

- Troubleshoot AlwaysOn Availability Groups Configuration (SQL Server)
- Troubleshoot a Failed Add-File Operation (AlwaysOn Availability Groups)

Related Content

- Microsoft SQL Server AlwaysOn Solutions Guide for High Availability and Disaster Recovery
- SQL Server AlwaysOn Team Blog: The official SQL Server AlwaysOn Team Blog

See Also

Database Mirroring Endpoint

Overview of AlwaysOn Availability Groups (SQL Server)

Specify the Endpoint URL When Adding or Modifying an Availability Replica

To host an availability replica for an availability group, a server instance must possess a database mirroring endpoint. The server instance uses this endpoint to listen for AlwaysOn Availability Groups messages from availability replicas hosted by other server instances. To define an availability replica for an availability group, you must specify the endpoint URL of the server instance that will host the replica. The *endpoint URL* identifies the transport protocol of the database mirroring endpoint—TCP, the system address of the server instance, and the port number associated with the endpoint.

Note

The term "endpoint URL" is synonymous with the term "server network address" used by the database mirroring user interface and documentation.

- Syntax for an Endpoint URL
- Finding the Fully Qualified Domain Name of A System
- Related Tasks
- Related Content

Syntax for an Endpoint URL

The syntax for an endpoint URL is of the form:

TCP**://**<*system-address*>**:**<*port*>

where

- <*system-address*> is a string that unambiguously identifies the target computer system. Typically, the server address is a system name (if the systems are in the same domain), a fully qualified domain name, or an IP address:
  - Because the nodes of Windows Server Failover Clustering (WSFC) cluster are the same domain, you can use the name of the computer system; for example, `SYSTEM46`.
  - To use an IP address, it must be unique in your environment. We recommend that you use an IP address only if it is static. The IP address can be IP Version 4 (IPv4) or IP Version 6 (IPv6). An IPv6 address must be enclosed within square brackets, for example: **[**<*IPv6_address*>**]**.

    To learn the IP address of a system, at the Windows command prompt, enter the **ipconfig** command.
  - The fully qualified domain name is guaranteed to work. This is a locally defined address string that takes different forms in different places. Often, but not always, a fully qualified domain name is a compound name that includes the computer name and a series of period-separated domain segments of the form:

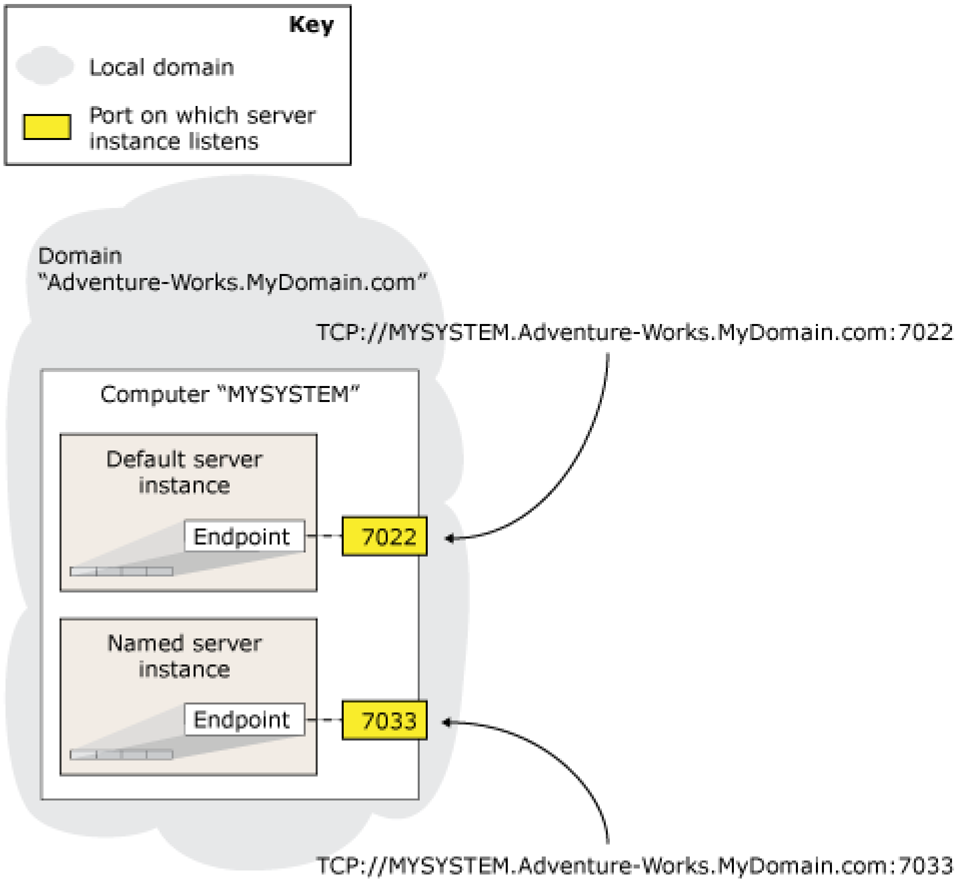
    *computer_name***.***domain_segment*[...**.***domain_segment*]

    where *computer_name i*s the network name of the computer running the server instance, and *domain_segment*[...**.***domain_segment*] is the remaining domain information of the server; for example: `localinfo.corp.Adventure-Works.com`.

    The content and number of domain segments are determined within the company or organization. For more information, see Finding the Fully Qualified Domain Name, later in this topic.
- <*port*> is the port number used by the mirroring endpoint of the partner server instance.

  A database mirroring endpoint can use any available port on the computer system. Each port number must be associated with only one endpoint, and each endpoint is associated with a single server instance; thus, different server instances on the same server listen on different endpoints with different ports. Therefore, the port you specify in the endpoint URL when you specify an availability replica will always direct incoming messages to the server instance whose endpoint is associated with that port.

IIn the endpoint URL, only the number of the port identifies the server instance that is associated with the mirroring endpoint on the target computer. The following figure illustrates the endpoint URLs of two server instances on a single computer. The default instance uses port `7022` and the named instance uses port `7033`. The endpoint URL for these two server instances are, respectively: `TCP://MYSYSTEM.Adventure-works.MyDomain.com:7022` and `TCP://MYSYSTEM.Adventure-works.MyDomain.com:7033`. Note that the address does not contain the name of the server instance.


Key
Local domain
Port on which server instance listens
Domain "Adventure-Works.MyDomain.com"
TCP://MYSYSTEM.Adventure-Works.MyDomain.com:7022
Computer "MYSYSTEM"
Default server instance
Endpoint
7022
Named server instance
Endpoint
7033
TCP://MYSYSTEM.Adventure-Works.MyDomain.com:7033

To identify the port currently associated with database mirroring endpoint of a server instance, use the following Transact-SQL statement:

```
SELECT type_desc, port FROM sys.TCP_endpoints
```

Find the row whose **type_desc** value is "DATABASE_MIRRORING," and use the corresponding port number.

Examples

A. Using a system name

The following endpoint URL specifies a system name, `SYSTEM46`, and port `7022`.

```
TCP://SYSTEM46:7022
```

B. Using a fully qualified domain name

The following endpoint URL specifies a fully qualified domain name, `DBSERVER8.manufacturing.Adventure-Works.com`, and port `7024`.

```
TCP://DBSERVER8.manufacturing.Adventure-Works.com:7024
```

C. Using IPv4

The following endpoint URL specifies an IPv4 address, `10.193.9.134`, and port `7023`.

```
TCP://10.193.9.134:7023
```

D. Using IPv6

The following endpoint URL contains an IPv6 address, `2001:4898:23:1002:20f:1fff:feff:b3a3`, and port `7022`.

```
TCP://[2001:4898:23:1002:20f:1fff:feff:b3a3]:7022
```

Finding the Fully Qualified Domain Name of A System

To find the fully qualified domain name of a system, at the Windows command prompt on that system, enter:

IPCONFIG /ALL

To form the fully qualified domain name, concatenate the values of<*host_name*> and <*Primary_Dns_Suffix*> as follows:

<*host_name*>**.**<*Primary_Dns_Suffix*>

For example, the IP configuration

```
Host Name  .  .  .  .  .  .  : MYSERVER
Primary Dns Suffix  .  .  .  : mydomain.Adventure-Works.com
```

equates to the following fully qualified domain name:

```
MYSERVER.mydomain.Adventure-Works.com
```

Note

If you need more information about a fully qualified domain name, see your system administrator.

Related Tasks

To Configure a Database Mirroring Endpoint

- Create a Database Mirroring Endpoint for AlwaysOn Availability Groups (SQL Server PowerShell)
- Create a Mirroring Endpoint for Windows Authentication (Transact-SQL)
- Use Certificates for a Database Mirroring Endpoint

    - Allow Database Mirroring to Use Certificates for Outbound Connections (Transact-SQL)
    - Allow a Database Mirroring Endpoint to Use Certificates for Inbound Connections (Transact-SQL)
- Specify a Server Network Address (Database Mirroring)
- Specify the Endpoint URL When Adding or Modifying an Availability Replica (SQL Server)
- Troubleshoot AlwaysOn Availability Groups Configuration (SQL Server)

To View Information About the Database Mirroring Endpoint

- sys.database_mirroring_endpoints (Transact-SQL)

To add an availability replica

- Add a Secondary Replica to an Availability Group (SQL Server)
- Join a Secondary Replica to an Availability Group (SQL Server)

Related Content

- Microsoft SQL Server AlwaysOn Solutions Guide for High Availability and Disaster Recovery

See Also

Creation and Configuration of Availability Groups (SQL Server)

Overview of AlwaysOn Availability Groups

CREATE ENDPOINT (Transact-SQL)

Join a Secondary Replica to an Availability Group

This topic describes how to join a secondary replica to an AlwaysOn availability group by using SQL Server Management Studio, Transact-SQL, or PowerShell in SQL Server 2012. After a secondary replica is added to an AlwaysOn availability group, the secondary replica must be joined to the availability group. The join-replica operation must be performed on the instance of SQL Server that is hosting the secondary replica.

- **Before you begin:**

  Prerequisites

  Security
- **To prepare a secondary database, using:**

  SQL Server Management Studio

  Transact-SQL

  PowerShell
- **Follow Up:** Configure Secondary Databases

Before You Begin

Prerequisites

- The primary replica of the availability group must currently be online.
- You must be connected to the server instance that hosts a secondary replica that has not yet have been joined to the availability group.
- The local server instance must be able to connect to the database mirroring endpoint of the server instance that is hosting the primary replica.

Important
If any prerequisite is not met, the join operation fails. After a failed join attempt, you might need to connect to the server instance that hosts the primary replica to remove and re-add the secondary replica before you can join it to the availability group. For more information, see Remove a Secondary Replica from an Availability Group (SQL Server) and Add a Secondary Replica to an Availability Group (SQL Server).

Security

Permissions

Requires ALTER AVAILABILITY GROUP permission on the availability group, CONTROL AVAILABILITY GROUP permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.

Using SQL Server Management Studio

To join an availability replica to an availability group

1. In Object Explorer, connect to the server instance that hosts the secondary replica, and click the server name to expand the server tree.
2. Expand the **AlwaysOn High Availability** node and the **Availability Groups** node.
3. Select the availability group of the secondary replica to which you are connected.
4. Right-click the secondary replica, and click **Join to Availability Group**.
5. This opens the **Join Replica to Availability Group** dialog box.
6. To join the secondary replica to the availability group, click **OK**.

Using Transact-SQL

To join an availability replica to an availability group

1. Connect to the server instance that hosts the secondary replica.
2. Use the ALTER AVAILABILITY GROUP statement, as follows:

   ALTER AVAILABILITY GROUP group_name JOIN

   where group_name is the name of the availability group.

   The following example, joins the secondary replica to the `MyAG` availability group.

   ```
   ALTER AVAILABILITY GROUP MyAG JOIN;
   ```

   Note

To see this Transact-SQL statement used in context, see Example: Setting Up an Availability Group Using Windows Authentication (Transact-SQL).

Using PowerShell

To join an availability replica to an availability group

In the SQL Server PowerShell provider:

1. Change directory (**cd**) to the server instance that hosts the secondary replica.
2. Join the secondary replica to the availability group by executing the **Join-SqlAvailabilityGroup** cmdlet with the name of the availability group.

   For example, the following command joins a secondary replica hosted by the server instance located at the specified path to the availability group named `MyAg`. This server instance must host a secondary replica in this availability group.

   ```
   Join-SqlAvailabilityGroup -Path
   SQLSERVER:\SQL\SecondaryServer\InstanceName -Name 'MyAg'
   ```

   Note
   To view the syntax of a cmdlet, use the **Get-Help** cmdlet in the SQL Server PowerShell environment. For more information, see Get Help SQL Server PowerShell.

To set up and use the SQL Server PowerShell provider

- SQL Server PowerShell Provider

Follow Up: Configure Secondary Databases

For every database in the availability group, you need a secondary database on the server instance that is hosting the secondary replica. You can configure secondary databases either before or after you join a secondary replica to an availability group, as follows:

1. Restore recent database and log backups of each primary database onto the server instance that hosts the secondary replica, using RESTORE WITH NORECOVERY for every restore operation. For more information, see Prepare a Secondary Database for an Availability Group (SQL Server)).
2. Join each secondary database to the availability group. For more information, see Join a Secondary Database to an Availability Group (SQL Server).

See Also

Creation and Configuration of Availability Groups (AlwaysOn Availability Groups)

AlwaysOn Availability Groups

Troubleshooting AlwaysOn Availability Groups Configuration (SQL Server)

Start Data Movement on an AlwaysOn Secondary Database

This topic contains information about how to start data synchronization after you add a database to an AlwaysOn availability group. For each new primary replica, secondary databases must be prepared on the server instances that host the secondary replicas. Then each of these secondary databases must be manually joined to the availability group.

Note
If the file paths are identical on every server instance that hosts an availability replica for an availability group, the New Availability Group Wizard, Add Replica to Availability Group Wizard, or Add Database to Availability Group Wizard might be able to automatically start data synchronization for you.

To start data synchronization manually, you need to connect, in turn, to each server instance that is hosting a secondary replica for the availability group and complete the following steps:

1. Restore current backups of each primary database and its transaction log (using RESTORE WITH NORECOVERY). You can use either of the following alternative approaches:
   - Manually restore a recent database backup of the primary database using RESTORE WITH NORECOVERY, and then restore each subsequent log backup using RESTORE WITH NORECOVERY. Perform this restore sequence on every server instance that hosts a secondary replica for the availability group.

     For more information:

     Manually Prepare a Secondary Database for an Availability Group (SQL Server)
   - If you are adding one or more log shipping primary databases to an availability group, you might be able to migrate one or more of the corresponding secondary databases from log shipping to AlwaysOn Availability Groups. Migrating a log shipping secondary database requires that it use the same database name as the primary database and that it reside on a server instance that is hosting a secondary replica for the availability group. Furthermore, the availability group must be configured so that the primary replica is preferred for backups and is a candidate for performing backups (that is, that has a backup priority that is >0). Once the backup job has run on the primary database, you will need to disable the backup job, and once the restore job has run on a given secondary database, you will need to disable the restore job.

     Note
     After you have created all the secondary databases for the availability group, if you want to perform backups on secondary replicas, you will need to re-configure the automated backup preference of the availability group.

     For more information:

     Prerequisites for Migrating from Log Shipping to AlwaysOn Availability Groups (SQL Server)

     Configure Backup on Availability Replicas (SQL Server)
2. As soon as possible, join each newly prepared secondary database to the availability group.

For more information:

Join a Secondary Database to an Availability Group (SQL Server)

Related Tasks

- Use the New Availability Group Wizard
- Use the Add Replica to Availability Group Wizard
- Use the Add Database to Availability Group Wizard

See Also

AlwaysOn Availability Groups

Manually Prepare a Secondary Database for an Availability Group

This topic describes how to prepare a secondary database for an AlwaysOn availability group in SQL Server 2012 by using SQL Server Management Studio, Transact-SQL, or PowerShell. Preparing a secondary database requires two steps: (1) restoring a recent database backup of the primary database and subsequent log backups onto each server instance that hosts the secondary replica, using RESTORE WITH NORECOVERY, and (2) joining the restored database to the availability group.

Tip

If you have an existing log shipping configuration, you might be able to convert the log shipping primary database along with one or more of its secondary databases to an AlwaysOn primary database and one or more AlwaysOn secondary databases. For more information, see Prerequisites for Migrating from Log Shipping to AlwaysOn Availability Groups (SQL Server).

- **Before you begin:**

  Prerequisites and Restrictions

  Recommendations

  Security
- **To prepare a secondary database, using:**

  SQL Server Management Studio

  Transact-SQL

  PowerShell
- Related Backup and Restore Tasks
- **Follow Up:** After Preparing a Secondary Database

Before You Begin

Prerequisites and Restrictions

- Make sure that the system where you plan to place database possesses a disk drive with sufficient space for the secondary databases.

- The name of the secondary database must be the same as the name of the primary database.
- Use RESTORE WITH NORECOVERY for every restore operation.
- If the secondary database needs to reside on a different file path (including the drive letter) than the primary database, the restore command must also use the WITH MOVE option for each of the database files to specify them to the path of the secondary database.
- If you restore the database filegroup by filegroup, be sure to restore the whole database.
- After restoring the database, you must restore (WITH NORECOVERY) every log backup created since the last restored data backup.

Recommendations

- On stand-alone instances of SQL Server, we recommend that, if possible, the file path (including the drive letter) of a given secondary database be identical to the path of the corresponding primary database. This is because if you move the database files when creating a secondary database, a later add-file operation might fail on the secondary database and cause the secondary database to be suspended.
- Before preparing your secondary databases, we strongly recommend that you suspend scheduled log backups on the databases in the availability group until the initialization of secondary replicas has completed.

Security

When a database is backed up, the TRUSTWORTHY database property is set to OFF. Therefore, TRUSTWORTHY is always OFF on a newly restored database.

Permissions

BACKUP DATABASE and BACKUP LOG permissions default to members of the **sysadmin** fixed server role and the **db_owner** and **db_backupoperator** fixed database roles. For more information, see BACKUP (Transact-SQL).

When the database being restored does not exist on the server instance, the RESTORE statement requires CREATE DATABASE permissions. For more information, see RESTORE (Transact-SQL).

Using SQL Server Management Studio

Note
If the backup and restore file paths are identical between the server instance that hosts the primary replica and every instance that hosts a secondary replica, you should be able create secondary databases by using New Availability Group Wizard, Add Replica to Availability Group Wizard, or Add Database to Availability Group Wizard.

To prepare a secondary database

1. Unless you already have a recent database backup of the primary database, create a new full or differential database backup. As a best practice, place this backup and any subsequent log backups onto the recommended network share.

2. Create at least one new log backup of the primary database.
3. On the server instance that hosts the secondary replica, restore the full database backup of the primary database (and optionally a differential backup) followed by any subsequent log backups.

   On the **RESTORE DATABASE Options** page, select **Leave the database non-operational, and do not roll back the uncommitted transactions. Additional transaction logs can be restored. (RESTORE WITH NORECOVERY)**.

   If the file paths of the primary database and the secondary database differ, for example, if the primary database is on drive 'F:' but the server instance that hosts the secondary replica lacks an F: drive, include the MOVE option in your WITH clause.
4. To complete configuration of the secondary database, you need to join the secondary database to the availability group. For more information, Join a Secondary Database to an Availability Group (SQL Server).

Note
For information about how to perform these backup and restore operations, see Related Backup and Restore Tasks, later in this section.

Related Backup and Restore Tasks

To create a database backup

- Create a Full Database Backup (SQL Server)
- Create a Differential Database Backup (SQL Server)

To create a log backup

- Back Up a Transaction Log (SQL Server Management Studio)

To restore backups

- Restore a Database Backup (SQL Server Management Studio)
- Restore a Differential Database Backup (SQL Server Management Studio)
- Restore a Transaction Log Backup (SQL Server Management Studio)
- Restore a Database to a New Location (SQL Server Management Studio)

Using Transact-SQL

To prepare a secondary database

Note
For an example of this procedure, see Example (Transact-SQL), earlier in this topic.

1. Unless you have a recent full backup of the primary database, connect to the server instance that hosts the primary replica and create a full database backup. As a best practice, place this backup and any subsequent log backups onto the recommended network share.

2. On the server instance that hosts the secondary replica, restore the full database backup of the primary database (and optionally a differential backup) followed by all subsequent log backups. Use WITH NORECOVERY for every restore operation.

   If the file paths of the primary database and the secondary database differ, for example, if the primary database is on drive 'F:' but the server instance that hosts the secondary replica lacks an F: drive, include the MOVE option in your WITH clause.

3. If any log backups have been taken on the primary database since the required log backup, you must also copy these to the server instance that hosts the secondary replica and apply each of those log backups to the secondary database, starting with the earliest and always using RESTORE WITH NORECOVERY.

   Note
   A log backup would not exist if the primary database has just been created and no log backup has been taken yet or if the recovery model has just been changed from simple to full.

4. To complete configuration of the secondary database, you need to join the secondary database to the availability group. For more information, [Join a Secondary Database to an Availability Group (SQL Server)](#).

Note
For information about how to perform these backup and restore operations, see Related Backup and Restore Tasks, later in this topic.

Transact-SQL Example

The following example prepares a secondary database. This example uses the   sample database, which uses the simple recovery model by default.

1. To use the   database, modify it to use the full recovery model:

```
USE master;

GO

ALTER DATABASE MyDB1

SET RECOVERY FULL;

GO
```

2. After modifying the recovery model of the database from SIMPLE to FULL, create a full backup, which can be used to create the secondary database. Because the recovery model has just been changed, the WITH FORMAT option is specified to create a new media set. This is useful to separate the backups under the full recovery model from any previous backups made under the simple recovery model. For the purpose of this example, the backup file (C:\ .bak) is created on the same drive as the database.

   Note
   For a production database, you should always back up to a separate device.

On the server instance that hosts the primary replica (`INSTANCE01`), create a full backup of the primary database as follows:

```
BACKUP DATABASE MyDB1
    TO DISK = 'C:\MyDB1.bak'
    WITH FORMAT
GO
```

3. Copy the full backup to the server instance that hosts the secondary replica.
4. Restore the full backup, using RESTORE WITH NORECOVERY, onto the server instance that hosts the secondary replica. The restore command depends on whether the paths of primary and secondary databases are identical.
   - **If the paths are identical:**

     On the computer that hosts the secondary replica, restore the full backup as follows:

     ```
     RESTORE DATABASE MyDB1
         FROM DISK = 'C:\MyDB1.bak'
         WITH NORECOVERY
     GO
     ```
   - **If the paths differ:**

     If the path of the secondary database differs from the path of the primary database (for instance, their drive letters differ), creating the secondary database requires that the restore operation include a MOVE clause.

     Important
     If the path names of the primary and secondary databases differ, you cannot add a file. This is because on receiving the log for the add file operation, the server instance of the secondary replica attempts to place the new file in the same path as used by the primary database.

     For example, the following command restores a backup of a primary database that resides in the data directory of the default instance of SQL Server 2012, C:\Program Files\Microsoft SQL Server\MSSQL11.MSSQLSERVER\MSSQL\DATA. The restore database operation must move the database to the data directory of a remote instance of SQL Server 2012 named (*AlwaysOn1*), which hosts the secondary replica on another cluster node. There, the data and log files are restored to the *C:\Program Files\Microsoft SQL Server\MSSQL11.ALWAYSON1\MSSQL\DATA* directory . The restore operation uses WITH NORECOVERY, to leave the secondary database in the restoring database.

     ```
     RESTORE DATABASE MyDB1
       FROM DISK='C:\MyDB1.bak'
      WITH NORECOVERY,
         MOVE 'MyDB1_Data' TO
     ```

```
      'C:\Program Files\Microsoft SQL
  Server\MSSQL11.ALWAYSON1\MSSQL\DATA\MyDB1_Data.mdf',
      MOVE 'MyDB1_Log' TO
      'C:\Program Files\Microsoft SQL
  Server\MSSQL11.ALWAYSON1\MSSQL\DATA\MyDB1_Data.ldf';
  GO
```

5. After you restore the full backup, you must create a log backup on the primary database. For example, the following Transact-SQL statement backs up the log to the a backup file named *E:\MyDB1_log.bak*:

   ```
   BACKUP LOG MyDB1
     TO DISK = 'E:\MyDB1_log.bak'
   GO
   ```

6. Before you can join the database to the secondary replica, you must apply the required log backup (and any subsequent log backups).

   For example, the following Transact-SQL statement restores the first log from *C:\MyDB1.bak*:

   ```
   RESTORE LOG MyDB1
     FROM DISK = 'E:\MyDB1_log.bak'
       WITH FILE=1, NORECOVERY
   GO
   ```

7. If any additional log backups occur before the database joins the secondary replica, you must also restore all of those log backups, in sequence, to the server instance that hosts the secondary replica using RESTORE WITH NORECOVERY.

   For example, the following Transact-SQL statement restores two additional logs from *E:\MyDB1_log.bak*:

   ```
   RESTORE LOG MyDB1
     FROM DISK = 'E:\MyDB1_log.bak'
       WITH FILE=2, NORECOVERY
   GO
   RESTORE LOG MyDB1
     FROM DISK = 'E:\MyDB1_log.bak'
       WITH FILE=3, NORECOVERY
   GO
   ```

Using PowerShell

To prepare a secondary database

1. If you need to create a recent backup of the primary database, change directory (**cd**) to the server instance that hosts the primary replica.
2. Use the **Backup-SqlDatabase** cmdlet to create each of the backups.
3. Change directory (**cd**) to the server instance that hosts the secondary replica.
4. To restore the database and log backups of each primary database, use the **restore-SqlDatabase** cmdlet, specifying the **NoRecovery** restore parameter. If the file paths differ between the computers that host the primary replica and the target secondary replica, also use the **RelocateFile** restore parameter.

   Note
   To view the syntax of a cmdlet, use the **Get-Help** cmdlet in the SQL Server PowerShell environment. For more information, see Get Help SQL Server PowerShell.
5. To complete configuration of the secondary database, you need to join it to the availability group. For more information, Join a Secondary Database to an Availability Group (SQL Server).

To set up and use the SQL Server PowerShell provider

- SQL Server PowerShell Provider

Sample Backup and Restore Script and Command

The following PowerShell commands back up a full database backup and transaction log to a network share and restore those backups from that share. This example assumes that the file path to which the database is restored is the same as the file path on which the database was backed up.

```
# Create database backup

Backup-SqlDatabase -Database "MyDB1" -BackupFile "\\share\backups\MyDB1.bak"
-ServerInstance "SourceMachine\Instance"

# Create log backup

Backup-SqlDatabase -Database "MyDB1" -BackupAction "Log" -BackupFile
"\\share\backups\MyDB1.trn" -ServerInstance "SourceMachine\Instance"

# Restore database backup

Restore-SqlDatabase -Database "MyDB1" -BackupFile "\\share\backups\MyDB1.bak"
-NoRecovery -ServerInstance "DestinationMachine\Instance"

# Restore log backup

Restore-SqlDatabase -Database "MyDB1" -BackupFile "\\share\backups\MyDB1.trn"
-RestoreAction "Log" -NoRecovery –ServerInstance
"DestinationMachine\Instance"
```

Follow Up: After Preparing a Secondary Database

To complete configuration of the secondary database, join the newly restored database to the availability group. For more information, see Join a Secondary Database to an Availability Group (SQL Server).

See Also

Overview (AlwaysOn Availability Groups)

BACKUP (Transact-SQL)

RESTORE Arguments (Transact-SQL)

RESTORE (Transact-SQL)

Troubleshooting a Failed Add-File Operation (AlwaysOn Availability Groups)

Join a Secondary Database to an Availability Group

This topic explains how to join a secondary database to an AlwaysOn availability group by using SQL Server Management Studio, Transact-SQL, or PowerShell in SQL Server 2012. After you prepare a secondary database for a secondary replica, you need to join the database to the availability group as soon as possible. This will start data movement from the corresponding primary database to the secondary database.

- **Before you begin:**

  Prerequisites

  Security

- **To prepare a secondary database, using:**

  SQL Server Management Studio

  Transact-SQL

  PowerShell

Note

For information about what happens after a secondary database joins the group, see AlwaysOn Availability Groups (SQL Server).

Before You Begin

Prerequisites

- You must be connected to the server instance that hosts the secondary replica.
- The secondary replica must already be joined to the availability group. For more information, see Join a Secondary Replica to an Availability Group (SQL Server).
- The secondary database must have been prepared recently. For more information, see Prepare a Secondary Database for an Availability Group (SQL Server).

Security

Permissions

Requires ALTER AVAILABILITY GROUP permission on the availability group, CONTROL AVAILABILITY GROUP permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.

Using SQL Server Management Studio

To join a secondary database to an availability group

1. In Object Explorer, connect to the server instance that hosts the secondary replica, and expand the server tree.
2. Expand the **AlwaysOn High Availability** node and the **Availability Groups** node.
3. Expand the availability group that you want to change, and expand the **Availability Databases** node.
4. Right-click the database, and click **Join to Availability Group**.
5. This opens the **Join Databases to Availability Group** dialog box. Verify the availability group name, which is displayed on the title bar, and database name or names displayed in the grid, and click **OK**, or click **Cancel**.

Using Transact-SQL

To join a secondary database to an availability group

1. Connect to the server instance that hosts the secondary replica.
2. Use the SET HADR clause of the ALTER DATABASE statement, as follows:

   ALTER DATABASE database_name SET HADR AVAILABILITY GROUP = group_name

   where database_name is the name of a database to be joined and group_name is the name of the availability group.

   The following example joins the secondary database, `Db1`, to the local secondary replica of the `MyAG` availability group.

   ```
   ALTER DATABASE Db1 SET HADR AVAILABILITY GROUP = MyAG;
   ```

   Note
   To see this Transact-SQL statement used in context, see Example: Setting Up an Availability Group Using Windows Authentication (Transact-SQL).

Using PowerShell

To join a secondary database to an availability group

1. Change directory (**cd**) to the server instance that hosts the secondary replica.
2. Use the **Add-SqlAvailabilityDatabase** cmdlet to join one or more secondary databases to the availability group.

   For example, the following command joins a secondary database, `Db1`, to the availability group `MyAG` on one of the server instances that hosts a secondary replica.

```
Add-SqlAvailabilityDatabase `
-Path
SQLSERVER:\SQL\SecondaryServer\InstanceName\AvailabilityGroups\MyAG `
-Database "Db1"
```

Note
To view the syntax of a cmdlet, use the **Get-Help** cmdlet in the SQL Server PowerShell environment. For more information, see Get Help SQL Server PowerShell.

To set up and use the SQL Server PowerShell provider

- SQL Server PowerShell Provider

Related Tasks

- Join a Secondary Replica to an Availability Group (SQL Server)
- Manually Prepare a Secondary Database for an Availability Group (SQL Server)

See Also

ALTER AVAILABILITY GROUP (Transact-SQL)

AlwaysOn Availability Groups

Troubleshoot AlwaysOn Availability Groups Configuration (SQL Server)

Management of Logins and Jobs for the Databases of an Availability Group

Logins and jobs that are associated with a database in an AlwaysOn availability group must be reproduced on every instance of SQL Server that hosts an availability replica for the availability group. If you are using partially contained databases, you can configure contained users in the databases, and for these users, you do not need to create logins on the server instances that host a secondary replica. For a non-contained availability database, you will need to create both logins and any relevant jobs on the server instances that host the availability replicas. Backup jobs require special consideration. The server instances that host the replicas of an availability group might be configured differently, with different tape drive letters or such. The jobs for each availability replica must allow for any such differences.

You should routinely maintain the same set of user logins and jobs on every primary database and its corresponding secondary databases.

Note
A database user for which the 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. If a user is orphaned on a given server instance,

you can set up the user logins at any time. For more information, see Troubleshooting Orphaned Users.

Related Tasks

- Create a Login (SQL Server Database Engine)
- Create a Database User.
- Create a Job

See Also

Overview of AlwaysOn Availability Groups (SQL Server)

Partially Contained Databases

Create Jobs

Troubleshoot AlwaysOn Availability Groups Configuration

This topic provides information to help you troubleshoot typical problems with configuring server instances for AlwaysOn Availability Groups. Typical configuration problems include AlwaysOn Availability Groups is disabled, accounts are incorrectly configured, the database mirroring endpoint does not exist, the endpoint is inaccessible (SQL Server Error 1418), network access does not exist, and a join database command fails (SQL Server Error 35250).

Note
Ensure that you are meeting the AlwaysOn Availability Groups prerequisites. For more information, see Prerequisites, Restrictions, and Recommendations (AlwaysOn Availability Groups).

Issue	Summary
AlwaysOn Availability Groups Is Not Enabled	If an instance of SQL Server is not enabled for AlwaysOn Availability Groups, the instance does not support availability group creation and cannot host any availability replicas.
Accounts	Discusses requirements for correctly configuring the accounts under which SQL Server is running.
Endpoints	Discusses how to diagnose issues with the database mirroring endpoint of a server instance.
System name	Summarizes the alternatives for specifying the system name of a server instance in an

Issue	Summary
	endpoint URL.
Network access	Documents the requirement that each server instance that is hosting an availability replica must be able to access the port of each of the other server instances over TCP.
Endpoint Access (SQL Server Error 1418)	Contains information about this SQL Server error message.
Join Database Fails (SQL Server Error 35250)	Discusses the possible causes and resolution of a failure to join secondary databases to an availability group because the connection to the primary replica is not active.

AlwaysOn Availability Groups Is Not Enabled

The AlwaysOn Availability Groups feature must be enabled on each of the instances of SQL Server 2012. For more information, see Enabling and Disabling the AlwaysOn Availability Groups Feature (SQL Server).

Accounts

The accounts under which SQL Server is running must be correctly configured.

1. Do the accounts have the correct permissions?
    a. If the partners run as the same domain user account, the correct user logins exist automatically in both **master** databases. This simplifies the security configuration the database and is recommended.
    b. If two server instances run as different accounts, the login each account must be created in **master** on the remote server instance, and that login must be granted CONNECT permissions to connect to the database mirroring endpoint of that server instance. For more information, see Setting Up Login Accounts for Database Mirroring.
2. If SQL Server is running as a built-in account, such as Local System, Local Service, or Network Service, or a nondomain account, you must use certificates for endpoint authentication. If your service accounts are using domain accounts in the same domain, you can choose to grant CONNECT access for each service account on all the replica locations or you can use certificates. For more information, see Using Certificates for Database Mirroring.

Endpoints

Endpoints must be correctly configured.

1. Make sure that each instance of SQL Server that is going to host an availability replica (each *replica location*) has a database mirroring endpoint. To determine whether a database mirroring endpoint exists on a given server instance, use the sys.database_mirroring_endpoints catalog view. For more information, see either How to: Create a Mirroring Endpoint for Windows Authentication (Transact-SQL) or How to: Allow Database Mirroring to Use Certificates for Outbound Connections (Transact-SQL).
2. Check that the port numbers are correct.

   To identify the port currently associated with database mirroring endpoint of a server instance, use the following Transact-SQL statement:

   ```
   SELECT type_desc, port FROM sys.tcp_endpoints;
   GO
   ```
3. For AlwaysOn Availability Groups setup issues that are difficult to explain, we recommend that you inspect each server instance to determine whether it is listening on the correct ports. For information about verifying port availability, see MSSQLSERVER_1418.
4. Make sure that the endpoints are started (STATE=STARTED). On each server instance, use the following Transact-SQL statement:

   ```
   SELECT state_desc FROM sys.database_mirroring_endpoints
   ```

   For more information about the **state_desc** column, see sys.endpoints (Transact-SQL).

   To start an endpoint, use the following Transact-SQL statement:

   ```
   ALTER ENDPOINT Endpoint_Mirroring
   STATE = STARTED
   AS TCP (LISTENER_PORT = <port_number>)
   FOR database_mirroring (ROLE = ALL);
   GO
   ```

   For more information, see ALTER ENDPOINT (Transact-SQL).
5. Make sure that the login from the other server has CONNECT permission. To determine who has CONNECT permission for an endpoint, on each server instance use the following Transact-SQL statement:

   ```
   SELECT 'Metadata Check';
   SELECT EP.name, SP.STATE,
      CONVERT(nvarchar(38), suser_name(SP.grantor_principal_id))
         AS GRANTOR,
      SP.TYPE AS PERMISSION,
      CONVERT(nvarchar(46),suser_name(SP.grantee_principal_id))
         AS GRANTEE
      FROM sys.server_permissions SP , sys.endpoints EP
   ```

```
        WHERE SP.major_id = EP.endpoint_id
        ORDER BY Permission,grantor, grantee;
    GO
```

System Name

For the system name of a server instance in an endpoint URL, you can use any name that unambiguously identifies the system. The server address can be a system name (if the systems are in the same domain), a fully qualified domain name, or an IP address (preferably, a static IP address). Using the fully qualified domain name is guaranteed to work. For more information, see Specifying the Endpoint URL When Adding or Modifying a "HADR" Availability Replica (SQL Server).

Network Access

Each server instance that is hosting an availability replica must be able to access the port of each of the other server instance over TCP. This is especially important if the server instances are in different domains that do not trust each other (untrusted domains).

Endpoint Access (SQL Server Error 1418)

This SQL Server message indicates that the server network address specified in the endpoint URL cannot be reached or does not exist, and it suggests that you verify the network address name and reissue the command. For more information, see MSSQLSERVER_1418.

Join Database Fails (SQL Server Error 35250)

This section discusses the possible causes and resolution of a failure to join secondary databases to the availability group because the connection to the primary replica is not active.

Resolution:

1. Check the firewall setting to see if whether allows the endpoint port communication between the server instances that host primary replica and the secondary replica (port 5022 by default).
2. Check whether the network service account has connect permission to the endpoint.

See Also

Database Mirroring Transport Security

Create a Mirroring Endpoint for Windows Authentication (Transact-SQL)

Specifying the Endpoint URL for an Availability Replica (SQL Server)

Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups (SQL Server)

Creation and Configuration of Availability Groups (SQL Server)

Manually Prepare a Secondary Database for an Availability Group (SQL Server)

Troubleshooting a Failed Add-File Operation (AlwaysOn Availability Groups)

Administration of an Availability Group

Managing an existing AlwaysOn availability group in SQL Server 2012 involves one or more of the following tasks:

- Altering the properties of an existing availability replica, for example to change client connection access (for configuring readable secondary replicas), changing its failover mode, availability mode, or session timeout setting.
- Adding or removing secondary replicas.
- Adding or removing a database.
- Suspending or resuming a database.
- Performing a planned manual failover (a *manual failover*) or a forced manual failover (a *forced failover*).
- Creating or configuring an availability group listener.
- Managing readable secondary replicas for a given availability group. This involves configuring one or more replicas to read-only access when running under the secondary role, and configuring read-only routing.
- Managing backups on secondary replicas for a given availability group. This involves configuring where you prefer that backup jobs run and then scripting backup jobs to implement your backup preference. you need to script backup jobs for every database in the availability group on every instance of SQL Server that hosts an availability replica.
- Deleting an availability group.

In This Topic:

- Related Tasks
- Related Content

Related Tasks

To configure an existing availability group

- Add a Secondary Replica to an Availability Group (SQL Server)
- Remove a Secondary Replica from an Availability Group (SQL Server)
- Add a Database to an Availability Group (SQL Server)
- Remove a Database from a Secondary Replica (AlwaysOn Availability Groups)
- Remove a Database from an Availability Group (AlwaysOn Availability Groups)
- Configure the Flexible Failover Policy to Control Conditions for Automatic Failover (AlwaysOn Availability Groups)

To manage an availability group

- Configure Backup on Availability Replicas (SQL Server)
- Perform a Planned Manual Failover an Availability Group (SQL Server)
- Perform a Forced Manual Failover of an Availability Group (SQL Server)

- Delete an Availability Group (SQL Server)

To manage an availability replica

- Add a Secondary Replica to an Availability Group (SQL Server)
- Join a Secondary Replica to an Availability Group (SQL Server)
- Remove a Secondary Replica from an Availability Group (SQL Server)
- Change the Availability Mode of an Availability Replica (SQL Server)
- Change the Failover Mode of an Availability Replica (SQL Server)
- Configure Backup on Availability Replicas (SQL Server)
- Configure Read-Only Access on an Availability Replica (SQL Server)
- Configure Read-Only Routing for an Availability Group (SQL Server)
- Change the Session-Timeout Period for an Availability Replica (SQL Server)

To manage an availability database

- Add a Database to an Availability Group (SQL Server)
- Join a Secondary Database to an Availability Group (SQL Server)
- Remove a Primary Database from an Availability Group (SQL Server)
- Remove a Secondary Database from an Availability Group (SQL Server)
- Suspend a Database on an Secondary Replica Location (SQL Server)
- Resume a Secondary Database on an Secondary Replica (SQL Server)

To monitor an availability group

- Monitoring of Availability Groups (SQL Server)

Related Content

- Microsoft SQL Server AlwaysOn Solutions Guide for High Availability and Disaster Recovery
- SQL Server AlwaysOn Team Blog: The official SQL Server AlwaysOn Team Blog

See Also

AlwaysOn Availability Groups (SQL Server)

Overview of AlwaysOn Availability Groups

Configuration of a Server Instance for AlwaysOn Availability Groups (SQL Server)

Creation and Configuration of Availability Groups (SQL Server)

Read-Only Access to Secondary Replicas

Backup on Secondary Replicas (AlwaysOn Availability Groups)

Availability Group Listeners, Client Connectivity, and Application Failover (SQL Server)

AlwaysOn Policies for Operational Issues with AlwaysOn Availability Groups (SQL Server)

Availability Group Monitoring (SQL Server)

AlwaysOn Availability Groups: Interoperability (SQL Server)

Overview of Transact-SQL Statements for AlwaysOn Availability Group

Overview of PowerShell Cmdlets for Availability Groups (SQL Server)

Perform a Planned Manual Failover of an Availability Group

This topic describes how to perform a manual failover without data loss (a *planned manual failover*) on an availability group by using SQL Server Management Studio, Transact-SQL, or PowerShell in SQL Server 2012. An availability group fails over at the level of an availability replica. A planned manual failover, like any AlwaysOn Availability Groups failover, transitions a secondary replica to primary role and, concurrently, transitions the former primary replica to the secondary role.

A planned manual failover, which is supported only when the primary replica and the target secondary replica are running in synchronous-commit mode and are currently synchronized, preserves all the data in the secondary databases that are joined to the availability group on the target secondary replica. Once the former primary replica transitions to the secondary role, its databases become secondary databases and begin synchronizing with the new primary databases. After they all transition into the SYNCHRONIZED state, the new secondary replica becomes eligible to serve as the target of a future planned manual failover.

Note

If the secondary and primary replicas are both configured for automatic failover mode, once the secondary replica is synchronized, it can also serve as the target for an automatic failover. For more information, see Availability Modes (AlwaysOn Availability Groups).

- **Before you begin:**

  Limitations and Restrictions

  Prerequisites and Restrictions

  Security

- **To manually fail over an availability group, using:**

  SQL Server Management Studio

  Transact-SQL

  PowerShell

- **Follow Up:** After Manually Failing Over an Availability Group

Before You Begin

Limitations and Restrictions

- A failover command returns as soon as the target secondary replica has accepted the command. However, database recovery occurs asynchronously after the availability group has finished failing over.

- Cross-database consistency across databases within the availability group is not maintained on failover.

  Note
  Cross-database transactions and distributed transactions are not supported by AlwaysOn Availability Groups. For more information, see Cross-Database Transactions Not Supported For Database Mirroring or AlwaysOn Availability Groups (SQL Server).

Prerequisites and Restrictions

- The target secondary replica and the primary replica must both be running in synchronous-commit availability mode.
- The target secondary replica must currently be synchronized with the primary replica. This requires that all the secondary databases on this secondary replica must have been joined to the availability group and be synchronized with their corresponding primary databases (that is, the local secondary databases must be SYNCHRONIZED).

  Tip
  To determine the failover readiness of an secondary replica, query the **is_failover_ready** column in the sys.dm_hadr_database_cluster_states dynamic management view, or look at the **Failover Readiness** column of the AlwaysOn Group Dashboard.
- This task is supported only on the target secondary replica. You must be connected to the server instance that hosts the target secondary replica.

Security

Permissions

Requires ALTER AVAILABILITY GROUP permission on the availability group, CONTROL AVAILABILITY GROUP permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.

Using SQL Server Management Studio

To manually fail over an availability group

1. In Object Explorer, connect to a server instance that hosts a secondary replica of the availability group that needs to be failed over, and expand the server tree.
2. Expand the **AlwaysOn High Availability** node and the **Availability Groups** node.
3. Right-click the availability group to be failed over, and select the **Failover** command.
4. This launches the Failover Availability Group Wizard. For more information, see Use the Failover Availability Group Wizard (SQL Server).

Using Transact-SQL

To manually fail over an availability group

1. Connect to the server instance that hosts the target secondary replica.
2. Use the ALTER AVAILABILITY GROUP statement, as follows:

   ALTER AVAILABILITY GROUP *group_name* FAILOVER

   where group_name is the name of the availability group.

   The following example manually fails over the *MyAg* availability group to the connected secondary replica.

   ```
   ALTER AVAILABILITY GROUP MyAg FAILOVER;
   ```

Using PowerShell

To manually fail over an availability group

1. Change directory (**cd**) to the server instance that hosts the target secondary replica.
2. Use the **Switch-SqlAvailabilityGroup** cmdlet.

   Note
   To view the syntax of a cmdlet, use the **Get-Help** cmdlet in the SQL Server 2012 PowerShell environment. For more information, see SQL Server PowerShell Help.

   The following example manually fails over the *MyAg* availability group to the secondary replica with the specified path.

   ```
   Switch-SqlAvailabilityGroup -Path
   SQLSERVER:\Sql\SecondaryServer\InstanceName\AvailabilityGroups\MyAg
   ```

To set up and use the SQL Server PowerShell provider

- Using the SQL Server PowerShell Provider
- SQL Server PowerShell Help

Follow Up: After Manually Failing Over an Availability Group

If you failed over outside of the automatic failover set of the availability group, adjust the quorum votes of the WSFC nodes to reflect your new availability group configuration. For more information, see Windows Server Failover Clusters (WSFC) with SQL Server.

See Also

AlwaysOn Availability Groups

Failover Modes (AlwaysOn Availability Groups)

Perform a Forced Manual Failover of an Availability Group (SQL Server)

Perform a Forced Manual Failover of an Availability Group

If the WSFC cluster has quorum, after losing the primary replica of an AlwaysOn availability group in SQL Server 2012, you can force the availability group to fail over, with the risk of possible data loss, to a secondary replica. This form of failover is known as a *forced failover*. After a forced failover, the secondary replica to which the availability group was failed over becomes the new primary replica. When the former primary replica becomes available, it transitions to the secondary role, and its availability databases become secondary databases and transition into the SUSPENDED state. While the databases are suspended, the database administrator can attempt to recover any lost data.

Caution
Forcing service, which might involve some data loss, is strictly for disaster recovery. Therefore, We strongly recommend that you force failover only if the primary replica is no longer running, you have no SYNCHRONIZED replica to which you can perform a manual failover. you are willing to risk losing data, and you must restore service to the availability group immediately. Note that if you issue a forced failover command on a synchronized secondary replica, the secondary replica behaves the same as for a manual failover.

- **Before you begin:**

  Limitations and Restrictions

  Prerequisites

  Recommendations

  Security
- **To force failover (with possible data loss), using:**

  SQL Server Management Studio

  Transact-SQL

  PowerShell
- **Follow Up:** Essential Tasks After a Forced Failover
- **Example Scenario:** Using a Forced Failover to Recover from a Catastrophic Failure
- Related Tasks
- Related Content

Before You Begin

Limitations and Restrictions

- Data loss is possible during the forced failover of an availability group. In addition, if the primary replica is running when you initiate a forced failover, clients might still be connected to former primary databases. Therefore, we strongly recommend that you force failover only if the primary replica is no longer running and if you are willing to risk losing data in order to restore access to databases in the availability group.
- When a database on a secondary replica is in the REVERTING or INITIALIZING state, forcing failover would cause the database to fail to start as a primary database. If the database was in the INTIAILIZGING state the you will need to apply the missing log records from a

database backup or fully restore the database from scratch. If the database was in the REVERTING state you will need to fully restore the database from backups.

- A failover command returns as soon as the target secondary replica has accepted the command. However, database recovery occurs asynchronously after the availability group has finished failing over.
- Cross-database consistency across databases within the availability group is not maintained on failover.

Note
Cross-database transactions and distributed transactions are not supported by AlwaysOn Availability Groups. For more information, see Cross-Database Transactions Not Supported For Database Mirroring or AlwaysOn Availability Groups (SQL Server).

Prerequisites

- The WSFC cluster has quorum. If the cluster lacks quorum, for information about see WSFC Quorum Failure with SQL Server.
- You must be able to connect to the server instance that hosts the target secondary replica.

Recommendations

- Do not force failover while the primary replica is still running.
- If any secondary replica is SYNCHRONIZED with the primary replica (in the FAILOVER_READY state) or the primary replica is running, perform a planned manual failover instead of a forced failover.

Tip
To determine the failover readiness of an secondary replica, query the **is_failover_ready** column in the sys.dm_hadr_database_cluster_states dynamic management view, or look at the **Failover Readiness** column of the AlwaysOn Group Dashboard.

- If possible, force fail over only to a secondary replica whose secondary databases are either in the NOT SYNCHRONIZED, SYNCHRONIZED, or SYNCHRONIZING state. For information about the implications of forcing failover when a secondary database is in the INTIAILIZGING or REVERTING state, see Limitations and Restrictions, earlier in this topic.
- Typically, the latency of a given secondary database, relative to the primary database, should be similar on different asynchronous-commit secondary replicas. However, when forcing failover, data loss can be a significant concern. Therefore, you consider taking time to determine the relative latency of the copies of the databases on different secondary replicas. To determine which copy of a given secondary database has the least latency, compare their end-of-log LSNs. A higher the end-of-log LSN indicates less latency.

Tip

To compare end-of-log LSNs, connect to each online secondary replica, in turn, and query sys.dm_hadr_database_replica_states for the **end_of_log_lsn** value of each local secondary database. Then, compare the end-of-log LSNs of the different copies of each database. Note that different databases might have their highest LSNs on different secondary replicas. In this case, the most appropriate failover target depends on the relative importance that you place on the data in the different databases. That is, for which of these databases would you most want to minimize possible data loss?

- If clients are able to connect to the original primary, a forced failover incurs some risk of split brain behavior. Before you force failover, we strongly recommend that you prevent clients from accessing the original primary replica. Otherwise, after failover is forced, the original primary databases and the current primary databases could be updated independently of the other.

Security

Permissions

Requires ALTER AVAILABILITY GROUP permission on the availability group, CONTROL AVAILABILITY GROUP permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.

Using SQL Server Management Studio

To force failover (with possible data loss)

1. In Object Explorer, connect to a server instance that hosts a secondary replica of the availability group that needs to be failed over, and expand the server tree.
2. Expand the **AlwaysOn High Availability** node and the **Availability Groups** node.
3. Right-click the availability group to be failed over, and select the **Failover** command.
4. This launches the Failover Availability Group Wizard. For more information, see Use the Failover Availability Group Wizard (SQL Server).
5. After forcing an availability group to fail over, complete the necessary follow-up steps. For more information, see Follow Up: Essential Tasks After a Forced Failover, later in this topic.

Using Transact-SQL

To force failover (with possible data loss)

1. Connect to the server instance that hosts the secondary replica to which you are forcing failover.
2. Use the ALTER AVAILABILITY GROUP statement, as follows:

   ALTER AVAILABILITY GROUP *group_name* FORCE_FAILOVER_ALLOW_DATA_LOSS

   where group_name is the name of the availability group.

   The following example forces the `AccountsAG` availability group to fail over to the local secondary replica.

```
ALTER AVAILABILITY GROUP AccountsAG FORCE_FAILOVER_ALLOW_DATA_LOSS;
```

3. After forcing an availability group to fail over, complete the necessary follow-up steps. For more information, see Follow Up: Essential Tasks After a Forced Failover, later in this topic.

Using PowerShell

To force failover (with possible data loss)

1. Change directory (**cd**) to the server instance that hosts the secondary replica to which you are forcing failover.
2. Use the **Switch-SqlAvailabilityGroup** cmdlet with the **AllowDataLoss** parameter in one of the following forms:
   - **-AllowDataLoss**

     By default **-AllowDataLoss** parameter causes **Switch-SqlAvailabilityGroup** to prompt you to remind you that forcing failover might result in the loss of uncommitted transactions and to request confirmation. To continue, enter **Y**; to cancel the operation, enter **N**.

     The following example performs a forced failover (with possible data loss) of the availability group `MyAg` to the secondary replica on the server instance named `SecondaryServer\InstanceName`. You will be prompted to confirm this operation.

     ```
     Switch-SqlAvailabilityGroup `
        -Path
     SQLSERVER:\Sql\SecondaryServer\InstanceName\AvailabilityGroups\MyAg `
        -AllowDataLoss
     ```
   - **-AllowDataLoss -Force**

     To initiate a forced failover without confirmation, specify both the **-AllowDataLoss** and **-Force** parameters. This is useful if you want to include the command in a script and run it without user interaction. However, use the **-Force** option with caution, because a forced failover might result in the loss of data from databases participating the availability group.

     The following example performs a forced failover (with possible data loss) of the availability group `MyAg` to the server instance named `SecondaryServer\InstanceName`. The **-Force** option suppresses confirmation of this operation.

     ```
     Switch-SqlAvailabilityGroup `
        -Path
     SQLSERVER:\Sql\SecondaryServer\InstanceName\AvailabilityGroups\MyAg `
        -AllowDataLoss -Force
     ```

Note

To view the syntax of a cmdlet, use the **Get-Help** cmdlet in the SQL Server PowerShell environment. For more information, see Get Help SQL Server PowerShell.

3. After forcing an availability group to fail over, complete the necessary follow-up steps. For more information, see Follow Up: Essential Tasks After a Forced Failover, later in this topic.

To set up and use the SQL Server PowerShell provider

- SQL Server PowerShell Provider

Follow Up: Essential Tasks After a Forced Failover

1. After a forced failover, the secondary replica to which you failed over becomes the new primary replica. However, to make that availability replica accessible to clients, you might need to reconfigure the WSFC quorum or adjust the availability-mode configuration of the availability group, as follows:

   - **If you failed over outside of the automatic failover set:** Adjust the quorum votes of the WSFC nodes to reflect your new availability group configuration. If the WSFC node that hosts the target secondary replica does not have a WSFC quorum vote, you might need to force WSFC quorum.

     Note
     An automatic failover set exists only if two availability replicas (including the previous primary replica) are configured for synchronous-commit mode with automatic failover.

     To adjust quorum votes

     - View Cluster Quorum NodeWeight Settings
     - Configure Cluster Quorum NodeWeight Settings
     - Force a WSFC Cluster to Start Without a Quorum

   - **If you failed over outside of the synchronous-commit failover set:** We recommend that you consider adjusting the availability mode and failover mode on the new primary replica and on remaining secondary replicas to reflect your desired synchronous-commit and automatic failover configuration.

     Note
     A synchronous-commit failover set exists only if the current primary replica is configured for synchronous-commit mode.

     To change the availability mode and failover mode

     - Set the Availability Mode of an Availability Replica (SQL Server)
     - Set the Failover Mode of an Availability Replica (SQL Server)

2. After a forced failover, all secondary databases are suspended. This includes the former primary databases, after the former primary replica comes back online and discovers that is now an secondary replica). You must manually resume each suspended database individually on the secondary replica.

When a secondary database is resumed, it initiates data synchronization with the corresponding primary database. The secondary database rolls back any log records that were never committed on the new primary database. Therefore, if you are concerned about possible data loss on the post-failover primary databases, you should attempt to create a database snapshot on the suspended databases on one the synchronous-commit secondary database.

Important
The transaction log truncation is delayed on a primary database while any of its secondary databases is suspended. Also the synchronization health of a synchronous-commit secondary replica cannot transition to HEALTHY as long as any local database remains suspended.

To create a database snapshot

- Create a Database Snapshot (Transact-SQL)

To resume an availability database

- Resume a Secondary Database on an Secondary Replica (SQL Server)

Caution
After resuming all the secondary databases, before attempting to fail over the group again, wait for every secondary database on the next failover target to enter the SYNCHRONIZING state. If any database is not yet SYNCHRONIZING, that database will be prevented from coming online as a primary database, and re-establishing data synchronization for the database might require restoring transaction logs, restoring a full database backup, or failing over back to the previous primary replica.

3. If an availability replica that failed will not be returning to the availability replica or will return too late for you to delay transaction log truncation on the new primary database, consider removing the failed replica from the availability group to avoid running out of disk space for your log files.

   To remove an secondary replica

   - Remove a Secondary Replica from an Availability Group (SQL Server)

4. If you follow a forced failover with one or more additional forced failovers, perform a log backup after each additional forced failover in the series. For information about the reason for this, see "Risks of Forcing Failover" in the "Forced Manual Failover (with Possible Data Loss)" section of Failover Modes (AlwaysOn Availability Groups).

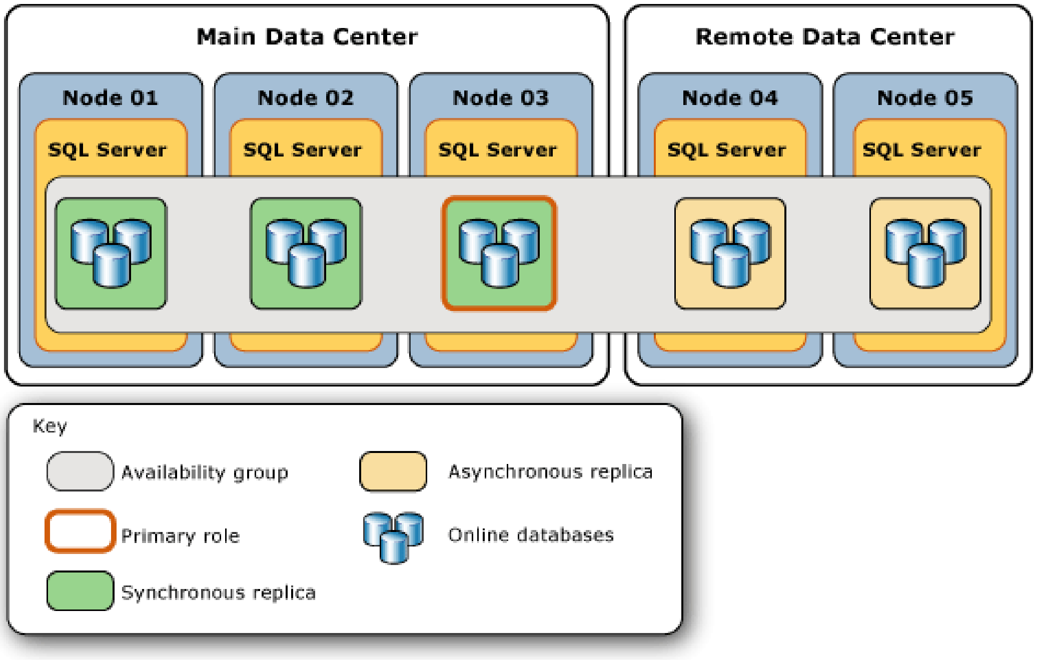
   To perform a log backup

   - Back Up a Transaction Log (SQL Server)

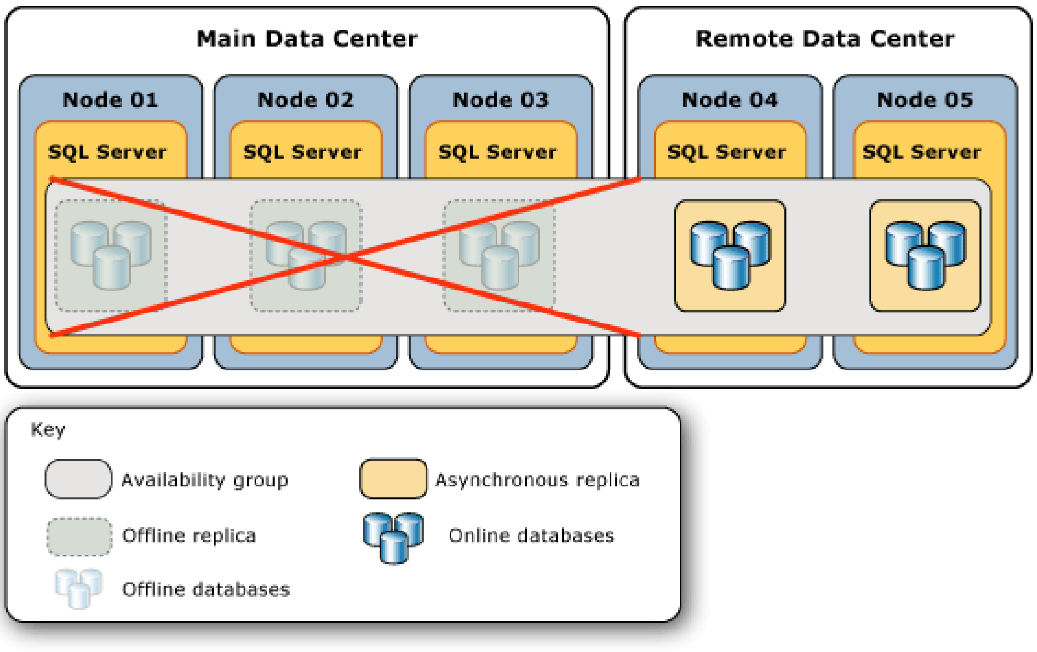
Example Scenario: Using a Forced Failover to Recover from a Catastrophic Failure

If the primary replica fails and no synchronized secondary replica is available, forcing the availability group to fail over might be an appropriate response. The appropriateness of forcing a failover depends on: (1) whether you expect the primary replica to be offline for longer than your service level agreement (SLA) tolerates, and (2) whether you are willing to risk potential data loss in order to make primary databases available quickly. If you decide that an availability group requires a forced failover, the actual forced failover is but one step of a multi-step process.

To illustrate the steps that are required to use a forced failover to recover from a catastrophic failure, this topic presents one possible disaster recovery scenario. The example scenario considers an availability group whose original topology consists of a main data center that hosts three synchronous-commit availability replicas, including the primary replica, and a remote data center that hosts two asynchronous-commit secondary replicas. The following figure illustrates the original topology of this example availability group. The availability group is hosted by a multi-subnet WSFC cluster with three nodes in the main data center (**Node 01**, **Node 02**, and **Node 03**) and two nodes in a remote data center (**Node 04** and **Node 05**).



The main data center shuts down unexpectedly. Its three availability replicas to go offline, and their databases become unavailable. The following figure illustrates the impact of this failure on the topology of the availability group.



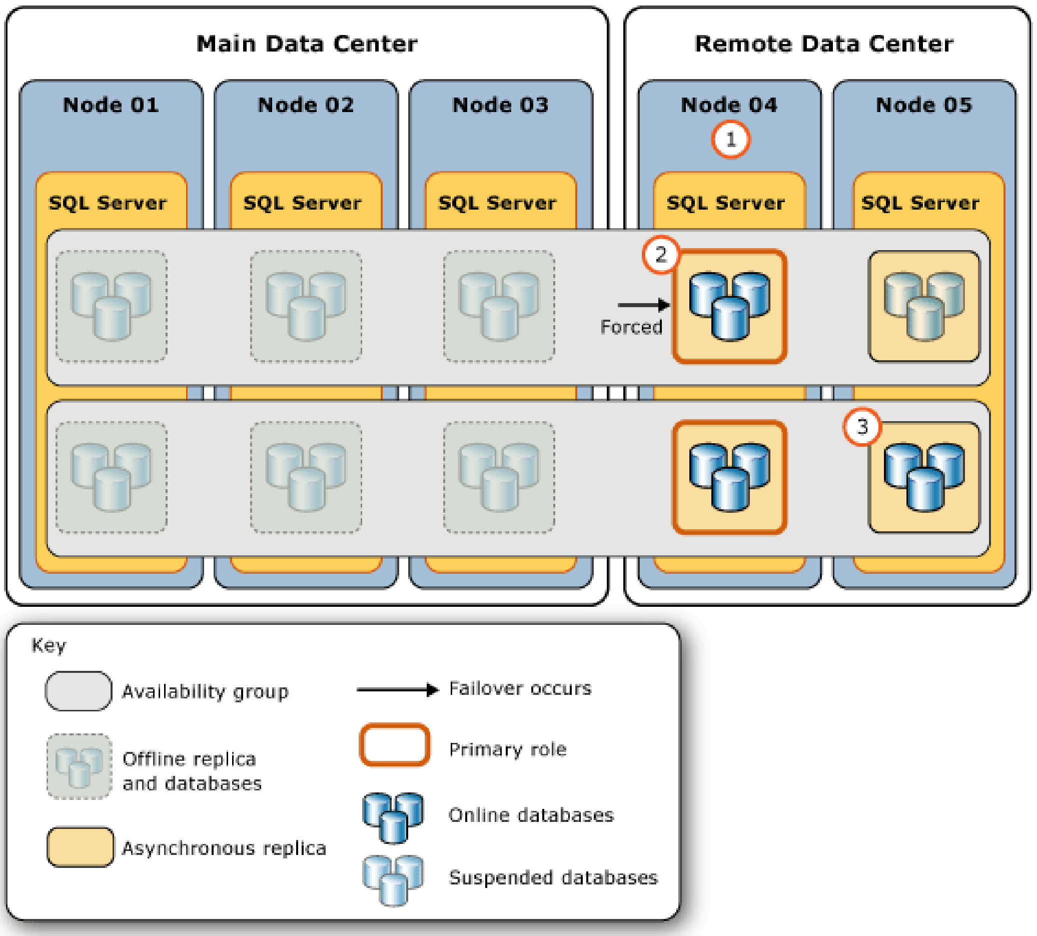
The database administrator (DBA) determines that the best possible response is to force failover of the availability group to one of the remote, asynchronous-commit secondary replicas. This example illustrates the typical steps involved when you force failover of the availability group to a remote replica and, eventually, return the availability group to its original topology.

The failure-response presented here consists of the following two phases:

- Responding to the catastrophic failure of the main data center
- Returning the Availability Group to its Original Topology

Responding to the Catastrophic Failure of the Main Data Center

The following figure illustrates the series of actions performed at the remote data center in response a catastrophic failure at the main data center.



The steps in this figure indicate the following steps:

Step	Action	Links
1.	The DBA or network administrator ensures that the WSFC cluster has a healthy quorum. In this example, quorum needs to be forced.	• WSFC Quorum Modes and Voting Configuration (SQL Server) • WSFC Disaster Recovery through Forced Quorum (SQL Server)
2.	The DBA connects to the server instance with the least latency (on **Node 04**) and performs a forced manual failover. The forced	• sys.dm_hadr_database_replica_states (Query the **end_of_log_lsn** column. For more information, see Recommendations, earlier in this topic.)

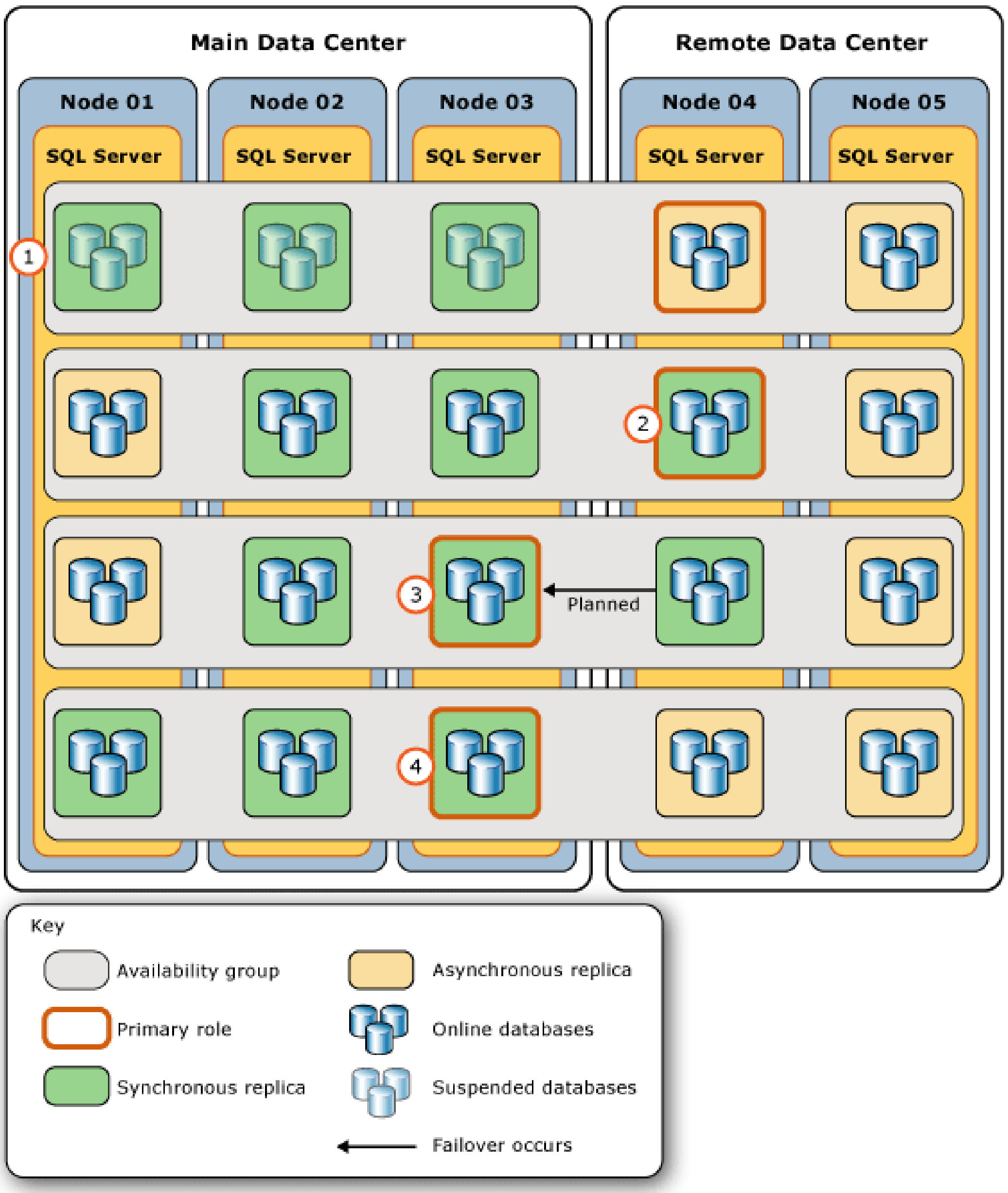
Step	Action	Links
	failover transitions this secondary replica to the primary role and suspends the secondary databases on the remaining secondary replica (on **Node 05**).	• Perform a Forced Manual Failover of an Availability Group (SQL Server)
3.	The DBA manually resumes each of the secondary databases on the remaining secondary replica.	Resume an Availability Database (SQL Server)

Returning the Availability Group to its Original Topology

The following figure illustrates the series of actions that return the availability group to its original topology after the main data center comes back online and the WSFC nodes re-establish communication with the WSFC cluster.

Important

If the WSFC cluster quorum has been forced, as the offline nodes restart they could form a new quorum if the following conditions both exist: (a) there is no network connectivity between any of the nodes in the forced-quorum set, and (b) the restarting nodes are the majority of the cluster nodes. This would result in a "split brain" condition in which the availability group would possess two independent primary replicas, one at each data center. Before forcing quorum to create a minority quorum set, see WSFC Disaster Recovery through Forced Quorum (SQL Server).



The steps in this figure indicate the following steps:

	Step	Links
1.	The nodes in the main data center come back online and re-establish communication with the WSFC cluster. Their	[Resume an Availability Database (SQL Server](#) **Tip**

	Step	Links
	availability replicas come online as secondary replicas with suspended databases, and the DBA will need to manually resume each of these databases soon.	If you are concerned about possible data loss on the post-failover primary databases, you should attempt to create a database snapshot on the suspended databases on one the synchronous-commit secondary database. Keep in mind that the transaction log truncation is delayed on a primary database while any of its secondary databases is suspended. Also the synchronization health of the synchronous-commit secondary replica cannot transition to HEALTHY as long as any local database remains suspended.
2.	Once the databases are resumed, the DBA changes the new primary replica to synchronous-commit mode temporarily. This involves two steps: 1. Change one offline availability replica to asynchronous-commit mode. 2. Change the new primary replica to synchronous-commit mode. **Note** This step enables resumed synchronous-commit secondary databases to become SYNCHRONIZED.	Change the Availability Mode of an Availability Replica (SQL Server)
3.	Once the synchronous-commit secondary replica on **Node 03** (the original primary replica)	• sys.dm_hadr_database_replica_states • Use AlwaysOn Policies to View the Health of an Availability Group (SQL

	Step	Links
	enters the HEALTHY synchronization state, the DBA performs a planned manual failover to that replica, to make it the primary replica again. The replica on Node 04 returns to being a secondary replica.	Server) • Perform a Planned Manual Failover of an Availability Group (SQL Server)
4.	The DBA connects to the new primary replica and: 1. Changes the former primary replica (in the remote center) back to asynchronous-commit mode. 2. Changes the asynchronous-commit secondary replica in the main data center back to synchronous-commit mode.	Change the Availability Mode of an Availability Replica (SQL Server)

Related Tasks

To adjust quorum votes

- View Cluster Quorum NodeWeight Settings
- Configure Cluster Quorum NodeWeight Settings
- Force a WSFC Cluster to Start Without a Quorum

Planned manual failover:

- Perform a Planned Manual Fail Over of an Availability Group (SQL Server)
- Use the Fail Over Availability Group Wizard (SQL Server Management Studio)

To troubleshoot:

- Troubleshoot AlwaysOn Availability Groups Configuration (SQL Server)
- Troubleshoot a Failed Add-File Operation (AlwaysOn Availability Groups)

Related Content

- Microsoft SQL Server AlwaysOn Solutions Guide for High Availability and Disaster Recovery
- SQL Server AlwaysOn Team Blog: The official SQL Server AlwaysOn Team Blog

See Also

AlwaysOn Availability Groups (SQL Server)

Availability Modes (AlwaysOn Availability Groups)

Failover Modes (AlwaysOn Availability Groups)

Client Connection Access to Availability Replicas (SQL Server)

Monitoring of Availability Groups (SQL Server)

Windows Server Failover Clusters (WSFC) with SQL Server

Use the Fail Over Availability Group Wizard (SQL Server Management Studio)

This topic describes how to perform a planned manual failover or forced manual failover (forced failover) on an AlwaysOn availability group by using SQL Server Management Studio, Transact-SQL, or PowerShell in SQL Server 2012. An availability group fails over at the level of an availability replica. If you fail over to a secondary replica in the SYNCHRONIZED state, the wizard performs a planned manual failover (without data loss). If you fail over to a secondary replica in the UNSYNCHRONIZED or NOT SYNCHRONIZING state, the wizard performs a forced manual failover—also known as a *forced failover* (with possible data loss). Both forms of manual failover transition the secondary replica to which you are connected to the primary role. A planned manual failover currently transitions the former primary replica to the secondary role. After a forced failover, when the former primary replica comes online, it transitions to the secondary role.

- **Before you begin:**

  Limitations and Restrictions

  Prerequisites for Using the Failover Availability Group Wizard

  Security

- **To fail over an availability group, using:**

  SQL Server Management Studio

- **Fail Over Availability Group Wizard pages:**

  Select New Primary Replica page (later in this topic)

  Connect to Replica page (later in this topic)

  Confirm Potential Data Loss page (later in this topic)

  Summary Page (AlwaysOn Availability Group wizards)

  Progress Page (AlwaysOn Availability Group wizards)

  Results Page (AlwaysOn Availability Group wizards)

Before You Begin

Before your first planned manual failover, see the "Before You Begin" section in Perform a Planned Manual Fail Over of an Availability Group.

Before your first forced failover, see the "Before You Begin" and "Follow Up: Essential Tasks After a Forced Failover" sections in Force Failover of an Availability Group (SQL Server).

Limitations and Restrictions

- A failover command returns as soon as the target secondary replica has accepted the command. However, database recovery occurs asynchronously after the availability group has finished failing over.
- Cross-database consistency across databases within the availability group is not maintained on failover.

> **Note**
> Cross-database transactions and distributed transactions are not supported by AlwaysOn Availability Groups. For more information, see Cross-Database Transactions Not Supported For Database Mirroring or AlwaysOn Availability Groups (SQL Server).

Prerequisites for Using the Failover Availability Group Wizard

- You must be connected to the server instance that hosts an availability replica that is currently available.

Security

Permissions

Requires ALTER AVAILABILITY GROUP permission on the availability group, CONTROL AVAILABILITY GROUP permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.

Using SQL Server Management Studio

To Use the Failover Availability Group Wizard

1. In Object Explorer, connect to the server instance that hosts a secondary replica of the availability group that needs to be failed over, and expand the server tree.
2. Expand the **AlwaysOn High Availability** node and the **Availability Groups** node.
3. To launch the Failover Availability Group Wizard, right-click the availability group that you are going to fail over, and select **Failover**.
4. The information presented by the **Introduction** page depends on whether any secondary replica is eligible for a planned failover. If this page says, "**Perform a planned failover for this availability group**", you can failover the availability group without data loss.
5. On the **Select New Primary Replica** page, you can view the status of the current primary replica and of the WSFC quorum, before you choose the secondary replica that will become the new primary replica (the *failover target*). For a planned manual failover, be sure to select a secondary replica whose **Failover Readiness** value is "**No data loss**". For a forced failover, for all the possible failover targets, this value will be “**Data loss, Warnings(#)**”, where # indicates the number of warnings that exist for a given secondary replica. To view the warnings for a given failover target, click its “Failover Readiness” value.

For more information, see Select New Primary Replica page, later in this topic.

6. On the **Connect to Replica** page, connect to the failover target. For more information, see Connect to Replica page, later in this topic.
7. If you are performing a forced failover, the wizard displays the **Confirm Potential Data Loss** page. To proceed with the failover, you must select **Click here to confirm failover with potential data loss**. For more information, see .Confirm Potential Data Loss page, later in this topic.
8. On the **Summary** page, review the implications of failing over to the selected secondary replica.

   If you are satisfied with your selections, optionally click **Script** to create a script of the steps the wizard will execute. Then, to failover the availability group to the selected secondary replica, click **Finish**.
9. The **Progress** page displays the progress of failing over the availability group.
10. When the failover operation finishes, the **Results** page displays the result. When the wizard completes, click **Close** to exit.

    For more information, see Results Page (AlwaysOn Availability Group Wizards).
11. After a forced failover, see the "Follow Up: After a Forced Failover" section in the Force Failover of an Availability Group (SQL Server).

Help for Pages that are Exclusive to This Wizard

This section describes the pages that are unique to the Fail Over Availability Group Wizard.

In This Section

- Select New Primary Replica page
- Connect to Replica page
- Confirm Potential Data Loss page

The other pages of this wizard share help with one or more of the other AlwaysOn Availability Groups wizards and are documented in separate F1 help topics.

Select New Primary Replica Page

This section describes the options of the **Select New Primary Replica** page. Use this page to select the secondary replica (failover target) to which the availability group will fail over. This replica will become the new primary replica.

Page Options

Current Primary Replica

Displays the name of the current primary replica, if it is online.

Primary Replica Status

Displays the status of the current primary replica, if it is online.

Quorum Status

Displays the WSFC quorum status for the availability replica, one of:

Value	Description
Normal quorum	The cluster has started with normal quorum.
Forced quorum	The cluster has started with forced quorum.
Unknown quorum	The cluster quorum status is unavailable.
Not applicable	The node that hosts the availability replica has no quorum.

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

Choose a new primary replica

Use this grid to select a secondary replica to become the new primary replica. The columns in this grid are as follows:

Server Instance

Displays the name of a server instance that hosts a secondary replica.

Availability Mode

Displays the availability mode of the server instance, one of:

Value	Description
Synchronous commit	Under synchronous-commit mode, before committing transactions, a synchronous-commit primary replica waits for a synchronous-commit secondary replica to acknowledge that it has finished hardening the log. Synchronous-commit mode ensures that once a given secondary database is synchronized with the primary database, committed transactions are fully protected.
Asynchronous commit	Under asynchronous-commit mode, the primary replica commits transactions without waiting for acknowledgement that an asynchronous-commit secondary

	replica has hardened the log. Asynchronous-commit mode minimizes transaction latency on the secondary databases but allows them to lag behind the primary databases, making some data loss possible.

For more information, see Availability Modes (AlwaysOn Availability Groups).

Failover Mode

Displays the failover mode of the server instance, one of:

Value	Description
Automatic	A secondary replica that is configured for automatic failover also supports planned manual failover whenever the secondary replica is synchronized with the primary replica.
Manual	Two types of manual failover exist: planned (without data loss) and forced (with possible data loss). For a given secondary replica, only one of these is supported, depending on the availability mode and, for synchronous-commit mode, the synchronization state of the secondary replica. To determine which form of manual failover is currently supported by a given secondary replica, see the **Failover Readiness** column of this grid.

For more information, see Failover Modes (AlwaysOn Availability Groups).

Failover Readiness

Displays failover readiness of the secondary replica, one of:

Value	Description
No data loss	This secondary replica currently supports planned failover. This value occurs only when a synchronous-commit mode

	secondary replica is currently synchronized with the primary replica.
Data loss, Warnings(#)	This secondary replica currently supports forced failover (with possible data loss). This value occurs whenever the secondary replica is not synchronized with the primary replica. Click the data-loss warnings link for information about the possible data loss.

Refresh

Click to update the grid.

Cancel

Click to cancel the wizard. On the **Select New Primary Replica** page, cancelling the wizard cause it to exit without performing any actions.

Confirm Potential Data Loss Page

This section describes the options of the **Confirm Potential Data Loss** page, which is displayed only if you are performing a forced failover. This topic is used only by the Fail Over Availability Group Wizard. Use this page to indicate whether you are willing to risk possible data loss in order to force the availability group to fail over.

Confirm Potential Data Loss Options

If the selected secondary replica is not synchronized with the primary replica, the wizard displays a warning that failing over to this secondary replica could cause data loss on one or more databases.

Click here to confirm failover with potential data loss.

If you are willing to risk data loss in order to make the databases in this availability group available to users, click this checkbox. If you are not willing to risk data loss, you can either click **Previous** to return to the **Select New Primary Replica** page, or click **Cancel** to exit the wizard without failing over the availability group.

Cancel

Click to cancel the wizard. On the **Confirm Potential Data Loss** page, cancelling the wizard cause it to exit without performing any actions.

Connect to Replica Page

This section describes the options of the **Connect to Replica** page of the Fail Over Availability Group Wizard. This page is displayed only if you are not connected to the target secondary replica. Use this page to connect to the secondary replica that you have selected as the new primary replica.

Page Options

Grid columns:

Server Instance

Displays the name of the server instance that will host the availability replica.

Connected As

Displays the account that is connected to the server instance, once the connection has been established. If this column displays "**Not Connected**" for a given server instance, you will need to click the **Connect** button.

Connect

Click if this server instance is running under a different account than other server instances to which you need to connect.

Cancel

Click to cancel the wizard. On the **Connect to Replica** page, cancelling the wizard cause it to exit without performing any actions.

See Also

Overview of AlwaysOn Availability Groups (SQL Server)

Availability Modes (AlwaysOn Availability Groups)

Failover Modes (AlwaysOn Availability Groups)

Perform a Planned Manual Fail Over of an Availability Group

Perform a Forced Manual Failover of an Availability Group (SQL Server)

WSFC Quorum Failure with SQL Server

Add a Database to an Availability Group

This topic describes how to add a database to an AlwaysOn availability group by using SQL Server Management Studio, Transact-SQL, or PowerShell in SQL Server 2012.

- **Before you begin:**

  Prerequisites and Restrictions

  Permissions

- **To add a database to an availability group, using:**

  SQL Server Management Studio

  Procedure Transact-SQL

  PowerShell

Before You Begin

Prerequisites and Restrictions

- You must be connected to the server instance that hosts the primary replica.
- The database must reside on the server instance that hosts the primary replica and comply with the prerequisites and restrictions for availability databases. For more information, see Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups (SQL Server).

Security

Permissions

Requires ALTER AVAILABILITY GROUP permission on the availability group, CONTROL AVAILABILITY GROUP permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.

Using SQL Server Management Studio

To add a database to an availability group

1. In Object Explorer, connect to the server instance that hosts the primary replica, and expand the server tree.
2. Expand the **AlwaysOn High Availability** node and the **Availability Groups** node.
3. Right-click the availability group, and select one of the following commands:
   - To launch the Add Database to Availability Group Wizard, select the **Add Database** command. For more information, see Add a Database to an Availability Group (Add Database Wizard).
   - To add one or more databases by specifying them in the **Availability Group Properties** dialog box, select the **Properties** command. The steps for adding a database are as follows:
     i. In the **Availability Databases** pane, click the **Add** button. This creates and selects a blank database field.
     ii. Enter the name of a database that meets the availability-databases prerequisites.

     To add another database, repeat the preceding steps. When you are done specifying databases, click **OK** to complete the operation.

     After you use the **Availability Group Properties** dialog box to add a database to an availability group, you need to configure the corresponding secondary database on each server instance that hosts a secondary replica. For more information, see Start Data Movement on an AlwaysOn Secondary Database (SQL Server).

Using Transact-SQL

To add a database to an availability group

1. Connect to the server instance that hosts the server instance that hosts the primary replica.
2. Use the ALTER AVAILABILITY GROUP statement, as follows:

   ALTER AVAILABILITY GROUP *group_name* ADD DATABASE database_name [,...n] where group_name is the name of the availability group and database_name is the name of a database to be added to the group.

The following example adds the *MyDb3* database to the *MyAG* availability group.

```
-- Connect to the server instance that hosts the primary replica.
-- Add an existing database to the availability group.
ALTER AVAILABILITY GROUP MyAG ADD DATABASE MyDb3;
GO
```

3. After you add a database to an availability group, you need to configure the corresponding secondary database on each server instance that hosts a secondary replica. For more information, see Start Data Movement on an AlwaysOn Secondary Database (SQL Server).

Using PowerShell

To add a database to an availability group

1. Change directory (**cd**) to the server instance that hosts the primary replica.
2. Use the **Add-SqlAvailabilityDatabase** cmdlet.

   For example, the following command adds the secondary database `MyDd` to the `MyAG` availability group, whose primary replica is hosted by `PrimaryServer\InstanceName`.

```
Add-SqlAvailabilityDatabase `
-Path
SQLSERVER:\SQL\PrimaryServer\InstanceName\AvailabilityGroups\MyAG `
-Database "MyDb"
```

Note
To view the syntax of a cmdlet, use the **Get-Help** cmdlet in the SQL Server PowerShell environment. For more information, see Get Help SQL Server PowerShell.

3. After you add a database to an availability group, you need to configure the corresponding secondary database on each server instance that hosts a secondary replica. For more information, see Start Data Movement on an AlwaysOn Secondary Database (SQL Server).

To set up and use the SQL Server PowerShell provider

- SQL Server PowerShell Provider

For a complete example, see Example (PowerShell), below.

Example (PowerShell)

The following example shows the full process for preparing a secondary database from a database on the server instance that hosts the primary replica of an availability group, adding the database to an availability group (as a primary database), and then joining the secondary database to the availability group. First, the example backs up the database and its transaction log. Then the example restores the database and log backups to the server instances that host a secondary replica.

The example calls **Add-SqlAvailabilityDatabase** twice: first on the primary replica to add the database to the availability group, and then on the secondary replica to join the secondary database on that replica to the availability group. If you have more than one secondary replica, restore and join the secondary database on each of them.

```
$DatabaseBackupFile = "\\share\backups\MyDatabase.bak"
$LogBackupFile = "\\share\backups\MyDatabase.trn"
$MyAgPrimaryPath = 
"SQLSERVER:\SQL\PrimaryServer\InstanceName\AvailabilityGroups\MyAg"
$MyAgSecondaryPath = 
"SQLSERVER:\SQL\SecondaryServer\InstanceName\AvailabilityGroups\MyAg"

Backup-SqlDatabase -Database "MyDatabase" -BackupFile $DatabaseBackupFile -
ServerInstance "PrimaryServer\InstanceName"
Backup-SqlDatabase -Database "MyDatabase" -BackupFile $LogBackupFile -
ServerInstance "PrimaryServer\InstanceName" -BackupAction 'Log'

Restore-SqlDatabase -Database "MyDatabase" -BackupFile $DatabaseBackupFile -
ServerInstance "SecondaryServer\InstanceName" -NoRecovery
Restore-SqlDatabase -Database "MyDatabase" -BackupFile $LogBackupFile -
ServerInstance "SecondaryServer\InstanceName" -RestoreAction 'Log' -
NoRecovery

Add-SqlAvailabilityDatabase -Path $MyAgPrimaryPath -Database "MyDatabase"
Add-SqlAvailabilityDatabase -Path $MyAgSecondaryPath -Database "MyDatabase"
```

See Also

AlwaysOn Availability Groups

Creation and Configuration of Availability Groups (SQL Server)

Use the Availability Group Dashboard (SQL Server Management Studio)

Monitoring Availability Groups (Transact-SQL)

Use the Add Database to Availability Group Wizard (SQL Server Management Studio)

Use the Add Database to Availability Group Wizard to help you add one or more databases to an existing AlwaysOn availability group.

Note
For information about using Transact-SQL or PowerShell to add a database, see Add a Database to an Availability Group (SQL Server).

In This Topic:

- **Before you begin:**

  Prerequisites and Restrictions

  Security

- **To add a database, using:** Add Database to Availability Group Wizard (SQL Server Management Studio)

Before You Begin

If you have never added a database to an availability group, see the "Availability Databases" section in Prerequisites and Restrictions for AlwaysOn Availability Groups (SQL Server).

Prerequisites, Restrictions, and Recommendations

- You must be connected to the server instance that hosts the current primary replica.
- If a database is encrypted or even contains a Database Encryption Key (DEK), you cannot use the New Availability Group Wizard or Add Database to Availability Group Wizard to add the database to an availability group. Even if an encrypted database has been decrypted, its log backups might contain encrypted data. In this case, full initial data synchronization could fail on the database. This is because the restore log operation might require the certificate that was used by the database encryption keys (DEKs), and that certificate might be unavailable.

  To make a decrypted database eligible to add to an availability group using the wizard:

  a. Create a log backup of the primary database.

  b. Create a full database backup of the primary database.

  c. Restore the database backup on the server instance that hosts the secondary replica.

  d. Create a new log backup from primary database.

  e. Restore this log backup on the secondary database.

- **Prerequisites for using full initial data synchronization**
  - All the database-file paths must be identical on every server instance that hosts a replica for the availability group.
  - No primary database name can exist on any server instance that hosts a secondary replica. This means that none of the new secondary databases can exist yet.

- You will need to specify a network share in order for the wizard to create and access backups. For the primary replica, the account used to start the Database Engine must have read and write file-system permissions on a network share. For secondary replicas, the account must have read permission on the network share.

If you are unable to use the wizard to perform full initial data synchronization, you need to prepare your secondary databases manually. You can do this before or after running the wizard. For more information, see Manually Prepare a Secondary Database for an Availability Group (SQL Server).

Security

Permissions

Requires ALTER AVAILABILITY GROUP permission on the availability group, CONTROL AVAILABILITY GROUP permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.

Using the Add Database to Availability Group Wizard (SQL Server Management Studio)

To Use the Add Database to Availability Group Wizard

1. In Object Explorer, connect to the server instance that hosts the primary replica of the availability group, and expand the server tree.
2. Expand the **AlwaysOn High Availability** node and the **Availability Groups** node.
3. Right-click the availability group to which you are adding a database, and select the **Add Database** command. This command launches the Add Database to Availability Group Wizard.
4. On the **Select Databases** page, select one or more databases. For more information, see Select Databases Page (New Availability Group Wizard/Add Database Wizard).
5. On the **Select Initial Data Synchronization** page, choose how you want your new secondary databases to be created and joined to the availability group. Choose one of the following options:
   - **Full**

     Select this option if your environment meets the requirements for automatically starting initial data synchronization (for more information, see Prerequisites, Restrictions, and Recommendations , earlier in this topic).

     If you select **Full**, after creating the availability group, the wizard will attempt to back up every primary database and its transaction log to a network share and restore the backups on every server instance that hosts an secondary replica. The wizard will then join every secondary database to the availability group.

     In the **Specify a shared network location accessible by all replicas:** field, specify a backup share to which all of the server instance that host replicas have read-write access.

The log backups will be part of your log backup chain. Store the log backup files appropriately.

noteDXDOC112778PADS Security Note

For information about the required file-system permissions, see Prerequisites, earlier in this topic.

- **Join only**

  If you have manually prepared secondary databases on the server instances that will host the secondary replicas, you can select this option. The wizard will join the existing secondary databases to the availability group.

- **Skip initial data synchronization**

  Select this option if you want to use your own database and log backups of your primary databases. For more information, see Manually Start Data Synchronization on an AlwaysOn Secondary Database (SQL Server).

For more information, see Select Initial Data Synchronization Page (AlwaysOn Availability Group wizards).

6. On the **Connect to Existing Secondary Replicas** page, if the instances of SQL Server that host the availability replicas for this availability group are all running as a service in the same user account, click **Connect all**. If any of the server instances are running as a service under different accounts, click the individual **Connect** button to the right of each server instance name.

   For more information, see Connect to Existing Secondary Replicas Page (Add Replica Wizard/Add Databases Wizard).

7. The **Validation** page verifies whether the values you specified in this Wizard meet the requirements of the New Availability Group Wizard. To make a change, you can click **Previous** to return to an earlier wizard page to change one or more values. The click **Next** to return to the **Validation** page, and click **Re-run Validation**.

   For more information, see Validation Page (AlwaysOn Availability Group Wizards).

8. On the **Summary** page, review your choices for the new availability group. To make a change, click **Previous** to return to the relevant page. After making the change, click **Next** to return to the **Summary** page.

   For more information, see Summary Page (AlwaysOn Availability Group Wizards).

   If you are satisfied with your selections, optionally click Script to create a script of the steps the wizard will execute. Then, to create and configure the new availability group, click **Finish**.

9. The **Progress** page displays the progress of the steps for creating the availability group (configuring endpoints, creating the availability group, and joining the secondary replica to the group).

   For more information, see Progress Page (AlwaysOn Availability Group Wizards).

10. When these steps complete, the **Results** page displays the result of each step. If all these steps succeed, the new availability group is completely configured. If any of the steps result

in an error, you might need to manually complete the configuration. For information about the cause of a given error, click the associated "Error" link in the **Result** column.

When the wizard completes, click **Close** to exit.

For more information, see Results Page (AlwaysOn Availability Group Wizards).

11. If initial data synchronization was not automatically started on all of you secondary database, you need to configure any not-yet-joined secondary databases. For more information, see Manually Start Data Synchronization on an AlwaysOn Secondary Database (SQL Server).

Related Tasks

- Prepare a Secondary Database for an Availability Group (SQL Server)
- Join a Secondary database to an Availability Group (SQL Server)

See Also

AlwaysOn Availability Groups (SQL Server)

Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups (SQL Server)

Add a Database to an Availability Group (SQL Server)

Manually Start Data Synchronization on an AlwaysOn Secondary Database (SQL Server)

Add a Database to an Availability Group (SQL Server)

Connect to Existing Secondary Replicas Page (Add Replica Wizard/Add Databases Wizard)

This help topic describes the options of the **Connect to Existing Secondary Replicas** page. This topic is used by the Add Replica to Availability Group Wizard and Add Database to Availability Group Wizard of SQL Server 2012.

Grid columns:

Server Instance

Displays the name of the server instance that will host the availability replica.

Connected As

Displays the account that is connected to the server instance, once the connection has been established. If this column displays "**Not Connected**" for a given server instance, you will need to click either the **Connect** or **Connect All** button.

Connect

Click if this server instance is running under a different account than other server instances to which you need to connect.

Connect All

Click only if every instance of SQL Server to which you need to connect is running as a service in the same user account.

Cancel

Click to cancel the wizard. On the **Connect to Existing Secondary Replica** page, cancelling the wizard cause it to exit without performing any actions.

Related Tasks

- Use the Add Replica to Availability Group Wizard
- Use the Add Database to Availability Group Wizard

See Also

AlwaysOn Availability Groups

Suspend an Availability Database

You can suspend an availability database in AlwaysOn Availability Groups by using SQL Server Management Studio, Transact-SQL, or PowerShell in SQL Server 2012. Note that a suspend command needs to be issued on the server instance that hosts the database to be suspended or resumed.

The effect of a suspend command depends on whether you suspend a secondary database or a primary database, as follows:

Suspended Database	**Effect of Suspend Command**
Secondary database	Only the local secondary database is suspended and its synchronization state becomes NOT SYNCHRONIZING. Other secondary databases are not affected. The suspended database stops receiving and applying data (log records) and begins to fall behind the primary database. Existing connections on the readable secondary remain usable. New connections to the suspended database on the readable secondary are not allowed until data movement is resumed. The primary database remains available. If you suspend each of the corresponding secondary databases, the primary database runs exposed. **Important** While a secondary database is

Suspended Database	Effect of Suspend Command
	suspended, the send queue of the corresponding primary database will accumulate unsent transaction log records. Connections to the secondary replica return data that was available at the time the data movement was suspended.
Primary database	The primary database stops data movement to every connected secondary database. The primary database continues running, in an exposed mode. The primary database remains available to clients, and existing connections on a readable secondary remain usable and new connections can be made.

Note
Suspending an AlwaysOn secondary database does not directly affect the availability of the primary database. However, suspending a secondary database can impact redundancy and failover capabilities for the primary database. This is in contrast to database mirroring, where the mirroring state is suspended on both the mirror database and the principal database. Suspending an AlwaysOn primary database suspends data movement on all the corresponding secondary databases, and redundancy and failover capabilities cease for that database until the primary database is resumed.

- **Before you begin:**

  Limitations and Restrictions

  Prerequisites

  Recommendations

  Security
- **To suspend a database, using:**
- SQL Server Management Studio

  Transact-SQL

  PowerShell
- **Follow up:** Avoiding a Full Transaction Log
- Related Tasks

Before You Begin

Limitations and Restrictions

A SUSPEND command returns as soon as it has been accepted by the replica that hosts the target database, but actually suspending the database occurs asynchronously.

Prerequisites

You must be connected to the server instance that hosts the database that you want to suspend. To suspend a primary database and the corresponding secondary databases, connect to the server instance that hosts the primary replica. To suspend a secondary database while leaving the primary database available, connect to the secondary replica.

Recommendations

During bottlenecks, suspending one or more secondary databases briefly might be useful to improve performance temporarily on the primary replica. As long as a secondary database remains suspended, the transaction log of the corresponding primary database cannot be truncated. This causes log records to accumulate on the primary database. Therefore, we recommend that you resume, or remove, a suspended secondary database quickly. For more information, see Follow up: Avoiding a Full Transaction Log, later in this topic.

Security

Permissions

Requires ALTER permission on the database.

Requires ALTER AVAILABILITY GROUP permission on the availability group, CONTROL AVAILABILITY GROUP permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.

Using SQL Server Management Studio

To suspend a database

1. In Object Explorer, connect to the server instance that hosts the availability replica on which you want to suspend a database, and expand the server tree. For more information, see Prerequisites, earlier in this topic.
2. Expand the **AlwaysOn High Availability** node and the **Availability Groups** node.
3. Expand the availability group.
4. Expand the **Availability Databases** node, right-click the database, and click **Suspend Data Movement**.
5. In the **Suspend Data Movement** dialog box, click **OK**.

   Object Explorer indicates that the database is suspended by changing the database icon to display a pause indicator.

Note

To suspend additional databases on this replica location, repeat steps 4 and 5 for each database.

Using Transact-SQL

To suspend a database

1. Connect to the server instance that hosts the replica whose database you want to suspend. For more information, see Prerequisites, earlier in this topic.
2. Suspend the database by using the following ALTER DATABASE statement:

   ALTER DATABASE database_name SET HADR SUSPEND

Using PowerShell

To suspend a database

1. Change directory (**cd**) to the server instance that hosts the replica whose database you want to suspend. For more information, see Prerequisites, earlier in this topic.
2. Use the **Suspend-SqlAvailabilityDatabase** cmdlet to suspend the availability group.

   For example, the following command suspends data synchronization for the availability database `MyDb3` in the availability group `MyAg` on the server instance named `Computer\Instance`.

   ```
   Suspend-SqlAvailabilityDatabase `
   -Path
   SQLSERVER:\Sql\Computer\Instance\AvailabilityGroups\MyAg\Databases\MyD
   b3
   ```

   Note
   To view the syntax of a cmdlet, use the **Get-Help** cmdlet in the SQL Server PowerShell environment. For more information, see Get Help SQL Server PowerShell.

To set up and use the SQL Server PowerShell provider

- SQL Server PowerShell Provider

Follow Up: Avoiding a Full Transaction Log

Normally, when an automatic checkpoint is performed on a database, its transaction log is truncated to that checkpoint after the next log backup. However, while a secondary database is suspended, all of the current log records remain active on the primary database. If the transaction log fills up (either because it reaches its maximum size or the server instance runs out of space), the database cannot perform any more updates.

To avoid this problem, you should do one of the following:

- Add more log space for the primary database.
- Resume the secondary database before the log fills up. For more information, see Resume an Availability Database (SQL Server).
- Remove the secondary database. For more information, see Remove a Database from a Secondary Replica (SQL Server).

To troubleshoot a full transaction log

- Troubleshoot a Full Transaction Log (SQL Server Error 9002)

Related Tasks

- Resume an Availability Database (SQL Server)

See Also

AlwaysOn Availability Groups (SQL Server)

Resume an Availability Database (SQL Server)

Resume an Availability Database

You can resume a suspended availability database in AlwaysOn Availability Groups by using SQL Server Management Studio, Transact-SQL, or PowerShell in SQL Server 2012. Resuming a suspended database puts the database into the SYNCHRONIZING state. Resuming the primary database also resumes any of its secondary databases that were suspended as the result of suspending the primary database. If any secondary database was suspended locally, from the server instance that hosts the secondary replica, that secondary database must be resumed locally. Once a given secondary database and the corresponding primary database are in the SYNCHRONIZING state, data synchronization resumes on the secondary database.

Note

Suspending and resuming an AlwaysOn secondary database does not directly affect the availability of the primary database. However, suspending a secondary database can impact redundancy and failover capabilities for the primary database, until the suspended secondary database is resumed. This is in contrast to database mirroring, where the mirroring state is suspended on both the mirror database and the principal database until mirroring is resumed. Suspending an AlwaysOn primary database suspends data movement on all the corresponding secondary databases, and redundancy and failover capabilities cease for that database until the primary database is resumed.

- **Before you begin:**

  Limitations and Restrictions

  Prerequisites

  Security

- **To resume a secondary database, using:**

  SQL Server Management Studio

  Transact-SQL

  PowerShell

- Related Tasks

Before You Begin

Limitations and Restrictions

A RESUME command returns as soon as it has been accepted by the replica that hosts the target database, but actually resuming the database occurs asynchronously.

Prerequisites

- You must be connected to the server instance that hosts the database to be resumed.
- The availability group must be online.
- The primary database must be online and available.

Security

Permissions

Requires ALTER permission on the database.

Requires ALTER AVAILABILITY GROUP permission on the availability group, CONTROL AVAILABILITY GROUP permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.

Using SQL Server Management Studio

To resume a secondary database

1. In Object Explorer, connect to the server instance that hosts the availability replica on which you want to resume a database, and expand the server tree.
2. Expand the **AlwaysOn High Availability** node and the **Availability Groups** node.
3. Expand the availability group.
4. Expand the **Availability Databases** node, right-click the database, and click **Resume Data Movement**.
5. In the **Resume Data Movement** dialog box, click **OK**.

Note
To resume additional databases on this replica location, repeat steps 4 and 5 for each database.

Using Transact-SQL

To resume a secondary database that was suspended locally

1. Connect to the server instance that hosts the secondary replica whose database you want to resume.
2. Resume the secondary database by using the following ALTER DATABASE statement:

   ALTER DATABASE database_name SET HADR RESUME

Using PowerShell

To resume a secondary database

1. Change directory (**cd**) to the server instance that hosts the replica whose database you want to resume. For more information, see Prerequisites, earlier in this topic.

2. Use the **Resume-SqlAvailabilityDatabase** cmdlet to resume the availability group.

   For example, the following command resumes data synchronization for the availability database `MyDb3` in the availability group `MyAg`.

   ```
   Resume-SqlAvailabilityDatabase `
   -Path
   SQLSERVER:\Sql\Computer\Instance\AvailabilityGroups\MyAg\Databases\MyD
   b3
   ```

   Note
   To view the syntax of a cmdlet, use the **Get-Help** cmdlet in the SQL Server PowerShell environment. For more information, see Get Help SQL Server PowerShell.

To set up and use the SQL Server PowerShell provider

- SQL Server PowerShell Provider

Related Tasks

- Suspend an Availability Database (SQL Server)

See Also

Overview AlwaysOn Availability Groups (SQL Server)

Remove a Secondary Database from an Availability Group

This topic describes how to remove a secondary database from an AlwaysOn availability group by using SQL Server Management Studio, Transact-SQL, or PowerShell in SQL Server 2012.

- **Before you begin:**

  Prerequisites

  Security

- **To remove a secondary database, using:**

  SQL Server Management Studio

  Transact-SQL

  PowerShell

- **Follow Up:** After Removing a Secondary Database from an Availability Group

Before You Begin

Prerequisites and Restrictions

- This task is supported only on secondary replicas. You must be connected to the server instance that hosts the secondary replica from which the database is to be removed.

Security

Permissions

Requires ALTER permission on the database.

🡅

Using SQL Server Management Studio

To remove a secondary database from an availability group

1. In Object Explorer, connect to the server instance that hosts the secondary replica from which you want to remove one or more secondary databases, and expand the server tree.
2. Expand the **AlwaysOn High Availability** node and the **Availability Groups** node.
3. Select the availability group, and expand the **Availability Databases** node.
4. This step depends on whether you want to remove multiple databases groups or only one database, as follows:
   - To remove multiple databases, use the **Object Explorer Details** pane to view and select all the databases that you want to remove. For more information, see Use Object Explorer Details to Monitor Availability Groups (SQL Server Management Studio).
   - To remove a single database, select it in either the **Object Explorer** pane or the **Object Explorer Details** pane.
5. Right-click the selected database or databases, and select **Remove Secondary Database** in the command menu.
6. In the **Remove Database from Availability Group** dialog box, to remove all the listed databases, click **OK**. If you do not want to remove all the listed databases, click **Cancel**.

🡅

Using Transact-SQL

To remove a secondary database from an availability group

1. Connect to the server instance that hosts the secondary replica.
2. Use the SET HADR clause of the ALTER DATABASE statement, as follows:

   ALTER DATABASE database_name SET HADR OFF

   where database_name is the name of a secondary database to be removed from the availability group to which it belongs.

   The following example removes the local secondary database *MyDb2* from its availability group.

   ```
   ALTER DATABASE MyDb2 SET HADR OFF;
   GO
   ```

🡅

Using PowerShell

To remove a secondary database from an availability group

1. Change directory (**cd**) to the server instance that hosts the secondary replica.
2. Use the **Remove-SqlAvailabilityDatabase** cmdlet, specifying the name of the availability database to be removed from the availability group. When you are connected to a server

instance that hosts a secondary replica, only the local secondary database is removed from the availability group.

For example, the following command removes the secondary database `MyDb8` from the secondary replica hosted by the server instance named `SecondaryComputer\Instance`. Data synchronization to the removed secondary databases ceases. This command does not affect the primary database or any other secondary databases.

```
Remove-SqlAvailabilityDatabase `
-Path
SQLSERVER:\Sql\SecondaryComputer\InstanceName\AvailabilityGroups\MyAg\
Databases\MyDb8
```

Note

To view the syntax of a cmdlet, use the **Get-Help** cmdlet in the SQL Server PowerShell environment. For more information, see Get Help SQL Server PowerShell.

To set up and use the SQL Server PowerShell provider

- SQL Server PowerShell Provider

Follow Up: After Removing a Secondary Database from an Availability Group

When a secondary database is removed, it is no longer joined to the availability group and all information about the removed secondary database is discarded by the availability group. The removed secondary database is placed in the RESTORING state.

Tip

For a short time after removing a secondary database, you might be able to restart AlwaysOn data synchronization on the database by re-joining it to the availability group. For more information, see Join a Secondary Database to an Availability Group (SQL Server).

At this point there are alternative ways of dealing with a removed secondary database:

- If you no longer need the secondary database, you can drop it.

  For more information, see DROP DATABASE (Transact-SQL) or How to: Delete a database (SQL Server Management Studio).

- If you want to access a removed secondary database after it has been removed from the availability group, you can recover the database. However, if you recover a removed secondary database, two divergent, independent databases that have the same name are online. You must make sure that clients can access only the current primary database.

  For more information, see Recover a Database Without Restoring Data (Transact-SQL).

See Also

Overview of AlwaysOn Availability Groups

Remove a Primary Database from an Availability Group

Remove a Primary Database from an Availability Group

This topic describes how to remove both the primary database and the corresponding secondary database(s) from an AlwaysOn availability group by using SQL Server Management Studio, Transact-SQL, or PowerShell in SQL Server 2012.

- **Before you begin:**

  Prerequisites and Restrictions

  Security

- **To remove an availability database, using:**

  SQL Server Management Studio

  Transact-SQL

  PowerShell

- **Follow Up:** After Removing an Availability Database from an Availability Group

Before You Begin

Prerequisites and Restrictions

- This task is supported only on primary replicas. You must be connected to the server instance that hosts the primary replica.

Security

Permissions

Requires ALTER AVAILABILITY GROUP permission on the availability group, CONTROL AVAILABILITY GROUP permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.

Using SQL Server Management Studio

To remove an availability database

1. In Object Explorer, connect to the server instance that hosts the primary replica of the database or databases to be removed, and expand the server tree.
2. Expand the **AlwaysOn High Availability** node and the **Availability Groups** node.
3. Select the availability group, and expand the **Availability Databases** node.
4. This step depends on whether you want to remove multiple databases groups or only one database, as follows:
   - To remove multiple databases, use the **Object Explorer Details** pane to view and select all the databases that you want to remove. For more information, see Use Object Explorer Details to Monitor Availability Groups (SQL Server Management Studio).

- To remove a single database, select it in either the **Object Explorer** pane or the **Object Explorer Details** pane.

5. Right-click the selected database or databases, and select **Remove Database from Availability Group** in the command menu.
6. In the **Remove Databases from Availability Group** dialog box, to remove all the listed databases, click **OK**. If you do not want to remove all them, click **Cancel**.

Using Transact-SQL

To remove an availability database

1. Connect to the server instance that hosts the primary replica.
2. Use the ALTER AVAILABILITY GROUP statement, as follows:

   ALTER AVAILABILITY GROUP *group_name* REMOVE DATABASE availability_database_name

   where group_name is the name of the availability group and database_name is the name of the database to be removed.

   The following example removes a databases named `Db6` from the `MyAG` availability group.

   ```
   ALTER AVAILABILITY GROUP MyAG REMOVE DATABASE Db6;
   ```

Using PowerShell

To remove an availability database

1. Change directory (**cd**) to the server instance that hosts the primary replica.
2. Use the **Remove-SqlAvailabilityDatabase** cmdlet, specifying the name of the availability database to be removed from the availability group. When you are connected to the server instance that hosts the primary replica, the primary database and its corresponding secondary databases are all removed from the availability group.

   For example, the following command removes the availability database `MyDb9` from the availability group named `MyAg`. Because the command is executed on the server instance that hosts the primary replica, the primary database and all its corresponding secondary databases are removed from the availability group. Data synchronization will no longer occur for this database on any secondary replica.

   ```
   Remove-SqlAvailabilityDatabase `
   -Path
   SQLSERVER:\Sql\PrimaryComputer\InstanceName\AvailabilityGroups\MyAg\Da
   tabases\MyDb9
   ```

   Note
   To view the syntax of a cmdlet, use the **Get-Help** cmdlet in the SQL Server 2012 PowerShell environment. For more information, see Get Help SQL Server PowerShell.

To set up and use the SQL Server PowerShell provider

- SQL Server PowerShell Provider

Follow Up: After Removing an Availability Database from an Availability Group

Removing an availability database from its availability group ends data synchronization between the former primary database and the corresponding secondary databases. The former primary database remains online. Every corresponding secondary database is placed in the RESTORING state.

At this point there are alternative ways of dealing with a removed secondary database:

- If you no longer need a given secondary database, you can drop it.

  For more information, see Delete a database (SQL Server Management Studio).
- If you want to access a removed secondary database after it has been removed from the availability group, you can recover the database. However, if you recover a removed secondary database, two divergent, independent databases that have the same name are online. You must make sure that clients can access only one of them, typically the most recent primary database.

  For more information, see Recover a Database Without Restoring Data (Transact-SQL).

See Also

AlwaysOn Availability Groups

Remove a Secondary Database from an Availability Group

Add a Secondary Replica to an Availability Group

This topic describes how to add a replica to an existing AlwaysOn availability group by using SQL Server Management Studio, Transact-SQL, or PowerShell. The new replica will be a secondary replica.

- **Before you begin:**

  Prerequisites and Restrictions

  Security
- **To add a replica, using:** SQL Server Management Studio

  Procedure Transact-SQL

  PowerShell
- **Follow Up:** After Adding a Secondary Replica

Before You Begin

We strongly recommend that you read this section before attempting to create your first availability group.

Prerequisites and Restrictions

- You must be connected to the server instance that hosts the primary replica.

For more information, see "Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups (SQL Server).

Security

Permissions

Requires ALTER AVAILABILITY GROUP permission on the availability group, CONTROL AVAILABILITY GROUP permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.

Using SQL Server Management Studio

To add a replica

1. In Object Explorer, connect to the server instance that hosts the primary replica, and expand the server tree.
2. Expand the **AlwaysOn High Availability** node and the **Availability Groups** node.
3. Right-click the availability group, and select one of the following commands:
   - Select the **Add Replica** command to launch the Add Replica to Availability Group Wizard. For more information, see Use the Add Replica to Availability Group Wizard (SQL Server Management Studio).
   - Alternatively, select the **Properties** command to open the **Availability Group Properties** dialog box. The steps for adding a replica in this dialog box are as follows:
     i. In the **Availability Replicas** pane of the dialog box, click the **Add** button. This creates and selects a replica entry in which the blank Server Instance field is selected.
     ii. Enter the name of a server instance that meets the prerequisites for hosting an availability replica.

     To add an additional replicas, repeat the preceding steps. When you are done specifying replicas, click **OK** to complete the operation.

Using Transact-SQL

To add a replica

1. Connect to the instance of SQL Server that hosts the primary replica.
2. Add the new secondary replica to the availability group by using the ADD REPLICA ON clause of the ALTER AVAILABILITY GROUP statement. The ENDPOINT_URL, AVAILABILITY_MODE, and FAILOVER_MODE options are required in an ADD REPLICA ON clause. The other replica options— BACKUP_PRIORITY, SECONDARY_ROLE, PRIMARY_ROLE, and SESSION_TIMEOUT—are optional. For more information, see ALTER AVAILABILITY GROUP (Transact-SQL).

   For example, the following Transact-SQL statement creates a new replica to an availability group named `MyAG` on the default server instance hosted by `COMPUTER04`, whose endpoint URL is `TCP://COMPUTER04.Adventure-Works.com:5022'`. This replica supports manual failover and asynchronous-commit availability mode.

```
ALTER AVAILABILITY GROUP MyAG ADD REPLICA ON 'COMPUTER04'
```

```
        WITH (
              ENDPOINT_URL = 'TCP://COMPUTER04.Adventure-Works.com:5022',
              AVAILABILITY_MODE = ASYNCHRONOUS_COMMIT,
              FAILOVER_MODE = MANUAL
              );
```

Using PowerShell

To add a replica

1. Change directory (**cd**) to the server instance that hosts the primary replica.
2. Use the **New-SqlAvailabilityReplica** cmdlet.

   For example, the following command adds an availability replica to an existing availability group named `MyAg`. This replica supports manual failover and asynchronous-commit availability mode. In the secondary role, this replica will support read access connections, allowing you to offload read-only processing to this replica.

   ```
   $agPath =
   "SQLSERVER:\Sql\PrimaryServer\InstanceName\AvailabilityGroups\MyAg"
   $endpointURL = "TCP://PrimaryServerName.domain.com:5022"
   $failoverMode = "Manual"
   $availabilityMode = "AsynchronousCommit"
   $secondaryReadMode = "AllowAllConnections"

   New-SqlAvailabilityReplica -Name SecondaryServer\Instance `
   -EndpointUrl $endpointURL `
   -FailoverMode $failoverMode `
   -AvailabilityMode $availabilityMode `
   -ConnectionModeInSecondaryRole $secondaryReadMode `
   -Path $agPath
   ```

Note
To view the syntax of a cmdlet, use the **Get-Help** cmdlet in the SQL Server PowerShell environment. For more information, see Get Help SQL Server PowerShell.

To set up and use the SQL Server PowerShell provider

- SQL Server PowerShell Provider

Follow Up: After Adding a Secondary Replica

To add a replica for an existing availability group, you must perform the following steps:

1. Connect to the server instance that is going to host the new secondary replica.
2. Join the new secondary replica to the availability group. For more information, see Join a Secondary Replica to an Availability Group (SQL Server).
3. For each database in the availability group, create a secondary database on the server instance that is hosting the secondary replica. For more information, see Prepare a Database for a Secondary Replica (Transact-SQL).
4. Join each of the new secondary databases to the availability group. For more information, see Join a Secondary Database to an Availability Group (SQL Server).

Related Tasks

To manage an availability replica

- Join a Secondary Replica to an Availability Group (SQL Server)
- Remove a Secondary Replica from an Availability Group (SQL Server)
- Configure Read-Only Access on a Secondary Availability Replica (SQL Server)
- Set the Availability Mode of an Availability Replica (SQL Server)
- Set the Failover Mode of an Availability Replica (SQL Server)
- Setting the Session-Timeout Period for an Availability Replica (SQL Server)
- Set the Session-Timeout Period for an Availability Replica (SQL Server)

See Also

ALTER AVAILABILITY GROUP (Transact-SQL)

AlwaysOn Availability Groups

Creation and Configuration of Availability Groups (SQL Server)

Use the Availability Group Dashboard (SQL Server Management Studio)

Monitoring Availability Groups (Transact-SQL)

Use the Add Replica to Availability Group Wizard (SQL Server Management Studio)

Use the Add Replica to Availability Group Wizard to help you a add new secondary replica to an existing AlwaysOn availability group.

Note
For information about using Transact-SQL or PowerShell to add a secondary replica to an availability group, see Add a Replica to an Availability Group (SQL Server).

- **Before you begin:**

  Prerequisites

Security

- **To add a replica, using:** Add Replica to Availability Group Wizard (SQL Server Management Studio)

Before You Begin

If you have never added any availability replica to an availability group, see the "Server instances" and "Availability groups and replicas" sections in Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups (SQL Server).

Prerequisites

- You must be connected to the server instance that hosts the current primary replica.
- Before adding a secondary replica, verify that the host instance of SQL Server is in the same Windows Server Failover Clustering (WSFC) cluster as the existing replicas but resides on a different cluster node. Also, verify that this server instance meets all other AlwaysOn Availability Groups prerequisites. For more information, see Prerequisites, Restrictions, and Recommendations (AlwaysOn Availability Groups).
- If a server instance that you select to host an availability replica does not yet have a database mirroring endpoint, the wizard can create the endpoint if the server instance is running under a domain service account. However, if the SQL Server service is running as a built-in account, such as Local System, Local Service, or Network Service, or a nondomain account, you must use certificates for endpoint authentication, and the wizard will be unable to create a database mirroring endpoint on the server instance. In this case, we recommend that you create the database mirroring endpoints before you launch the Add Replica to Availability Group Wizard. For more information, see Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups (SQL Server).
- **Prerequisites for using full initial data synchronization**
    - All the database-file paths must be identical on every server instance that hosts a replica for the availability group.
    - No primary database name can exist on any server instance that hosts a secondary replica. This means that none of the new secondary databases can exist yet.
    - You will need to specify a network share in order for the wizard to create and access backups. For the primary replica, the account used to start the Database Engine must have read and write file-system permissions on a network share. For secondary replicas, the account must have read permission on the network share.

  If you are unable to use the wizard to perform full initial data synchronization, you need to prepare your secondary databases manually. You can do this before or after running the wizard. For more information, see Manually Prepare a Secondary Database for an Availability Group (SQL Server).

Security

Permissions

Requires ALTER AVAILABILITY GROUP permission on the availability group, CONTROL AVAILABILITY GROUP permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.

Using the Add Replica to Availability Group Wizard (SQL Server Management Studio)

To Use the Add Replica to Availability Group Wizard

1. In Object Explorer, connect to the server instance that hosts the primary replica of the availability group, and expand the server tree.
2. Expand the **AlwaysOn High Availability** node and the **Availability Groups** node.
3. Right-click the availability group to which you are adding a secondary replica, and select the **Add Replica** command. This launches the Add Replica to Availability Group Wizard.
4. On the **Connect to Existing Secondary Replicas** page, connect to every secondary replica in the availability group. For more information, see Connect to Existing Secondary Replicas Page (Add Replica Wizard/Add Databases Wizard).
5. On the **Specify Replicas** page, specify and configure one or more new secondary replicas for the availability group. This page contains three tabs. The following table introduces these tabs. For more information, see Specify Replicas page (New Availability Group Wizard/Add Replica Wizard).

Tab	Brief Description
Replicas	Use this tab to specify each instance of SQL Server that will host a new secondary replica.
Endpoints	Use this tab to verify the existing database mirroring endpoint, if any, for each new secondary replica. If this endpoint is lacking on a server instance whose service accounts use Windows Authentication, the wizard will attempt to create the endpoint automatically. **Note** If any server instance is running under a non-domain service account, you need to do make a manual change to your server instance before you can proceed in the wizard.
Backup Preferences	Use this tab to specify your backup preference for the availability group as a whole, if you wish to modify the current setting, and to specify your backup priorities for the individual availability replicas.

6. On the **Select Initial Data Synchronization** page, choose how you want your new secondary databases to be created and joined to the availability group. Choose one of the following options:
   - **Full**

     Select this option if your environment meets the requirements for automatically starting initial data synchronization (for more information, see Prerequisites, Restrictions, and Recommendations , earlier in this topic).

     If you select **Full**, after creating the availability group, the wizard will back up every primary database and its transaction log to a network share and restore the backups on every server instance that hosts a new secondary replica. The wizard will then join every new secondary database to the availability group.

     In the **Specify a shared network location accessible by all replicas:** field, specify a backup share to which all of the server instance that host replicas have read-write access.

The log backups will be part of your log backup chain. Store the log backup files appropriately.

noteDXDOC112778PADS Security Note
For information about the required file-system permissions, see Prerequisites, earlier in this topic.

- **Join only**

  If you have manually prepared secondary databases on the server instances that will host the new secondary replicas, you can select this option. The wizard will join these new secondary databases to the availability group.

- **Skip initial data synchronization**

  Select this option if you want to use your own database and log backups of your primary databases. For more information, see Manually Start Data Synchronization on an AlwaysOn Secondary Database (SQL Server).

7. The **Validation** page verifies whether the values you specified in this Wizard meet the requirements of the Add Replica to Availability Group Wizard. To make a change, click **Previous** to return to an earlier wizard page to change one or more values. The click **Next** to return to the **Validation** page, and click **Re-run Validation**.
8. On the **Summary** page, review your choices for the new availability group. To make a change, click **Previous** to return to the relevant page. After making the change, click **Next** to return to the **Summary** page.

   If you are satisfied with your selections, optionally click Script to create a script of the steps the wizard will execute. Then, to create and configure the new availability group, click **Finish**.
9. The **Progress** page displays the progress of the steps for creating the availability group (configuring endpoints, creating the availability group, and joining the secondary replica to the group).
10. When these steps complete, the **Results** page displays the result of each step. If all these steps succeed, the new availability group is completely configured. If any of the steps result in an error, you might need to manually complete the configuration. For information about the cause of a given error, click the associated "Error" link in the **Result** column.

    When the wizard completes, click **Close** to exit.

Important
After adding a replica, see the "Follow Up: After Adding a Replica" section in Add a Replica to an Availability Group (SQL Server).

Related Tasks

- Add a Replica to an Availability Group (SQL Server)

See Also

[AlwaysOn Availability Groups (SQL Server)](#)

[Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups (SQL Server)](#)

[Add a Replica to an Availability Group (SQL Server)](#)

Change the Session-Timeout Period for an Availability Replica

This topic describes how to configure the session-timeout period of an AlwaysOn availability replica by using SQL Server Management Studio, Transact-SQL, or PowerShell in SQL Server 2012. The session-timeout period is a replica property that controls how many seconds (in seconds) that an availability replica waits for a ping response from a connected replica before considering the connection to have failed. By default, a replica waits 10 seconds for a ping response. This replica property applies only the connection between a given secondary replica and the primary replica of the availability group. For more information about the session-timeout period, see [Overview of AlwaysOn Availability Groups](#).

- **Before you begin:**

  Prerequisites

  Recommendations

  Security

- **To change the session-timeout period, using:**

  SQL Server Management Studio

  Transact-SQL

  PowerShell

Before You Begin

Prerequisites

- You must be connected to the server instance that hosts the primary replica.

Recommendations

We recommend that you keep the time-out period at 10 seconds or greater. Setting the value to less than 10 seconds creates the possibility of a heavily loaded system missing PINGs and declaring a false failure.

Security

Permissions

Requires ALTER AVAILABILITY GROUP permission on the availability group, CONTROL AVAILABILITY GROUP permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.

Using SQL Server Management Studio

To change the session-timeout period for an availability replica

1. In Object Explorer, connect to the server instance that hosts the primary replica, and expand the server tree.
2. Expand the **AlwaysOn High Availability** node and the **Availability Groups** node.
3. Click the availability group whose availability replica you want to configure.
4. Right-click the replica to be configured, and click **Properties**.
5. In the **Availability Replica Properties** dialog box, use the **Session timeout (seconds)** field to change the number of seconds for the session-timeout period on this replica.

Using Transact-SQL

To change the session-timeout period for an availability replica

1. Connect to the server instance that hosts the primary replica.
2. Use the ALTER AVAILABILITY GROUP statement, as follows:

   ALTER AVAILABILITY GROUP *group_name*

   MODIFY REPLICA ON 'instance_name' WITH (SESSION_TIMEOUT = seconds)

   where group_name is the name of the availability group, instance_name is the name of the server instance that hosts the availability replica to be modified, and seconds specifies the minimum number of seconds that the replica must wait before applying log to databases when acting as a secondary replica. The default is 0 seconds, which indicates that there is no apply delay.

   The following example, entered on the primary replica of the `AccountsAG` availability group, changes the session-timeout value to `15` seconds for the replica located on the `INSTANCE09` server instance.

   ```
   ALTER AVAILABILITY GROUP AccountsAG
      MODIFY REPLICA ON 'INSTANCE09' WITH (SESSION_TIMEOUT = 15);
   ```

Using PowerShell

To change the session-timeout period for an availability replica

1. Change directory (**cd**) to the server instance that hosts the primary replica.
2. Use the **Set-SqlAvailabilityReplica** cmdlet with the **SessionTimeout** parameter to change the number of seconds for the session-timeout period on a specified availability replica.

   For example, the following command sets the session-timeout period to 15 seconds.

   ```
   Set-SqlAvailabilityReplica –SessionTimeout 15 `
   -Path
   SQLSERVER:\Sql\PrimaryServer\InstanceName\AvailabilityGroups\MyAg\Repl
   icas\MyReplica
   ```

   Note

To view the syntax of a cmdlet, use the **Get-Help** cmdlet in the SQL Server PowerShell environment. For more information, see Get Help SQL Server PowerShell.

To set up and use the SQL Server PowerShell provider

- SQL Server PowerShell Provider

See Also

Overview of AlwaysOn Availability Groups

Remove a Secondary Replica from an Availability Group

This topic describes how to remove a secondary replica from an AlwaysOn availability group by using SQL Server Management Studio, Transact-SQL, or PowerShell in SQL Server 2012.

- **Before you begin:**

  Limitations and Restrictions

  Prerequisites

  Security

- **To remove a secondary replica, using:**

  SQL Server Management Studio

  Transact-SQL

  PowerShell

- **Follow Up:** After Removing a Secondary Replica

Before You Begin

Limitations and Restrictions

- This task is supported only on the primary replica.
- Only a secondary replica can be removed from an availability group.

Prerequisites

- You must be connected to the server instance that hosts the primary replica of the availability group.

Security

Permissions

Requires ALTER AVAILABILITY GROUP permission on the availability group, CONTROL AVAILABILITY GROUP permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.

Using SQL Server Management Studio

To remove a secondary replica

1. In Object Explorer, connect to the server instance that hosts the primary replica, and expand the server tree.
2. Expand the **AlwaysOn High Availability** node and the **Availability Groups** node.
3. Select the availability group, and expand the **Availability Replicas** node.
4. This step depends on whether you want to remove multiple replicas or only one replica, as follows:
   - To remove multiple replicas, use the **Object Explorer Details** pane to view and select all the replicas that you want to remove. For more information, see Use Object Explorer Details to Monitor Availability Groups (SQL Server Management Studio).
   - To remove a single replica, select it in either the **Object Explorer** pane or the **Object Explorer Details** pane.
5. Right-click the selected secondary replica or replicas, and select **Remove from Availability Group** in the command menu.
6. In the **Remove Secondary Replicas from Availability Group** dialog box, to remove all the listed secondary replicas, click **OK**. If you do not want to remove all the listed replicas, click **Cancel**.

Using Transact-SQL

To remove a secondary replica

1. Connect to the server instance that hosts the primary replica.
2. Use the ALTER AVAILABILITY GROUP statement, as follows:

   ALTER AVAILABILITY GROUP *group_name* REMOVE REPLICA ON 'instance_name' [,...n]

   where group_name is the name of the availability group and instance_name is the server instance where the secondary replica is located.

   The following example removes a secondary replica from the *MyAG* availability group. The target secondary replica is located on a server instance named *HADR_INSTANCE* on a computer named *COMPUTER02*.

   ```
   ALTER AVAILABILITY GROUP MyAG REMOVE REPLICA ON
   'COMPUTER02\HADR_INSTANCE';
   ```

Using PowerShell

To remove a secondary replica

1. Change directory (**cd**) to the server instance that hosts the primary replica.
2. Use the **Remove-SqlAvailabilityReplica** cmdlet.

   For example, the following command removes the availability replica on the server `MyReplica` from the availability group named `MyAg`. This command must be run on the server instance that hosts the primary replica of the availability group.

   ```
   Remove-SqlAvailabilityReplica `
   ```

```
-Path
SQLSERVER:\SQL\PrimaryServer\InstanceName\AvailabilityGroups\MyAg\Repl
icas\MyReplica
```

Note
To view the syntax of a cmdlet, use the **Get-Help** cmdlet in the SQL Server PowerShell environment. For more information, see Get Help SQL Server PowerShell.

To set up and use the SQL Server PowerShell provider

- SQL Server PowerShell Provider

Follow Up: After Removing a Secondary Replica

If you specify a replica that is currently unavailable, when the replica comes online, it will discover that it has been removed.

Removing a replica causes it to stop receiving data. After a secondary replica confirms that it has been removed from the global store, the replica removes the availability group settings from its databases, which remain on the local server instance in the RECOVERING state.

See Also

Overview of AlwaysOn Availability Groups

Add a Secondary Replica to an Availability Group (SQL Server)

Delete an Availability Group (SQL Server)

Remove an Availability Group Listener

This topic describes how to remove an availability group listener from an AlwaysOn availability group by using SQL Server Management Studio, Transact-SQL, or PowerShell in SQL Server 2012.

- **Before you begin:**

  Prerequisites

  Recommendations

  Security

- **To remove a listener, using:**

  SQL Server Management Studio

  Transact-SQL

  PowerShell

Before You Begin

Prerequisites

- You must be connected to the server instance that hosts the primary replica.

Recommendations

Before you delete an availability group listener, we recommend that you ensure that no applications are using it.

Security

Permissions

Requires ALTER AVAILABILITY GROUP permission on the availability group, CONTROL AVAILABILITY GROUP permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.

Using SQL Server Management Studio

To remove an availability group listener

1. In Object Explorer, connect to the server instance that hosts the primary replica, and click the server name to expand the server tree.
2. Expand the **AlwaysOn High Availability** node and the **Availability Groups** node.
3. Expand the node of the availability group, and expand the **Availability Groups Listeners** node.
4. Right-click the listener to be removed, and select the **Delete** command.
5. This opens the **Remove Listener from Availability Group** dialog box. For more information, see Remove Listener from Availability Group, later in this topic.

Remove Listener from Availability Group (Dialog Box)

Name

The name of the listener to be removed.

Result

Displays a link, either **Success** or **Error**, which you can click for more information.

Using Transact-SQL

To remove an availability group listener

1. Connect to the server instance that hosts the primary replica.
2. Use the ALTER AVAILABILITY GROUP statement, as follows:

   ALTER AVAILABILITY GROUP group_name REMOVE LISTENER 'dns_name'

   where group_name is the name of the availability group and dns_name is the DNS name of the availability group listener.

   The following example deletes the listener of the `AccountsAG` availability group. The DNS name is AccountsAG_Listener.

```
ALTER AVAILABILITY GROUP AccountsAG REMOVE LISTENER
'AccountsAG_Listener';
```

Using PowerShell

To remove an availability group listener

1. Set default (**cd**) to the server instance that hosts the primary replica.
2. Use the built in **Remove-Item** cmdlet to remove a listener. For example, the following command removes a listener named `MyListener` from an availability group named `MyAg`.

   ```
   Remove-Item `
   SQLSERVER:\Sql\PrimaryServer\InstanceName\AvailabilityGroups\MyAg\AGLi
   steners\MyListener
   ```

   Note
   To view the syntax of a cmdlet, use the **Get-Help** cmdlet in the SQL Server PowerShell environment. For more information, see SQL Server PowerShell Help.

Related Tasks

- Create or Configure an Availability Group Listener (SQL Server)
- View Availability Group Listener Properties (SQL Server)

See Also

AlwaysOn Availability Groups (SQL Server)

Client Connectivity and Application Failover (AlwaysOn Availability Groups)

Remove an Availability Group

This topic describes how to delete (drop) an AlwaysOn availability group by using SQL Server Management Studio, Transact-SQL, or PowerShell in SQL Server 2012. You can drop an availability group from any Windows Server Failover Clustering (WSFC) node that possesses the correct security credentials for the availability group. This enables you to delete an availability group when none of its availability replicas remain. Dropping an availability group deletes any associated availability group listener.

Note
If a server instance that hosts one of the availability replicas is offline when you delete an availability group, after coming online, the server instance will drop the local availability replica.

- **Before you begin:**

Limitations and Restrictions

Security

- **To delete an availability group, using:**

  SQL Server Management Studio

  Transact-SQL

  PowerShell

Before You Begin

Limitations and Restrictions

When the availability group is online, deleting it from a secondary-replica causes the primary replica to transition to the RESOLVING state.

If you delete an availability group from a computer that has been removed or evicted from the WSFC failover cluster, the availability group is only deleted locally.

Security

Permissions

Requires ALTER AVAILABILITY GROUP permission on the availability group, CONTROL AVAILABILITY GROUP permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission. To drop an availability group that is not hosted by the local server instance you need CONTROL SERVER permission or CONTROL permission on that Availability Group.

Using SQL Server Management Studio

To delete an availability group

1. In Object Explorer, connect to the server instance that hosts primary replica, if possible, or connect to another server instance that is enabled for AlwaysOn Availability Groups on a WSFC node that possess the correct security credentials for the availability group. Expand the server tree.
2. Expand the **AlwaysOn High Availability** node and the **Availability Groups** node.
3. This step depends on whether you want to delete multiple availability groups or only one availability group, as follows:
   - To delete multiple availability groups (whose primary replicas are on the connected server instance), use the **Object Explorer Details** pane to view and select all the availability groups that you want to delete. For more information, see Use Object Explorer Details to Monitor Availability Groups (SQL Server Management Studio).
   - To delete a single availability group, select it in either the **Object Explorer** pane or the **Object Explorer Details** pane.
4. Right-click the selected availability group or groups, and select the **Delete** command.
5. In the **Remove Availability Group** dialog box, to delete all the listed availability groups, click **OK**. If you do not want to remove all the listed availability groups, click **Cancel**.

Using Transact-SQL

To delete an availability group

1. Connect to the server instance that hosts the primary replica, if possible, or connect to another server instance that is enabled for AlwaysOn Availability Groups on a WSFC node that possess the correct security credentials for the availability group.
2. Use the DROP AVAILABILITY GROUP statement, as follows

   DROP AVAILABILITY GROUP *group_name*

   where group_name is the name of the availability group to be dropped.

   The following example deletes the `MyAG` availability group.

   ```
   DROP AVAILABILITY GROUP MyAG;
   ```

Using PowerShell

To delete an availability group

In the SQL Server PowerShell provider:

1. Change directory (**cd**) to the server instance that hosts the primary replica, if possible, or connect to another server instance that is enabled for AlwaysOn Availability Groups on a WSFC node that possess the correct security credentials for the availability group.
2. Use the **Remove-SqlAvailabilityGroup** cmdlet.

   For example, the following command removes the availability group named `MyAg`. This command can be executed on any server instance that hosts an availability replica for the availability group.

   ```
   Remove-SqlAvailabilityGroup `

   -Path SQLSERVER:\Sql\Computer\Instance\AvailabilityGroups\MyAg
   ```

   Note
   To view the syntax of a cmdlet, use the **Get-Help** cmdlet in the SQL Server PowerShell environment. For more information, see Get Help SQL Server PowerShell.

To set up and use the SQL Server PowerShell provider

- SQL Server PowerShell Provider

See Also

AlwaysOn Availability Groups

Creation and Configuration of Availability Groups

Troubleshoot a Failed Add-File Operation (AlwaysOn Availability Groups)

In some AlwaysOn availability group deployments, file paths differ between the system that hosts the primary replica and systems that host a secondary replica. If the file path of an add-file operation does not exist on a secondary replica, the add-file operation will succeed on the primary database. But the add-file operation will cause the secondary database to be suspended. This, in turn, causes the secondary replica to enter the NOT SYNCHRONIZING state.

Note

We recommend that, if possible, the file path (including the drive letter) of a given secondary database be identical to the path of the corresponding primary database.

Problem Resolution

To resolve this problem the database owner must complete the following steps:

1. Remove the secondary database from the availability group. For more information, see Removing a Database from a Secondary Replica (AlwaysOn Availability Groups).
2. On the existing secondary database, restore a full backup of the filegroup that contains the added file to the secondary database, using WITH NORECOVERY and WITH MOVE (specifying the file path on the server instance that hosts the secondary replica). For more information, see How to: Restore a Database to a New Location (SQL Server Management Studio).
3. Back up the transaction log that contains the add-file operation on the primary database, and manually restore the log backup on the secondary database using WITH NORECOVERY and WITH MOVE.
4. Prepare the secondary database for re-joining the availability group, by restoring, WITH NO RECOVERY, any other outstanding log backups from the primary database.
5. Rejoin the secondary database to the availability group. For more information, see Joining a Secondary Database to an Availability Group (SQL Server).

See Also

Overview of AlwaysOn Availability Groups

Manually Prepare a Secondary Database for an Availability Group (SQL Server)

Troubleshoot Orphaned Users (SQL Server)

Troubleshoot AlwaysOn Availability Groups Configuration (SQL Server)

AlwaysOn Policies for Operational Issues with AlwaysOn Availability Groups

The AlwaysOn Availability Groups health model evaluates a set of predefined policy based management (PBM) policies. You can use theses for viewing the health of an availability group and its availability replicas and databases in SQL Server 2012.

In this Topic:

- Terms and Definitions
- Predefined Policies and Issues
- AlwaysOn Dashboard
- Extending the AlwaysOn Health Model
- Related Tasks
- Related Content

Terms and Definitions

AlwaysOn predefined policies

A set of built-in policies that allow a database administrator to check an availability group and its availability replicas and databases for compliance with the states that are defined by the AlwaysOn policies.

AlwaysOn Availability Groups

A high-availability and disaster-recovery solution that provides an enterprise-level alternative to database mirroring.

availability group

A container for a discrete set of user databases, known as *availability databases*, that fail over together.

availability replica

An instantiation of an availability group that is hosted by a specific instance of SQL Server and that maintains a local copy of each availability database that belongs to the availability group. Two types of availability replicas exist: a single *primary replica* and one to four *secondary replicas*. The server instances that host the availability replicas for a given availability group must reside on different nodes of a single Windows Server Failover Clustering (WSFC) cluster.

availability database

A database that belongs to an availability group. For each availability database, the availability group maintains a single read-write copy (the *primary database*) and one to four read-only copies (*secondary databases*).

AlwaysOn Dashboard

A SQL Server Management Studio dashboard that provides an at-a-glance view of the health of an availability group. For more information, see AlwaysOn Dashboard, later in this topic.

Predefined Policies and Issues

The following table summarizes the predefined policies.

Policy name	Issue	Category*	Facet
WSFC Cluster State	WSFC cluster service is offline.	Critical	Instance of SQL Server
Availability Group Online State	Availability group is offline.	Critical	Availability group
Availability Group Automatic Failover Readiness	Availability group is not ready for automatic failover.	Critical	Availability group
Availability Replicas Data Synchronization State	Some availability replicas are not synchronizing data.	Warning	Availability group
Synchronous Replicas Data Synchronization State	Some synchronous replicas are not synchronized.	Warning	Availability group
Availability Replicas Role State	Some availability replicas do not have a healthy role.	Warning	Availability group
Availability Replicas Connection State	Some availability replicas are disconnected.	Warning	Availability group
Availability Replica Role State	Availability replica does not have a healthy role.	Critical	Availability replica
Availability Replica Connection State	Availability replica is disconnected.	Critical	Availability replica
Availability Replica Join State	Availability Replica is not joined.	Warning	Availability replica
Availability Replica Data Synchronization State	Data synchronization state of some availability database is not healthy.	Warning	Availability replica
Availability Database Suspension State	Availability database is suspended.	Warning	Availability database
Availability Database Join State	Secondary database is not joined.	Warning	Availability database
Availability Database	Data synchronization	Warning	Availability database

Policy name	Issue	Category*	Facet
Data Synchronization State	state of availability database is not healthy.		

Important
*For AlwaysOn policies, the category names are used as IDs. Changing the name of an AlwaysOn category would break its health-evaluation functionality. Therefore, do not modify the names of AlwaysOn categories.

AlwaysOn Dashboard

The AlwaysOn Dashboard gives you an at-a-glance view of the health of an availability group. The AlwaysOn Dashboard includes the following features:

- Enables you to easily display details about a given availability group, its availability replicas, and its databases.
- Displays visual indications of key states to help database administrators make quick operational decisions.
- Provides launch points for troubleshooting scenarios.
- For a given operational issue, populates the **Policy Evaluation Result** dialog box with information about specific AlwaysOn health policy violations and with links to remediation help.
- Provides an health extended event viewer to show previous events for AlwaysOn-specific issues.
- If failing over the availability group is a possible remediation for an issue, provides a launch point for the links Fail Over Availability Group Wizard. This wizard takes a database administrator through the manual failover process.

Extending the AlwaysOn Health Model

Extending the AlwaysOn Availability Groups health model is simply a matter of creating your own user-defined policies and putting them into certain categories based on the type of object that you are monitoring. After you a alter few settings, the AlwaysOn dashboard will automatically evaluate your own user-defined policies, as well as the AlwaysOn predefined policies.

A user-defined policy can use any of the available PBM facets, including those used by the AlwaysOn predefined policies (see Predefined Policies and Issues, earlier in this topic). The Server facet provides the following properties for monitoring AlwaysOn Availability Groups health: (**IsHadrEnabled** and **HadrManagerStatus**). The Server facet also provides properties the following policies for monitoring the WSFC cluster configuration: **ClusterQuorumType**, and **ClusterQuorumState**.

For more information, see The AlwaysOn Health Model Part 2 -- Extending the Health Model (a SQL Server AlwaysOn Team blog).

Related Tasks

- Use Policy-Based Management to View the Health of an Availability Group (SQL Server)
- Use the Availability Group Dashboard (SQL Server Management Studio)
- WSFC Disaster Recovery through Forced Quorum (SQL Server)
- Force a WSFC Cluster to Start Without a Quorum
- Perform a Forced Manual Failover of an Availability Group (SQL Server)
- Troubleshoot a Failed Add-File Operation (AlwaysOn Availability Groups)

Related Content

- The AlwaysOn Health Model Part 1 -- Health Model Architecture
- The AlwaysOn Health Model Part 2 -- Extending the Health Model
- Microsoft SQL Server AlwaysOn Solutions Guide for High Availability and Disaster Recovery

See Also

AlwaysOn Availability Groups (SQL Server)

Overview of AlwaysOn Availability Groups (SQL Server)

Administration of an Availability Group (SQL Server)

Monitoring of Availability Groups (SQL Server)

Use AlwaysOn Policies to View the Health of an Availability Group

This topic describes how to determine the operational health of an AlwaysOn availability group by using an AlwaysOn policy in SQL Server Management Studio or PowerShell in SQL Server 2012. For information about AlwaysOn Policy Based Management, see Policy-Based Management of Operational Issues with AlwaysOn Availability Groups (SQL Server).

Important
For AlwaysOn policies, the category names are used as IDs. Changing the name of an AlwaysOn category would break its health-evaluation functionality. Therefore, the names of AlwaysOn category should never be modified.

- **Before you begin:** Security
- **Use AlwaysOn policies to view the health of an availability group, using:**

  AlwaysOn Dashboard

  PowerShell

Before You Begin

Security

Permissions

Requires CONNECT, VIEW SERVER STATE, and VIEW ANY DEFINITION permissions.

Using the AlwaysOn Dashboard

To open the AlwaysOn Dashboard

1. In Object Explorer, connect to the server instance that hosts one of the availability replicas. To view information about all of the availability replicas in an availability group, use to the server instance that hosts the primary replica.
2. Click the server name to expand the server tree.
3. Expand the **AlwaysOn High Availability** node.

   Either right-click the **Availability Groups** node or expand this node and right-click a specific availability group.
4. Select the **Show Dashboard** command.

For information about how to use the AlwaysOn Dashboard, see Use the AlwaysOn Group Dashboard (SQL Server Management Studio).

Using PowerShell

Use AlwaysOn policies to view the health of an availability group

1. Set default (**cd**) to a server instance that hosts one of the availability replicas. To view information about all of the availability replicas in an availability group, use to the server instance that hosts the primary replica.
2. Use the following cmdlets:

   Test-SqlAvailabilityGroup

   Assesses the health of an availability group by evaluating SQL Server policy based management (PBM) policies. You must have CONNECT, VIEW SERVER STATE, and VIEW ANY DEFINITION permissions to execute this cmdlet.

   For example, the following command shows all availability groups with a health state of "Error" on the server instance `Computer\Instance`.

   ```
   Get-ChildItem SQLSERVER:\Sql\Computer\Instance\AvailabilityGroups `

   | Test-SqlAvailabilityGroup | Where-Object { $_.HealthState -eq "Error" }
   ```

   Test-SqlAvailabilityReplica

   Assesses the health of availability replicas by evaluating SQL Server policy based management (PBM) policies. You must have CONNECT, VIEW SERVER STATE, and VIEW ANY DEFINITION permissions to execute this cmdlet.

   For example, the following command evaluates the health of the availability replica named

`MyReplica` in the availability group `MyAg` and outputs a brief summary.

```
Test-SqlAvailabilityReplica `
-Path
SQLSERVER:\Sql\Computer\Instance\AvailabilityGroups\MyAg\AvailabilityRepl
icas\MyReplica
```

Test-SqlDatabaseReplicaState

Assesses the health of an availability database on all joined availability replicas by evaluating SQL Server policy based management (PBM) policies.

For example, the following command evaluates the health of all availability databases in the availability group `MyAg` and outputs a brief summary for each database.

```
Get-ChildItem
SQLSERVER:\Sql\Computer\Instance\AvailabilityGroups\MyAg\DatabaseReplicaS
tates `
  | Test-SqlDatabaseReplicaState
```

These cmdlets accept the following options:

Option	Description
AllowUserPolicies	Runs user policies found in the AlwaysOn policy categories.
InputObject	A collection of objects that, represent availability groups, availability replicas, or availability database states (depending on which cmdlet you are using). The cmdlet will compute the health of the specified objects.
NoRefresh	When this parameter is set, the cmdlet will not manually refresh the objects specified by the **-Path** or **-InputObject** parameter.
Path	The path to the availability group, one or more availability replicas, or database replica cluster state of the availability database (depending on which cmdlet you are using). This is an optional parameter. If not specified, the value of this parameter defaults to the current working location.
ShowPolicyDetails	Shows the result of each policy evaluation performed by this cmdlet. The cmdlet outputs one object per policy evaluation, and this object has fields describing the results of evaluation (whether the policy passed or not, the policy name and category, and so forth).

For example, the following **Test-SqlAvailabilityGroup** command specifies the **-ShowPolicyDetails** parameter to show the result of each policy evaluation performed by this cmdlet for each policy-based management (PBM) policy that was executed on the availability group named `MyAg`.

```
Test-SqlAvailabilityGroup `
-Path SQLSERVER:\Sql\Computer\Instance\AvailabilityGroups\AgName `
-ShowPolicyDetails
```

Note

To view the syntax of a cmdlet, use the **Get-Help** cmdlet in the SQL Server PowerShell environment. For more information, see SQL Server PowerShell Help.

To set up and use the SQL Server PowerShell provider

- Using the SQL Server PowerShell Provider
- SQL Server PowerShell Help

Related Content

SQL Server AlwaysOn Team Blogs—Monitoring AlwaysOn Health with PowerShell:

- Part 1: Basic Cmdlet Overview
- Part 2: Advanced Cmdlet Usage
- Part 3: A Simple Monitoring Application
- Part 4: Integration with SQL Server Agent

See Also

Overview of AlwaysOn Availability Groups (SQL Server)

Administration of an Availability Group (SQL Server)

Monitoring of Availability Groups (SQL Server)

Policy-Based Management of Operational Issues with AlwaysOn Availability Groups (SQL Server)

Use the AlwaysOn Dashboard (SQL Server Management Studio)

Database administrators use the AlwaysOn Dashboard to obtains an at-a-glance view the health of an AlwaysOn availability group and its availability replicas and databases in SQL Server 2012. Some of the typical uses for the AlwaysOn Dashboard are:

- Choosing a replica for a manual failover.
- Estimating data loss if you force failover.
- Evaluating data-synchronization performance.
- Evaluating the performance impact of a synchronous-commit secondary replica

The AlwaysOn Dashboard provides key availability group states and performance indicators allowing you to easily make high availability operational decisions using the following types of information.

- Replica roll-up state
- Synchronization mode and state
- Estimate Data Loss
- Estimated Recovery Time (redo catch up)
- Database Replica details
- Synchronization mode and state

- Time to restore log

In This Topic:

- **Before you begin:**

  Prerequisites

  Security

  Permissions
- **Getting started with:**

  AlwaysOn Dashboard

  To Change AlwaysOn Dashboard Options
- **Dashboard panes:**

  Availability Groups Summary

  Availability Group Details

  Availability Replica Details

  To Group Availability Group Information
- Related Tasks

Before You Begin

Prerequisites

You must be connected to the instance of SQL Server (server instance) that hosts either the primary replica or a secondary replica of an availability group.

Security

Permissions

Requires CONNECT, VIEW SERVER STATE, and VIEW ANY DEFINITION permissions.

To start the AlwaysOn Dashboard

1. In Object Explorer, connect to the instance of SQL Server on which you want to run the AlwaysOn Dashboard.
2. Expand the **AlwaysOn High Availability** node, right-click the **Availability Groups** node, and then click **Show Dashboard**.

To Change AlwaysOn Dashboard Options

You can use the SQL Server Management Studio **Options** dialog box to configure the SQL Server AlwaysOn Dashboard behavior for automatic refreshing and enabling an auto-defined AlwaysOn policy.

1. From the **Tools** menu, click **Options**.

2. To automatically refresh the dashboard, in the **Options** dialog box, select **Turn on automatic refresh**, enter the refresh interval in seconds, and then enter the number of times you want to retry the connection.
3. To enable a user-defined policy, select **Enable user-defined AlwaysOn policy**.

Availability Group Summary

The availability group screen displays a summary line for each availability group for which the connected server instance hosts a replica. This pane displays the following columns.

Availability Group Name

The name of an availability group for which the connected server instance hosts a replica.

Primary Instance

Name of the server instance that is hosting the primary replica of the availability group.

Failover Mode

Displays the failover mode for which the replica is configured. The possible failover mode values are:

- **Automatic**. Indicates that one or more replicas is in automatic-failover mode.
- **Manual**. Indicates that no replica is automatic-failover mode.

Issues

Click the **Issues** link to open troubleshooting documentation for a given issue. For a list of all the AlwaysOn policy issues, see Policy-Based Management of AlwaysOn Availability Groups (SQL Server).

Tip

Click the column headings to sort the availability group information by the name of the availability group, primary instance, failover mode, or Issue.

Availability Group Details

The following detail information is displayed for the availability group that you select from the summary screen:

Availability group state

Displays the state of health for the availability group.

Primary instance

Name of the server instance that is hosting the primary replica of the availability group.

Failover mode

Displays the failover mode for which the replica is configured. The possible failover mode values are:

- **Automatic**. Indicates that one or more replicas is in automatic-failover mode.

- **Manual**. Indicates that no replica is automatic-failover mode.

Cluster state

Name and state of the cluster where the instance of the connected server and the availability group is a member node.

Availability Replica Details

The **Availability replica** pane displays the following columns:

Name

The name of the server instance that hosts the availability replica. This column is shown by default.

Role

Indicates the current role of the availability replica, either **Primary** or **Secondary**. For information about AlwaysOn Availability Groups roles, see "HADR" Overview (SQL Server). This column is shown by default.

Failover Mode

Displays the failover mode for which the replica is configured. The possible failover mode values are:

- **Automatic**. Indicates that one or more replicas is in automatic-failover mode.
- **Manual**. Indicates that no replica is automatic-failover mode.

Synchronization State

Indicates whether a secondary replica is currently synchronized with primary replica. This column is shown by default. The possible values are:

- **Not Synchronized**. One or more databases in the replica are not synchronized or have not yet been joined to the availability group.
- **Synchronizing**. One or more databases in the replica are being synchronized.
- **Synchronized**. All databases in the secondary replica are synchronized with the corresponding primary databases on the current primary replica, if any, or on the last primary replica.

  Note

  In performance mode, the database is never in the synchronized state.

- **NULL**. Unknown state. This value occurs when the local server instance cannot communicate with the WSFC failover cluster (that is the local node is not part of WSFC quorum).

Issues

Lists the issue name. This value is shown by default. For a list of all the AlwaysOn policy issues, see Policy-Based Management of AlwaysOn Availability Groups (SQL

Server).

Availability Mode

Indicates the replica property that that you set separately for each availability replica. This value is hidden by default. The possible values are:

- **Asynchronous**. The secondary replica never becomes synchronized with the primary replica.
- **Synchronous**. When catching up to the primary database, a secondary database enters this state, and it remains caught up as long as data synchronization continues for the database.

Primary Connection Mode

Indicates the mode that is used to connect to the primary replica. This value is hidden by default.

Secondary Connection Mode

Indicates the mode that is used to connect to the secondary replica. This value is hidden by default.

Connection State

Indicates whether a secondary replica is currently connected to the primary replica. This column is hidden by default. The possible values are:

- **Disconnected**. For a remote availability replica, indicates that it is disconnected from the local availability replica. The response of the local replica to the Disconnected state depends on its role, as follows:
  - On the primary replica, if a secondary replica is disconnected, the secondary databases are marked as **Not Synchronized** on the primary replica, and the primary replica waits for the secondary to reconnect.
  - On the secondary replica, upon detecting that it is disconnected, the secondary replica attempts to reconnect to the primary replica.
- **Connected**. A remote availability replica that is currently connected to the local replica.

Operational State

Indicates the current operational state of the secondary replica. This value is hidden by default. The possible values are:

0. Pending failover

1. Pending

2. Online

3. Offline

4. Failed

5. Failed, no quorum

NULL. Replica is not local

Last Connection Error No.

Number of the last connection error. This value is hidden by default.

Last Connection Error Description

Description of the last connection error. This value is hidden by default.

Last Connection Error Timestamp

Timestamp of the last connection error. This value is hidden by default.

Note

For information about performance counters for availability replicas, see SQL Server, HADR Availability Replica.

To Group Availability Group Information

To group the information, click **Group by**, and select one of the following:

- **Availability replicas**
- **Availability databases**
- **Synchronization state**
- **Failover readiness**
- **Issues**

The pane that displays the grouped information displays the following columns:

Name

The name of the availability database. This value is shown by default.

Replica

The name of the instance of SQL Server that hosts the availability replica. This value is shown by default.

Synchronization State

Indicates whether the availability database is currently synchronized with primary replica. This value is shown by default. The possible synchronization states are:

- **Not synchronizing**.
  - For the primary role, indicates that the database is not ready to synchronize its transaction log with the corresponding secondary databases.
  - For a secondary database, indicates that the database has not started log synchronization because of a connection issue, is being suspended, or is going through transition states during startup or a role switch.
- **Synchronizing**.

  On a primary replica:
  - For a primary database, indicates that this database is ready to accept a scan

request from a secondary database.

- On a secondary replica, indicates that there is active data movement going on for that secondary database.

On a secondary replica, indicates that there is active data movement going on for that replica.

- **Synchronized**.

  For a primary database, indicates that at least one secondary database is synchronized.

  For a secondary database, indicates that the database is synchronized with the corresponding primary database.

- **Reverting**.

  Indicates the phase in the undo process when a secondary database is actively getting pages from the primary database.

  Caution
  When a database is in the REVERTING state, forcing failover to the secondary replica can leave that database in a state in which it cannot be started.

- **Initializing**.

  Indicates the phase of undo when the transaction log required for a secondary database to catch up to the undo LSN is being shipped and hardened on a secondary replica.

  Caution
  When a database is in the INITIALIZING state, forcing failover to the secondary replica will always leave that database in a state in which it cannot be started.

Failover Readiness

Indicates which availability replica can be failed over with or without potential data loss. This column is shown by default. The possible values are:

- **Data Loss**
- **No Data Loss**

Issues

Lists the issue name. This column is shown by default. The possible values are:

- **Warnings**. Click to display the thresholds and warnings issues.
- **Critical**. Click to display the critical issues.

For a list of all the AlwaysOn policy issues, see Policy-Based Management of AlwaysOn Availability Groups (SQL Server).

Suspended

Indicates whether the database is **Suspended** or has been **Resumed**. This value is hidden by default.

Suspend Reason

Indicates the reason for the suspended state. This value is hidden by default.

Estimate Data Loss (seconds)

Indicates the time difference of the last transaction log record in the primary replica and secondary replica. If the primary replica fails, all transaction log records within the time window will be lost. This value is hidden by default.

Estimated Recovery Time (seconds)

Indicates the time in seconds it takes to redo the catch-up time. The *catch-up time* is the time it will take for the secondary replica to catch up with the primary replica. This value is hidden by default.

Synchronization Performance (seconds)

Indicates the time in seconds it takes to synchronize between the primary and secondary replicas. This value is hidden by default.

Log Send Queue Size (KB)

Indicates the amount of log records in the log files of the primary database that have not been sent to the secondary replica. This value is hidden by default.

Log Send Rate (KB/sec)

Indicates the rate in KB per second at which log records are being sent to the secondary replica This value is hidden by default.

Redo Queue Size (KB)

Indicates the amount of log records in the log files of the secondary replica that have not yet been redone. This value is hidden by default.

Redo Rate (KB/sec)

Indicates the rate in KB per second at which the log records are being redone. This value is hidden by default.

FileStream Send Rate (KB/sec)

Indicates the rate of the FileStream in KB per second at which transactions are being sent to the replica. This value is hidden by default.

End of Log LSN

Indicates the actual log sequence number (LSN) that corresponds to the last log record in the log cache on the primary and secondary replicas. This value is hidden by default.

Recovery LSN

Indicates the end of the transaction log before the replica writes any new log records after recovery or failover on the primary replica. This value is hidden by default.

Truncation LSN

Indicates the minimum log truncation value for the primary replica. This value is hidden by default.

Last Commit LSN

Indicates the actual LSN corresponding to the last commit record in the transaction log. This value is hidden by default.

Last Commit Time

Indicates the time corresponding to the last commit record. This value is hidden by default.

Last Sent LSN

Indicates the point up to which all log blocks have been sent by the primary replica. This value is hidden by default.

Last Sent Time

Indicates the time when the last log block was sent. This value is hidden by default.

Last Received LSN

Indicates the point up to which all log blocks have been received by the secondary replica that hosts the secondary database. This value is hidden by default.

Last Received Time

Indicates the time when the log block identifier in last message received was read on the secondary replica. This value is hidden by default.

Last Hardened LSN

Indicates the point up to which all log records have been flushed to disk on the secondary replica. This value is hidden by default.

Last Hardened Time

Indicates the time when the log-block identifier was received for the last hardened LSN on the secondary replica. This value is hidden by default.

Last Redone LSN

Indicates the actual LSN of the log record that was redone last on the secondary replica. This value is hidden by default.

Last Redone Time

Indicates the time when the last log record was redone on the secondary database. This value is hidden by default.

Related Tasks

- Use Policy-Based Management to Monitor an Availability Group (SQL Server)

See Also

sys.dm_os_performance_counters (Transact-SQL)

Monitoring of Availability Groups (SQL Server)

Options (SQL Server AlwaysOn, Dashboard Page)

Use the **SQL Server AlwaysOn Dashboard** page of the SQL Server Management Studio **Options** dialog box to configure the AlwaysOn Dashboard.

To access this page:

On the **Tools** menu, click **Options**, expand the **SQL Server AlwaysOn** folder, and then click **Dashboard**.

On This Page

Turn on automatic refresh.

Click to enable automatic refresh. The options are:

- The **Refresh interval (in seconds)** field displays the number of seconds at which the dashboard will refresh. The default value is 30. When automatic refresh is enabled, you can edit this field to change the refresh interval.
- The **Number of connection retries** displays the number of times that the dashboard will attempt to connect to an instance of SQL Server that hosts an availability replica for an availability group that the Dashboard is monitoring. The default value is 65535. When automatic refresh is enabled, you can edit this field to change the number of connection retries.

Enable your user-defined AlwaysOn policy.

If you have defined your own AlwaysOn policy, click this option to enable your policy.

See Also

Use the Availability Group Dashboard (SQL Server Management Studio)

Policy Evaluation Result (AlwaysOn)

Use the **Policy Evaluation Result** page of the AlwaysOn Dashboard to view any current policy issues.

In This Topic:

- Dialog-Box Options
- Related Tasks

Dialog-Box Options

Detected Issue

Displays a row for each detected issue. The icon associated with an issue provides a visual indicator to indicate the severity of the issue, as follows:

Icon	Severity

❌	Critical
⚠	Warning

Description

This grid displays a brief description of the selected issue.

More Information

Click this link to open the help topic for the selected issue.

Related Tasks

- Use the Availability Group Dashboard (SQL Server Management Studio)
- Use Policy-Based Management to View the Health of an Availability Group (SQL Server)

See Also

AlwaysOn Availability Groups

Policy-Based Management of Operational Issues with AlwaysOn Availability Groups (SQL Server)

WSFC cluster service is offline

Introduction

Policy Name	WSFC Cluster State
Issue	WSFC cluster service is offline.
Category	**Critical**
Facet	Instance of SQL Server

Description

This policy checks the state of the Windows Server Failover Cluster (WSFC). The policy is in an unhealthy state and an alert is raised when the WSFC cluster is offline or in the forced quorum state. All availability groups hosted within this cluster are offline or a disaster recovery action is required.

The policy state is healthy when the cluster state is in the normal quorum.

Note

For this release of SQL Server 2012, information about possible causes and solutions is located at WSFC cluster service is offline on the TechNet Wiki.

Possible Causes

This issue can be caused by a cluster service issue or by the loss of the quorum in the cluster.

Possible Solution

Use the Cluster Administrator tool to perform the forced quorum or disaster recovery workflow. If you cannot resolve the issue by performing the forced quorum or disaster recovery, contact your cluster administrator to help resolve this issue. For more information, see Force a WSFC Cluster to Start Without a Quorum in SQL Server Books Online.

See Also

AlwaysOn Availability Groups (SQL Server)

Use the Availability Group Dashboard (SQL Server Management Studio)

Availability group is offline

Introduction

Policy Name	Availability Group Online State
Issue	Availability group is offline.
Category	**Critical**
Facet	Availability group

Description

This policy checks the online or offline state of the availability group. The policy is in an unhealthy state and an alert is raised when the cluster resource of the availability group is offline or the availability group does not have a primary replica.

The policy state is healthy when the cluster resource of the availability group is online and the availability group has a primary replica.

Note
For this release of SQL Server 2012, information about possible causes and solutions is located at Availability group is offline on the TechNet Wiki.

Possible Causes

This issue can be caused by a failure in the server instance that hosts the primary replica or by the Windows Server Failover Cluster (WSFC) availability group resource going offline. Following are possible causes for the availability group to be offline:

- The availability group is not configured with automatic failover mode. The primary replica becomes unavailable and the role of all replicas in the availability group become RESOLVING.
    - The primary replica instance service is down or unresponsive.

    - The availability group has a connectivity issue with the cluster.
- The availability group is configured with automatic failover mode and does not complete successfully.
    - During the automatic failover, the primary readiness check on the target replica fails, and there is no replica available to become the new primary.
- The availability group resource in the cluster becomes offline.
    - Any dependent cluster resource encounters a critical issue and becomes offline. The availability group resource is also offline until the dependent resource becomes online.
    - A critical issue in the cluster turns off the availability group resource.
- There is an automatic, manual, or forced failover in progress for the availability group.

Possible Solutions

Following are possible solutions for this issue:

- If the SQL Server instance of the primary replica is down, restart the server and then verify that the availability group recovers to a healthy state.
- If the automatic failover appears to have failed, verify that the databases on the replica are synchronized with the previously known primary replica, and then failover to the primary replica. If the databases are not synchronized, select a replica with a minimum loss of data, and then recover to failover mode.
- If the resource in the cluster is offline while the instances of SQL Server appear to be healthy, use Failover Cluster Manager to check the cluster health or other cluster issues on the server. You can also use the Failover Cluster Manager to attempt to turn the availability group resource online.
- If there is a failover in progress, wait for the failover to complete.

See Also

AlwaysOn Availability Groups (SQL Server)

Use the Availability Group Dashboard (SQL Server Management Studio)

Availability group is not ready for automatic failover

Introduction

Policy Name	Availability Group Automatic Failover Readiness
Issue	Availability group is not ready for automatic failover.
Category	**Critical**

Facet	Availability group

Description

This policy checks to verify that the availability group has at least one secondary replica that is failover ready. The policy is in an unhealthy state and an alert is raised when the failover mode of the primary replica is automatic, however none of the secondary replicas in the availability group are failover ready.

The policy is in a healthy state when at least one secondary replica is automatic failover ready.

Note
For this release of SQL Server 2012, information about possible causes and solutions is located at Availability group is not ready for automatic failover on the TechNet Wiki.

Possible Causes

The availability group is not ready for automatic failover. The primary replica is configured for automatic failover; however, the secondary replica is not ready for automatic failover. The secondary replica that is configured for automatic failover might be unavailable or its data synchronization state is currently not SYNCHRONIZED.

Possible Solutions

Following are possible solutions for this issue:

- Verify that at least one secondary replica is configured as automatic failover. If there is not a secondary replica configured as automatic failover, update the configuration of a secondary replica to be the automatic failover target with synchronous commit.
- Use the policy to verify that the data is in a synchronization state and the automatic failover target is SYNCHRONIZED, and then resolve the issue at the availability replica.

See Also

AlwaysOn Availability Groups (SQL Server)

Use the Availability Group Dashboard (SQL Server Management Studio)

Some availability replicas are not synchronizing data

Introduction

Policy Name	Availability Replicas Data Synchronization State
Issue	Some availability replicas are not synchronizing data.
Category	**Warning**

Facet	Availability group

Description

This policy rolls up the data synchronization state of all availability replicas in the availability group and checks if the synchronization of any availability replica is not operational. The policy is in an unhealthy state if any of the data synchronization states of the availability replica is NOT SYNCRONIZING.

This policy is in a healthy state if none of the data synchronization states of the availability replica is NOT SYNCHRONIZING.

Note
For this release of SQL Server 2012, information about possible causes and solutions is located at Some availability replicas are not synchronizing data on the TechNet Wiki.

Possible Causes

In this availability group, at least one secondary replica has a NOT SYNCHRONIZING synchronization state and is not receiving data from the primary replica.

Possible Solution

Use the availability replica policy state to find the availability replica with a NOT SYNCHROINIZING state, and then resolve the issue at the availability replica.

See Also

AlwaysOn Availability Groups (SQL Server)

Use the Availability Group Dashboard (SQL Server Management Studio)

Some synchronous replicas are not synchronized

Introduction

Policy Name	Synchronous Replicas Data Synchronization State
Issue	Some synchronous replicas are not synchronized.
Category	**Warning**
Facet	Availability group

Description

This policy rolls up the data synchronization state of all availability replicas and checks for any availability replicas that are not in the expected synchronization state. The policy is in an unhealthy state when any asynchronous replica is not in a SYNCHRONIZING state and any synchronous replica is not in a SYNCHRONIZED state. The policy state is otherwise healthy.

Note
For this release of SQL Server 2012, information about possible causes and solutions is located at Some synchronous replicas are not synchronized on the TechNet Wiki.

Possible Causes

In this availability group, at least one synchronous replica is not currently synchronized. The replica synchronization state could be either SYNCHONIZING or NOT SYNCHRONIZING.

Possible Solution

Use the availability replica policy state to find the availability replica with the incorrect synchronization state, and then resolve the issue at the availability replica.

See Also

AlwaysOn Availability Groups (SQL Server)

Use the Availability Group Dashboard (SQL Server Management Studio)

Some availability replicas do not have a healthy role

Introduction

Policy Name	Availability Replicas Role State
Issue	Some availability replicas do not have a healthy role.
Category	**Warning**
Facet	Availability group

Description

This policy rolls up the connection state of all availability replicas and checks if there are any availability replicas that are not in a healthy role. The policy is in an unhealthy state when any availability replica is neither primary nor secondary. The policy is otherwise in a healthy state.

Note
For this release of SQL Server 2012, information about possible causes and solutions is located at Some availability replicas do not have a healthy role on the TechNet Wiki.

Possible Causes

In this availability group, at least one availability replica does not currently have the primary or secondary role.

Possible Solution

Use the availability replica policy state to find the availability replica whose role is not primary or secondary, and then resolve the issue at the availability replica.

See Also

AlwaysOn Availability Groups (SQL Server)

Use the Availability Group Dashboard (SQL Server Management Studio)

Some availability replicas are disconnected

Introduction

Policy Name	Availability Replicas Connection State
Issue	Some availability replicas are disconnected.
Category	**Warning**
Facet	Availability group

Description

This policy rolls up the connection state of all availability replicas and checks for any availability replicas that are DISCONENCTED. The policy is in an unhealthy state when any availability replica is DISCONNECTED. The policy is otherwise in a healthy state.

Note
For this release of SQL Server 2012, information about possible causes and solutions is located at Some availability replicas are disconnected on the TechNet Wiki.

Possible Causes

In this availability group, at least one secondary replica is not connected to the primary replica. The connected state is DISCONNECTED.

Possible Solution

Use the availability replica policy state to find the availability replica that is DISCONNECTED, and then resolve the issue at the availability replica.

See Also

AlwaysOn Availability Groups (SQL Server)

Use the Availability Group Dashboard (SQL Server Management Studio)

Availability replica does not have a healthy role

Introduction

Policy Name	Availability Replica Role State
Issue	Availability replica does not have a healthy role.
Category	**Critical**
Facet	Availability replica

Description

This policy checks the state of the role of the availability replica. The policy is in an unhealthy state when the role of the availability replica is neither primary nor secondary. The policy is otherwise in a healthy state.

Note
For this release of SQL Server 2012, information about possible causes and solutions is located at Availability replica does not have a healthy role on the TechNet Wiki.

Possible Causes

The role of this availability replica is unhealthy. The replica does not have either the primary or secondary role.

Possible Solution: Information_still_to_come

See Also

AlwaysOn Availability Groups (SQL Server)

Use the Availability Group Dashboard (SQL Server Management Studio)

Availability replica is disconnected

Introduction

Policy Name	Availability Replica Connection State
Issue	Availability replica is disconnected.
Category	**Critical**

Facet	Availability replica

Description

This policy checks the connection state between availability replicas. The policy is in an unhealthy state when the connection state of the availability replica is DISCONNECTED. The policy is otherwise in a healthy state.

Note
For this release of SQL Server 2012, information about possible causes and solutions is located at Availability replica is disconnected on the TechNet Wiki.

Possible Causes

The secondary replica is not connected to the primary replica. The connected state is DISCONNECTED. This issue can be caused by the following:

- The connection port might be in conflict with another application.
- The encryption type or algorithm is mismatched.
- The connection endpoint has been deleted or has not been started.
- The transport is disconnected.

Possible Solutions

Following are possible solutions for this issue:

- Check the database mirroring endpoint configuration for the instances of the primary and secondary replica and update the mismatched configuration.
- Check if the port is conflicting, and if so, change the port number.

See Also

AlwaysOn Availability Groups (SQL Server)

Use the Availability Group Dashboard (SQL Server Management Studio)

Data synchronization state of availability database is not healthy

Introduction

Policy Name	Availability Database Data Synchronization State
Issue	Data synchronization state of availability database is not healthy.
Category	**Warning**

Facet	Availability database

Description

This policy rolls up the data synchronization state of all availability databases (also known as "database replicas") in the availability replica. The policy is in an unhealthy sate when any database replica is not in the expected data synchronization state. The policy is otherwise in a healthy state.

Note
For this release of SQL Server 2012, information about possible causes and solutions is located at Data synchronization state of some availability database is not healthy on the TechNet Wiki.

Possible Causes

The data synchronization state of this availability database is unhealthy. On an asynchronous-commit availability replica, every availability database should be in the SYNCHRONIZING state. On a synchronous-commit replica, every availability database must be in the SYNCHRONIZED state.

Possible Solution

Use the database replica policy to find the database replica with an unhealthy data synchronization state, and then resolve the issue at the database replica.

See Also

AlwaysOn Availability Groups (SQL Server)

Use the Availability Group Dashboard (SQL Server Management Studio)

Availability replica is not joined

Introduction

Policy Name	Availability Replica Join State
Issue	Availability Replica is not joined.
Category	**Warning**
Facet	Availability replica

Description

This policy checks the join state of the availability replica. The policy is in an unhealthy state when the availability replica is added to the availability group, but is not joined properly. The policy is otherwise in a healthy state.

Note
For this release of SQL Server 2012, information about possible causes and solutions is located at Availability replica is not joined on the TechNet Wiki.

Possible Causes

The secondary replica is not joined to the availability group. For an availability replica to be successfully joined to the availability group, the join state must be Joined Standalone Instance (1) or Joined Failover Cluster (2).

Possible Solution

Use Transact-SQL, PowerShell, or SQL Server Management Studio to join the secondary replica to the availability group. For more information about joining secondary replicas to availability groups, see Joining a Secondary Replica to an Availability Group (SQL Server).

See Also

AlwaysOn Availability Groups (SQL Server)

Use the Availability Group Dashboard (SQL Server Management Studio)

Availability database is suspended

Introduction

Policy Name	Availability Database Suspension State
Issue	Availability database is suspended.
Category	**Warning**
Facet	Availability database

Description

This policy checks the state of data movement of the secondary database (also known as a "secondary database replica"). The policy is in an unhealthy state when the data movement is suspended. The policy is otherwise in a healthy state.

Note
For this release of SQL Server 2012, information about possible causes and solutions is located at Availability database is suspended on the TechNet Wiki.

Possible Causes

Data synchronization on this availability database might have been suspended because of the following:

- Due to an error, the system might have suspended data synchronization.
- The database administrator might have suspended data synchronization for maintenance purposes.

Possible Solution

Resume data synchronization. If the issue persists, check the availability group in the Event log, and then diagnose why the system suspended data movement.

See Also

AlwaysOn Availability Groups (SQL Server)

Use the Availability Group Dashboard (SQL Server Management Studio)

Secondary database is not joined

Introduction

Policy Name	Availability Database Join State
Issue	Secondary database is not joined.
Category	**Warning**
Facet	Availability database

Description

This policy checks the join state of the secondary database (also known as a "secondary database replica"). The policy is in an unhealthy state when the dataset replica is not joined. The policy is otherwise in a healthy state.

Note
For this release of SQL Server 2012, information about possible causes and solutions is located at Secondary database is not joined on the TechNet Wiki.

Possible Causes

This secondary database is not joined to the availability group. The configuration of this secondary database is incomplete.

Possible Solution

Use Transact-SQL, PowerShell, or SQL Server Management Studio to join the secondary replica to the availability group. For more information about joining secondary replicas to availability groups, see Joining a Secondary Replica to an Availability Group (SQL Server).

See Also

AlwaysOn Availability Groups (SQL Server)

Use the Availability Group Dashboard (SQL Server Management Studio)

Data synchronization state of some availability database is not healthy

Introduction

Policy Name	Availability Replica Data Synchronization State
Issue	Data synchronization state of some availability database is not healthy.
Category	**Warning**
Facet	Availability replica

Description

This policy checks the data synchronization state of the availability database (also known as a "database replica"). The policy is in an unhealthy state when the data synchronization state is NOT SYNCHRONIZING or the state is not SYNCHRONIZED for the synchronous-commit database replica.

Note
For this release of SQL Server 2012, information about possible causes and solutions is located at Data synchronization state of availability database is not healthy on the TechNet Wiki.

Possible Causes

At least one availability database on the replica has an unhealthy data synchronization state. If this is an asynchronous-commit availability replica, all availability databases should be in the SYNCHRONIZING state. If this is a synchronous-commit availability replica, all availability databases should be in the SYNCHRONIZED state. This issue can be caused by the following:

- The availability replica might be disconnected.
- The data movement might be suspended.
- The database might not be accessible.
- There might be a temporary delay issue due to network latency or the load on the primary or secondary replica.

Possible Solution

Resolve any connection or data movement suspend issues. Check the events for this issue using SQL Server Management Studio, and find the database error. Follow the troubleshooting steps for the specific error.

See Also

AlwaysOn Availability Groups (SQL Server)

Use the Availability Group Dashboard (SQL Server Management Studio)

Monitoring of Availability Groups

To monitor the properties and state of an AlwaysOn availability group you can use the following tools.

Tool	**Brief Description**	**Links**
System Center Monitoring pack for SQL Server	The Monitoring pack for SQL Server (SQLMP) is the recommended solution for monitoring availability groups, availability replica and availability databases for IT administrators. Monitoring features that are particularly relevance to AlwaysOn Availability Groups include the following: • Automatic discoverability of availability groups, availability replicas, and availability database from among hundreds of computers. This enables you to easily keep track of your AlwaysOn Availability Groups inventory. • Fully capable System Center Operations Manager (SCOM) alerting and ticketing. These features provide detailed knowledge that enables faster resolution to a problem.	To download the monitoring pack (SQLServerMP.msi) and *SQL Server Management Pack Guide for System Center Operations Manager* (SQLServerMPGuide.doc), see: System Center Monitoring pack for SQL Server

Tool	Brief Description	Links
	• A custom extension to AlwaysOn Health monitoring using Policy Based management (PBM). • Health roll ups from availability databases to availability replicas. • Custom tasks that manage AlwaysOn Availability Groups from the System Center Operations Manager console.	
Transact-SQL	AlwaysOn Availability Groups catalog and dynamic management views provide a wealth of information about your availability groups and their replicas, databases, listeners, and WSFC cluster environment.	Monitor Availability Groups (Transact-SQL)
SQL Server Management Studio	The **Object Explorer Details** pane displays basic information about the availability groups hosted on the instance of SQL Server to which you are connected. **Tip** Use this pane to select multiple availability groups, replicas, or databases and to perform routine administrative tasks on the selected objects; for example, removing multiple availability replicas or databases from an availability group.	Use Object Explorer Details to Monitor Availability Groups (SQL Server Management Studio)

Tool	Brief Description	Links
SQL Server Management Studio	**Properties** dialog boxes enable you to view the properties of availability groups, replicas, or listeners and, in some cases, to change their values.	• View Availability Group Properties (SQL Server Management Studio) • View Availability Replica Properties (SQL Server Management Studio) • View Availability Group Listener Properties (SQL Server)
System Monitor	The **SQLServer:Availability Replica** performance object contains performance counters that report information about availability replicas.	SQL Server, HADR Availability Replica
System Monitor	The **SQLServer:Database Replica** performance object contains performance counters that report information about the secondary databases on a given secondary replica. The **SQLServer:Databases** object in SQL Server contains performance counters that monitor transaction log activities, among other things. The following counters are particularly relevant for monitoring transaction-log activity on availability databases: **Log Flush Write Time (ms)**, **Log Flushes/sec**, **Log Pool Cache Misses/sec**, **Log Pool Disk Reads/sec**, and **Log Pool Requests/sec**.	SQL Server, HADR Database Replica and SQL Server, Databases Object

See Also

AlwaysOn Availability Groups Catalog Views (Transact-SQL)

AlwaysOn Availability Groups Dynamic Management Views and Functions (Transact-SQL)

AlwaysOn Availability Groups

Automatic Page Repair (Availability Groups/Database Mirroring)

Use the AlwaysOn Group Dashboard (SQL Server Management Studio)

Monitor Availability Groups (Transact-SQL)

For monitoring availability groups and replicas and the associated databases by using Transact-SQL, AlwaysOn Availability Groups provides a set of catalog and dynamic management views and server properties. Using Transact-SQL SELECT statements, you can use the views to monitor availability groups and their replicas and databases. The information returned for a given availability group depends on whether you are connected to the instance of SQL Server that is hosting the primary replica or a secondary replica.

Tip

Many of these views can be joined using their ID columns to return information from multiple views in a single query.

In This Topic:

- Permissions
- **Using Transact-SQL to monitor:**

  AlwaysOn Availability Groups feature on a server instance

  Availability groups on the WSFC cluster

  Availability groups

  Availability replicas

  Availability databases

  Availability group listeners
- Related Tasks

Permissions

AlwaysOn Availability Groups catalog views require VIEW ANY DEFINITION permission on the server instance. AlwaysOn Availability Groups dynamic management views require VIEW SERVER STATE permission on the server.

Monitoring the AlwaysOn Availability Groups Feature on a Server Instance

To monitor the AlwaysOn Availability Groups feature on a server instance, use the following built-in function:

SERVERPROPERTY

Returns server property information about whether AlwaysOn Availability Groups is enabled and, if so, whether it has started on the server instance.

Column names: IsHadrEnabled, HadrManagerStatus

Monitoring Availability Groups on the WSFC Cluster

To monitor the Windows Server Failover Clustering (WSFC) cluster that hosts a local server instance that is enabled for AlwaysOn Availability Groups, use the following views:

sys.dm_hadr_cluster

If the Windows Server Failover Clustering (WSFC) node that hosts an instance of SQL Server with AlwaysOn Availability Groups enabled has WSFC quorum, **sys.dm_hadr_cluster** returns a row that exposes the cluster name and information about the quorum. If the WSFC node has no quorum, no rows are returned.

Column names: cluster_name, quorum_type, quorum_type_desc, quorum_state, quorum_state_desc

sys.dm_hadr_cluster_members

If the WSFC node that hosts the local AlwaysOn-enabled instance of SQL Server has WSFC quorum, returns a row for each of the members that constitute the quorum and the state of each of them.

Column names: member_name, member_type, member_type_desc, member_state, member_state_desc, number_of_quorum_votes

sys.dm_hadr_cluster_networks

Returns a row for every member that is participating in an availability group's subnet configuration. You can use this dynamic management view to validate the network virtual IP that is configured for each availability replica.

Column names: member_name, network_subnet_ip, network_subnet_ipv4_mask, network_subnet_prefix_length, is_public, is_ipv4

Primary key: member_name + network_subnet_IP + network_subnet_prefix_length

sys.dm_hadr_instance_node_map

For every instance of SQL Server that hosts an availability replica that is joined to its AlwaysOn availability group, returns the name of the Windows Server Failover Clustering (WSFC) node that hosts the server instance. This dynamic management view has the following uses:

- This dynamic management view is useful for detecting an availability group with multiple availability replicas that are hosted on the same WSFC node, which is an unsupported configuration that could occur after an FCI failover if the availability group is incorrectly configured.
- When multiple SQL Server instances are hosted on the same WSFC node, the Resource DLL uses this dynamic management view to determine the instance of SQL Server to connect to.

Column names: ag_resource_id, instance_name, node_name

sys.dm_hadr_name_id_map

Shows the mapping of AlwaysOn availability groups that the current instance of SQL Server has joined to three unique IDs: an availability group ID, a WSFC resource ID, and a WSFC

Group ID. The purpose of this mapping is to handle the scenario in which the WSFC resource/group is renamed.

Column names: ag_name, ag_id, ag_resource_id, ag_group_id

Note

Also see **sys.dm_hadr_availability_replica_cluster_nodes** and **sys.dm_hadr_availability_replica_cluster_states** in the Monitoring Availability Replicas section and **sys.availability_databases_cluster** and **sys.dm_hadr_database_replica_cluster_states** in the Monitoring Availability Databases section, later in this topic.

For information about WSFC clusters and AlwaysOn Availability Groups, see Windows Server Failover Clustering (WSFC) with SQL Server and Failover Clustering and AlwaysOn Availability Groups (SQL Server).

Monitoring Availability Groups

To monitor the availability groups for which the server instance hosts an availability replica, use the following views:

sys.availability_groups

Returns a row for each availability group for which the local instance of SQL Server hosts an availability replica. Each row contains a cached copy of the availability group metadata.

Column names: group_id, name, resource_id, resource_group_id, failure_condition_level, health_check_timeout, automated_backup_preference, automated_backup_preference_desc

sys.availability_groups_cluster

Returns a row for each availability group in the WSFC cluster. Each row contains the availability group metadata from the Windows Server Failover Clustering (WSFC) cluster.

Column names: group_id, name, resource_id, resource_group_id, failure_condition_level, health_check_timeout, automated_backup_preference, automated_backup_preference_desc

sys.dm_hadr_availability_group_states

Returns a row for each availability group that possesses an availability replica on the local instance of SQL Server. Each row displays the states that define the health of a given availability group.

Column names: group_id, primary_replica, primary_recovery_health, primary_recovery_health_desc, secondary_recovery_health, secondary_recovery_health_desc, synchronization_health, synchronization_health_desc

Monitoring Availability Replicas

To monitor availability replicas, use the following views and system function:

sys.availability_replicas

Returns a row for every availability replica in each availability group for which the local instance of SQL Server hosts an availability replica.

Column names: replica_id, group_id, replica_metadata_id, replica_server_name, owner_sid, endpoint_url, availability_mode, availability_mode_desc, failover_mode, failover_mode_desc, session_timeout, primary_role_allow_connections, primary_role_allow_connections_desc, secondary_role_allow_connections, secondary_role_allow_connections_desc, create_date, modify_date, backup_priority, read_only_routing_url

sys.availability_read_only_routing_lists

Returns a row for the read only routing list of each availability replica in an AlwaysOn availability group in the WSFC failover cluster.

Column names: replica_id, routing_priority, read_only_replica_id

sys.dm_hadr_availability_replica_cluster_nodes

Returns a row for every availability replica (regardless of join state) of the AlwaysOn availability groups in the Windows Server Failover Clustering (WSFC) cluster.

Column names: group_name, replica_server_name, node_name

sys.dm_hadr_availability_replica_cluster_states

Returns a row for each replica (regardless of join state) of all AlwaysOn availability groups (regardless of replica location) in the Windows Server Failover Clustering (WSFC) cluster.

Column names: replica_id, replica_server_name, group_id, join_state, join_state_desc

sys.dm_hadr_availability_replica_states

Returns a row showing the state of each local availability replica and a row for each remote availability replica in the same availability group.

Column names: replica_id, group_id, is_local, role, role_desc, operational_state, operational_state_desc, connected_state, connected_state_desc, recovery_health, recovery_health_desc, synchronization_health, synchronization_health_desc, last_connect_error_number, last_connect_error_description, and last_connect_error_timestamp

sys.fn_hadr_backup_is_preferred_replica

Determines whether the current replica is the preferred backup replica.

Note

For information about performance counters for availability replicas (the **SQLServer:Availability Replica** performance object), see SQL Server, HADR Availability Replica.

Monitoring Availability Databases

To monitor availability databases, use the following views:

sys.availability_databases_cluster

Contains one row for each database on the instance of SQL Server that are part of all AlwaysOn Availability Groups in the cluster, regardless of whether the local copy database has been joined to the availability group yet.

Note

When a database is added to an availability group, the primary database is automatically joined to the group. Secondary databases must be prepared on each secondary replica before they can be joined to the availability group.

Column names: group_id, group_database_id, database_name

sys.databases

Contains one row per database in the instance of SQL Server. If a database belongs to an availability replica, the row for that database displays the GUID of the replica and the unique identifier of the database within its availability group.

AlwaysOn Availability Groups column names: replica_id, group_database_id

sys.dm_hadr_auto_page_repair

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. This view contains rows for the latest automatic page-repair attempts on a given primary or secondary database, with a maximum of 100 rows per database. As soon as a database reaches the maximum, the row for its next automatic page-repair attempt replaces one of the existing entries.

Column names: database_id, file_id, page_id, error_type, page_status, modification_time

sys.dm_hadr_database_replica_states

Returns a row for each database that is participating in any availability group for which the local instance of SQL Server is hosting an availability replica.

Column names: database_id, group_id, replica_id, group_database_id, is_local, synchronization_state, synchronization_state_desc, is_commit_participant, synchronization_health, synchronization_health_desc, database_state, database_state_desc, is_suspended, suspend_reason, suspend_reason_desc, recovery_lsn, truncation_lsn, last_sent_lsn, last_sent_time, last_received_lsn, last_received_time, last_hardened_lsn, last_hardened_time, last_redone_lsn, last_redone_time, log_send_queue_size, log_send_rate, redo_queue_size, redo_rate, filestream_send_rate, end_of_log_lsn, last_commit_lsn, last_commit_time, low_water_mark_for_ghosts

sys.dm_hadr_database_replica_cluster_states

Returns a row containing information intended to provide you with insight into the health of the availability databases in each availability group on the Windows Server Failover Clustering (WSFC) cluster. This dynamic management view is useful when planning or responding to a failover or for discovering which secondary replica in an availability group is holding up log truncation on a given primary database.

Column names: replica_id, group_database_id, database_name, is_failover_ready, is_pending_secondary_suspend, is_database_joined, recovery_lsn, truncation_lsn

Note
The primary replica location is the authoritative source for an availability group.

Note
For information about the AlwaysOn Availability Groups performance counters for availability databases (the **SQLServer:Database Replica** performance object), see SQL Server, HADR Database Replica. Also, to monitor transaction-log activity on availability databases, use the following counters of the **SQLServer:Databases** performance object: **Log Flush Write Time (ms)**, **Log Flushes/sec**, **Log Pool Cache Misses/sec**, **Log Pool Disk Reads/sec**, and **Log Pool Requests/sec**. For more information, see SQL Server, Databases Object.

Monitoring Availability Group Listeners

To monitor the availability group listeners on subnets of the WSFC cluster, use the following views:

sys.availability_group_listener_ip_addresses

Returns a row for every conformant virtual IP address that is currently online for an availability group listener.

Column names: listener_id, ip_address, ip_subnet_mask, is_dhcp, network_subnet_ip, network_subnet_prefix_length, network_subnet_ipv4_mask, state, state_desc

sys.availability_group_listeners

For a given availability group, returns either zero rows indicating that no network name is associated with the availability group, or returns a row for each availability-group listener configuration in the WSFC cluster.

Column names: group_id, listener_id, dns_name, port, is_conformant, ip_configuration_string_from_cluster

sys.dm_tcp_listener_states

Returns a row containing dynamic-state information for each TCP listener.

Column names: listener_id, ip_address, is_ipv4, port, type, type_desc, state, state_desc, start_time

Primary key: listener_id

For information about availability group listeners, see Availability Group Listeners, Client Connectivity, and Application Failover (AlwaysOn Availability Groups).

Related Tasks

AlwaysOn Availability Groups monitoring tasks:

- Use Object Explorer Details to Monitor Availability Groups (SQL Server Management Studio)
- View Availability Group Properties (SQL Server Management Studio)
- View Availability Replica Properties (SQL Server Management Studio)
- View Availability Group Listener Properties (SQL Server)

AlwaysOn Availability Groups monitoring reference (Transact-SQL):

- SERVERPROPERTY (Transact-SQL)
- sys.availability_group_listener_ip_addresses (Transact-SQL)
- sys.availability_group_listeners (Transact-SQL)
- sys.availability_databases_cluster (Transact-SQL)
- sys.availability_groups (Transact-SQL)
- sys.availability_read_only_routing_lists (Transact-SQL)
- sys.availability_replicas (Transact-SQL)
- sys.dm_hadr_availability_replica_cluster_nodes (Transact-SQL)
- sys.dm_hadr_availability_replica_cluster_states (Transact-SQL)
- sys.database_mirroring_endpoints (Transact-SQL)
- sys.dm_hadr_auto_page_repair (Transact-SQL)
- sys.dm_hadr_availability_group_states (Transact-SQL)
- sys.dm_hadr_availability_replica_cluster_states (Transact-SQL)
- sys.dm_hadr_availability_replica_states (Transact-SQL)
- sys.dm_hadr_database_replica_states (Transact-SQL)
- sys.dm_hadr_database_replica_cluster_states (Transact-SQL)
- sys.dm_hadr_cluster (Transact-SQL)
- sys.dm_hadr_cluster_members (Transact-SQL)
- sys.dm_hadr_cluster_networks (Transact-SQL)
- sys.dm_hadr_database_replica_cluster_states (Transact-SQL)
- sys.dm_hadr_database_replica_states (Transact-SQL)
- sys.dm_hadr_instance_node_map
- sys.dm_hadr_name_id_map
- sys.dm_os_performance_counters (Transact-SQL)
- sys.dm_tcp_listener_states (Transact-SQL)
- sys.fn_hadr_backup_is_preferred_replica

AlwaysOn performance counters:

- SQL Server, HADR Availability Replica
- SQL Server, HADR Database Replica

- SQL Server, Databases Object

Policy-based management for AlwaysOn Availability Groups

- Use Policy-Based Management to View the Health of an Availability Group (SQL Server)
- Use the AlwaysOn Group Dashboard (SQL Server Management Studio)

See Also

AlwaysOn Availability Groups (SQL Server)

Overview of AlwaysOn Availability Groups

Monitoring (AlwaysOn Availability Groups)

Use the Object Explorer Details to Monitor Availability Groups (SQL Server Management Studio)

This topic describes how to use the **Object Explorer Details** pane of SQL Server Management Studio to monitor and manage existing AlwaysOn availability groups, availability replicas, and availability databases.

Note

For information about using the Object Explorer Details pane, see Using Object Explorer Details and SQL Server Object Search.

- **Before you begin:** Prerequisites
- **To Monitor an Availability Group, using:** SQL Server Management Studio
- **Object Explorer Details:**

  Availability Group Details

  Availability Replica Details

  Availability Database Details

Before You Begin

Prerequisites

You must be connected to the instance of SQL Server (server instance) that hosts either the primary replica or a secondary replica.

Using SQL Server Management Studio

To monitor availability groups, availability replicas, and availability databases

1. On the View menu, click **Object Explorer Details**, or press the **F7** key.
2. In Object Explorer, connect to the instance of SQL Server on which you want to monitor an availability group, and click the server name to expand the server tree.
3. Expand the **AlwaysOn High Availability** node and the **Availability Groups** node.
4. The **Object Explorer Details** pane displays every availability group for which the connected server instance hosts a replica. For each availability group, the **Server Instance (Primary)**

column displays the name of the server instance that is currently hosting the primary replica. To display more information about a given availability group, select it in Object Explorer.

5. The **Object Explorer Details** pane then displays the **Availability Replicas** and **Availability Databases** nodes for this availability group:
   - When you expand the **Availability Group** node in Object Explorer and select the **Availability Replicas** node, the **Object Explorer Details** pane displays information about each of the availability replicas in the group. For more information, see Availability Replica Details, later in this topic.

     To perform operations on multiple availability replicas, select them, and right-click them to open up a context menu that lists the available commands.
   - When you expand the **Availability Group** node in Object Explorer and select the **Availability Databases** node, the **Object Explorer Details** pane displays information about each of the availability databases in the group. For more information, see Availability Database Details, later in this topic.

     To perform operations on multiple availability databases, select them, and right-click them to open up a context menu that lists the available commands.

Availability Groups Details

The **Availability Groups** details screen displays the following columns:

Name

Lists the **Availability Replicas**, **Availability Databases**, and **Availability Group** Listeners folders of the selected availability group.

Availability Replica Details

The **Availability Replica** details screen displays the following columns:

Server Instance

Displays the name of the server instance that hosts the availability replica, along with an icon that indicates the current connection state of the server instance to the local server instance.

Role

Indicates the current role of the availability replica, either **Primary** or **Secondary**. For information about AlwaysOn Availability Groups roles, see "HADR" Overview (SQL Server).

Connection Mode in Secondary Role

Indicates whether the databases of a given availability replica that is performing the secondary role (that is, is acting as a secondary replica) can accept connections from clients.

The possible values are as follows:

Value	Description
Disallow Connections	No direct connections are allowed to the availability databases when this availability replica is acting as a secondary replica. Secondary databases are not available for read access.
Allow Only Read Intent Connections	Only direct read-only connections are allowed when this replica acting as a secondary replica. All database(s) in the replica are available for read access.
Allow All Connections	All connections are allowed to these databases for read-only access when this replica acting as a secondary replica.

Connection State

Indicates whether a secondary replica is currently connected to the primary replica. The possible values are as follows:

Value	Description
Disconnected	For a remote availability replica, indicates that it is disconnected from the local availability replica. The response of the local replica to the Disconnected state depends on its role, as follows: • On the primary replica, if a secondary replica is disconnected, the secondary databases are marked as **Not Synchronized** on the primary replica, and the primary replica waits for the secondary to reconnect. • On the secondary replica, upon detecting that it is disconnected, the secondary replica attempts to reconnect to the primary replica.
Connected	A remote availability replica that is currently connected to the local replica.
NULL	If the local replica is a secondary replica, this value is NULL for other secondary

	replicas.

Synchronization State

Indicates whether a secondary replica is currently synchronized with primary replica. The possible values are as follows:

Value	Description
Not Synchronized	The database is not synchronized or has not yet been joined to the availability group.
Synchronized	The database is synchronized with the primary database on the current primary replica, if any, or on the last primary replica. **Note** In performance mode, the database is never in the Synchronized state.
NULL	Unknown state. This value occurs when the local server instance cannot communicate with the WSFC failover cluster (that is the local node is not part of WSFC quorum).

Note

For information about performance counters for availability replicas, see SQL Server, HADR Availability Replica.

Availability Database Details

The **Availability Database** details screen displays the following properties of the availability databases in a given availability group:

Name

The name of the availability database.

Synchronization State

Indicates whether the availability database is currently synchronized with primary replica.

The possible synchronization states are as follows:

Value	Description
Synchronizing	The secondary database has received the transaction log records for the primary

	database that are not yet written to disk (hardened). **nNote** In asynchronous-commit mode, the synchronization state is always **Synchronizing**.

Suspended

Indicates whether the availability database is currently online. The possible values are as follows:

Value	Description
Suspended	This state indicates that the database is suspended locally and needs to be manually resumed. On the primary replica, the value is unreliable for a secondary database. To reliably determine whether a secondary database is suspended, query it on the secondary replica that hosts the database.
Not Joined	Indicates that the secondary database either has not been joined to the availability group or has been removed from the group.
Online	Indicates that the database is not suspended on the local availability replica and that the database is connected.
Not Connected	Indicates that the secondary replica is currently unable to connect to the primary replica.

Note

For information about performance counters for availability databases, see SQL Server, HADR Database Replica.

See Also

sys.dm_os_performance_counters (Transact-SQL)

Use the Availability Group Dashboard (SQL Server Management Studio)

View Availability Group Properties (SQL Server)

View Availability Replica Properties (SQL Server)

View Availability Group Properties

This topic describes how to view the properties of an availability group for an AlwaysOn availability group by using SQL Server Management Studio or Transact-SQL in SQL Server 2012.

- **To view availability group properties, using:**

  SQL Server Management Studio

  Transact-SQL

Using SQL Server Management Studio

To view and change the properties an availability group

1. In Object Explorer, connect to the server instance that hosts the primary replica, and expand the server tree.
2. Expand the **AlwaysOn High Availability** node and the **Availability Groups** node.
3. Right-click the availability group whose properties you want to view, and select the **Properties** command.
4. In the **Availability Group Properties** dialog box, use the **General** and **Backup Preferences** pages to view and, in some cases, change properties of the selected availability group. For more information, see Availability Group Properties (General Page) and Availability Group Properties (Backup Preferences Page).

   Use the **Permissions** page to view the current logins, roles, and explicit permissions associated with the availability group. For more information, see Permissions or Securables Page.

Using Transact-SQL

To view the properties and state of an availability group

To query the properties and states of the availability groups for which the server instance hosts an availability replica, use the following views:

sys.availability_groups

Returns a row for each availability group for which the local instance of SQL Server hosts an availability replica. Each row contains a cached copy of the availability group metadata.

Column names: group_id, name, resource_id, resource_group_id, failure_condition_level, health_check_timeout, automated_backup_preference, automated_backup_preference_desc

sys.availability_groups_cluster

Returns a row for each availability group in the WSFC cluster. Each row contains the availability group metadata from the Windows Server Failover Clustering (WSFC) cluster.

Column names: group_id, name, resource_id, resource_group_id, failure_condition_level, health_check_timeout, automated_backup_preference, automated_backup_preference_desc

sys.dm_hadr_availability_group_states

Returns a row for each availability group that possesses an availability replica on the local instance of SQL Server. Each row displays the states that define the health of a given availability group.

Column names: group_id, primary_replica, primary_recovery_health, primary_recovery_health_desc, secondary_recovery_health, secondary_recovery_health_desc, synchronization_health, synchronization_health_desc

Related Tasks

To view information about availability groups

- View and Change Availability Replica Properties
- View Availability Group Listener Properties (SQL Server)
- Use Policy-Based Management to View the Health of an Availability Group (SQL Server)
- Use the AlwaysOn Group Dashboard (SQL Server Management Studio)
- Monitor Availability Groups (Transact-SQL)

To configure an existing availability group

- Add a Secondary Replica to an Availability Group (SQL Server)
- Remove a Secondary Replica from an Availability Group (SQL Server)
- Add a Database to an Availability Group (SQL Server)
- Remove a Database from a Secondary Replica (AlwaysOn Availability Groups)
- Create or Configure an Availability Group Listener (SQL Server)
- Remove an Availability Group Listener (SQL Server)
- Remove a Database from an Availability Group (AlwaysOn Availability Groups)
- Delete an Availability Group (SQL Server)

To manually fail over an availability group

- Manually Fail Over an Availability Group (SQL Server)
- Force Failover of an Availability Group (SQL Server)

See Also

Overview of AlwaysOn Availability Groups

Monitor Availability Groups (Transact-SQL)

Policy-Based Management of Operational Issues with AlwaysOn Availability Groups (SQL Server)

Availability Group Properties/New Availability Group (General Page)

This topic applies to the **General** tab of both the **New Availability Group** dialog box and the **Availability Group Properties** dialog box. The **New Availability Group** dialog box enables you to create a new availability group without using the New Availability Group Wizard. The **Availability Group Properties** dialog box enables you to view and alter the configuration of an existing availability group.

To view availability group properties

- Viewing Availability Group Properties (SQL Server)
- Using the Availability Group Dashboard (SQL Server Management Studio)

UI Element List

Availability group name

Name of the availability group. This is a user-specified name that must be unique within the Windows Server Failover Cluster (WSFC).

Availability Databases

Database Name

Name of a database that has been added to the availability group.

Add

Click to add a database to the availability group.

Remove

Click to remove a selected database from the availability group.

Availability Replicas

Server instance

Server name of the instance of SQL Server that is hosting this replica and, for a non-default instance, its instance name.

Role

Primary

Currently the primary replica.

Secondary

Currently a secondary replica.

Resolving

Currently the replica role is in the process of being resolved to either the primary or secondary role.

Availability Mode

The availability mode of the replica, one of:

Asynchronous commit

The primary replica can commit transactions without waiting for the secondary to write the log to disk.

Synchronous commit

The primary replica waits to commit a given transaction until the secondary replica has written the transaction to disk.

For more information, see Availability Modes (AlwaysOn Availability Group).

Failover Mode

The failover mode of the replica, one of:

Automatic

Automatic failover. The replica is a target for automatic failovers. This is supported only if the availability mode is set to synchronous commit.

Manual

Manual failover. The replica can only be failed over to manually by the database administrator.

Connections in Primary Role

The type of client connections supported when the replica owns the primary role.

Allow all connections

All connections are allowed to the databases in the primary replica. This is the default setting.

Allow read/write connections

Connections where the Application Intent connection property is set to **ReadOnly** are disallowed. When the Application Intent property is set to **ReadWrite** or the Application Intent connection property is not set, the connection is allowed. For more information about Application Intent connection property, see Using Connection String Keywords with SQL Server Native Client.

Readable Secondary

Whether an availability replica that is performing the secondary role (that is, a secondary replica) can accept connections from clients, one of:

No

No direct connections are allowed to secondary databases of this replica. They are not available for read access. This is the default setting.

Read-intent only

Only direct read-only connections are allowed to secondary databases of this replica. The secondary database(s) are all available for read access.

Yes

All connections are allowed to secondary databases of this replica, but only for read access.

The secondary database(s) are all available for read access.

Session Timeout (seconds)

The number of seconds for the session-timeout period on this replica.

Endpoint URL

The URL of the endpoint. For information the format of these URLs, see Specify the Endpoint URL When Adding or Modifying an Availability Replica (SQL Server).

Add

Click to add a secondary replica to the availability group.

Remove

Click to remove a secondary replica from the availability group.

See Also

Overview (AlwaysOn Availability Group)

Availability Group Properties/New Availability Group (Backup Preferences Page)

Use this dialog box to view and change the backup preferences of the selected availability group.

To view availability group properties

- View Availability Group Properties (SQL Server)
- Use the Availability Group Dashboard (SQL Server Management Studio)

Where should backups occur?

Prefer Secondary

Specifies that backups should occur on a secondary replica except when the primary replica is the only replica online. In that case, the backup should occur on the primary replica. This is the default option.

Secondary only

Specifies that backups should never be performed on the primary replica. If the primary replica is the only replica online, the backup should not occur.

Primary

Specifies that the backups should always occur on the primary replica. This option is useful if you need backup features, such as creating differential backups, that are not supported when backup is run on a secondary replica.

Any Replica

Specifies that you prefer that backup jobs ignore the role of the availability replicas when choosing the replica to perform backups. Note backup jobs might evaluate other factors such as backup priority of each availability replica in combination with its operational state and connected state.

Important

There is no enforcement of the backup-preference setting. The interpretation of this preference depends on the logic, if any, that you script into back jobs for the databases in a given availability group. For more information, see Backup on Secondary Replicas (AlwaysOn Availability Groups).

Replica backup priorities

This grid displays the current backup priority of each server instance that hosts a replica for the availability group. Use this grid to change the backup priority of one or more availability replicas.

Server Instance

The name of the instance of SQL Server that hosts the availability replica.

Backup Priority (Lowest=1, Highest=100)

Specifies your priority for performing backups on this replica relative to the other replicas in the same availability group. The value is an integer in the range of 0..100. 1 indicates the lowest priority, and 100 indicates the highest priority. If **Backup Priority** = 1, the availability replica would be chosen for performing backups only if no higher priority availability replicas are currently available.

Exclude Replica

Select if you never want this availability replica to be chosen for performing backups. This is useful, for example, for a remote availability replica to which you never want backups to fail over.

See Also

Backup on Secondary Replicas (AlwaysOn Availability Groups)

ALTER AVAILABILITY GROUP

View Availability Replica Properties

This topic describes how to view the properties of an availability replica for an AlwaysOn availability group by using SQL Server Management Studio or Transact-SQL in SQL Server 2012.

- **To view availability replica properties, using:**

  SQL Server Management Studio

  Transact-SQL

Using SQL Server Management Studio

To view and change properties an availability replica

1. In Object Explorer, connect to the server instance that hosts the primary replica, and expand the server tree.

2. Expand the **AlwaysOn High Availability** node and the **Availability Groups** node.
3. Expand the availability group to which the availability replica belongs, and expand the **Availability Replicas** node.
4. Right-click the availability replica whose properties you want to view, and select the **Properties** command.
5. In the **Availability Replica Properties** dialog box, use the **General** page to view the properties of this replica. If you are connected to the primary replica, you can change the following properties: availability mode, failover mode, connection access for the primary role, read-access for the secondary role (readable-secondary), and the session-timeout value. For more information, see Availability Replica Properties (General Page).

Using Transact-SQL

To view properties and states of availability replicas

To view the properties and states of availability replicas, use the following views and system function:

sys.availability_replicas

Returns a row for every availability replica in each availability group for which the local instance of SQL Server hosts an availability replica.

Column names: replica_id, group_id, replica_metadata_id, replica_server_name, owner_sid, endpoint_url, availability_mode, availability_mode_desc, failover_mode, failover_mode_desc, session_timeout, primary_role_allow_connections, primary_role_allow_connections_desc, secondary_role_allow_connections, secondary_role_allow_connections_desc, create_date, modify_date, backup_priority, read_only_routing_url

sys.availability_read_only_routing_lists

Returns a row for the read only routing list of each availability replica in an AlwaysOn availability group in the WSFC failover cluster.

Column names: replica_id, routing_priority, read_only_replica_id

sys.dm_hadr_availability_replica_cluster_nodes

Returns a row for every availability replica (regardless of join state) of the AlwaysOn availability groups in the Windows Server Failover Clustering (WSFC) cluster.

Column names: group_name, replica_server_name, node_name

sys.dm_hadr_availability_replica_cluster_states

Returns a row for each replica (regardless of join state) of all AlwaysOn availability groups (regardless of replica location) in the Windows Server Failover Clustering (WSFC) cluster.

Column names: replica_id, replica_server_name, group_id, join_state, join_state_desc

sys.dm_hadr_availability_replica_states

Returns a row showing the state of each local availability replica and a row for each remote availability replica in the same availability group.

Column names: replica_id, group_id, is_local, role, role_desc, operational_state, operational_state_desc, connected_state, connected_state_desc, recovery_health, recovery_health_desc, synchronization_health, synchronization_health_desc, last_connect_error_number, last_connect_error_description, and last_connect_error_timestamp

sys.fn_hadr_backup_is_preferred_replica

Determines whether the current replica is the preferred backup replica. Returns 1 if the database on the current server instance is the preferred replica. Otherwise, it returns 0.

Note

For information about performance counters for availability replicas (the **SQLServer:Availability Replica** performance object), see SQL Server, HADR Availability Replica.

Related Tasks

To view information about availability groups

- View Availability Group Properties
- View Availability Group Listener Properties (SQL Server)
- Use Policy-Based Management to View the Health of an Availability Group (SQL Server)
- Use the AlwaysOn Group Dashboard (SQL Server Management Studio)
- Monitor Availability Groups (Transact-SQL)

To manage availability replicas

- Add a Secondary Replica to an Availability Group (SQL Server)
- Join a Secondary Replica to an Availability Group (SQL Server)
- Configure Read-Only Access on a Secondary Availability Replica (SQL Server)
- Set the Availability Mode of an Availability Replica (SQL Server)
- Set the Failover Mode of an Availability Replica (SQL Server)
- Set the Session-Timeout Period for an Availability Replica (SQL Server)
- Remove a Secondary Replica from an Availability Group (SQL Server)

To manage an availability database

- Add a Database to an Availability Group (SQL Server)
- Join a Secondary Database to an Availability Group (SQL Server)
- Suspend a Database on an Secondary Replica Location (SQL Server)
- Resume a Secondary Database on an Secondary Replica (SQL Server)
- Remove a Secondary Database from an Availability Group (SQL Server)
- Remove a Primary Database from an Availability Group (SQL Server)

See Also

[Overview of AlwaysOn Availability Groups]

[Monitor Availability Groups (Transact-SQL)]

[Policy-Based Management of Operational Issues with AlwaysOn Availability Groups (SQL Server)]

[Administration of an Availability Group (SQL Server)]

Availability Replica Properties (General Page)

Use this dialog box to view the properties of an availability replica.

Task List

To view availability replica properties

- [View Availability Replica Properties (SQL Server)]
- [Use the Availability Group Dashboard (SQL Server Management Studio)]

UI Element List

Availability group name

Name of the availability group. This is a user-specified name that must be unique within the Windows Server Failover Cluster (WSFC).

Server instance

Server name of the instance of SQL Server that is hosting this replica and, for a non-default instance, its instance name.

Role

Primary

Currently the primary replica.

Secondary

Currently a secondary replica.

Resolving

Currently the replica role is in the process of being resolved to either the primary or secondary role.

Availability mode

The availability mode of the replica, one of:

Asynchronous commit

The primary replica can commit transactions without waiting for the secondary to write the log to disk.

Synchronous commit

The primary replica waits to commit a given transaction until the secondary replica has written the transaction to disk.

For more information, see Availability Modes (AlwaysOn Availability Group).

Failover mode

The failover mode of the replica, one of:

Automatic

Automatic failover. The replica is a target for automatic failovers. This is supported only if the availability mode is set to synchronous commit.

Manual

Manual failover. The replica can only be failed over to manually by the database administrator.

Connections in primary role

The type of client connections supported when the replica owns the primary role.

Allow all connections

All connections are allowed to the databases in the primary replica. This is the default setting.

Allow read/write connections

Connections where the Application Intent connection property is set to **ReadOnly** are disallowed. When the Application Intent property is set to **ReadWrite** or the application intent connection property is not set, the connection is allowed.

Readable Secondary

Whether an availability replica that is performing the secondary role (that is, a secondary replica) can accept connections from clients, one of:

No

No direct connections are allowed to secondary databases of this replica. They are not available for read access. This is the default setting.

Read-intent only

Only direct read-only connections are allowed to secondary databases of this replica. The secondary database(s) are all available for read access.

Yes

All connections are allowed to secondary databases of this replica, but only for read access. The secondary database(s) are all available for read access.

For more information, see Read-Only Access to Secondary replicas (AlwaysOn Availability Group).

Session timeout (seconds)

The time-out period, in seconds. The time-out period is the maximum time that the replica waits to receive a message from another replica before considering connection between the primary and secondary replica have failed. Session timeout detects whether secondaries are connected the primary replica. On detecting a failed connection with a secondary replica, the primary replica considers the secondary replica to be NOT_SYNCHRONIZED. On detecting a failed connection with the primary replica, a secondary replica simply attempts to reconnect.

Note

Session timeouts do not cause automatic failovers.

Endpoint URL

String representation of the user-specified database mirroring endpoint that is used by connections between primary and secondary replicas for data synchronization. For information about the syntax of endpoint URLs, see Specifying the Endpoint URL When Adding or Modifying an Availability Replica (AlwaysOn Availability Group).

See Also

Overview (AlwaysOn Availability Group)

View Availability Group Listener Properties

This topic describes how to view the properties of an AlwaysOn *availability group listener* by using SQL Server Management Studio or Transact-SQL in SQL Server 2012.

- **To view listener properties, using:**

  SQL Server Management Studio

  Transact-SQL

Using SQL Server Management Studio

To view listener properties

1. In Object Explorer, connect to a server instance that hosts any availability replica of the availability group whose listener you want to view. Click the server name to expand the server tree.
2. Expand the **AlwaysOn High Availability** node and the **Availability Groups** node.
3. Expand the node of the availability group, and expand the **Availability Groups Listeners** node.
4. Right-click the listener that you want to view, and select the **Properties** command.
5. This opens the **Availability Group Listener Properties** dialog box. For more information, see Availability Group Listener Properties (Dialog Box), later in this topic.

Availability Group Listener Properties (Dialog Box)

Listener DNS Name

The network name of the availability group listener.

Port

The TPC port used by this listener.

Note

If you are connected the primary replica, you can use this field to modify the port number of the listener. This requires ALTER AVAILABILITY GROUP permission on the availability group, CONTROL AVAILABILITY GROUP permission, ALTER ANY AVAILABILITY GROUP permission, or CONTROL SERVER permission.

Network Mode

Indicates the TCP protocol used by the listener, one of:

DHCP

The listener uses an dynamic IP address that is assigned by a server running the Dynamic Host Configuration Protocol (DHCP).

Static IP

The listener uses one or more Static IP addresses. To access the different subnets, an availability group listener must use static IP addresses.

The grid displays each of the subnets on which the listener listens and the IP address associated with that subnet.

Using Transact-SQL

To view listener properties

To monitor the availability group listeners, use the following views:

sys.availability_group_listener_ip_addresses

Returns a row for every conformant virtual IP address that is currently online for an availability group listener.

Column names: listener_id, ip_address, ip_subnet_mask, is_dhcp, network_subnet_ip, network_subnet_prefix_length, network_subnet_ipv4_mask, state, state_desc

sys.availability_group_listeners

For a given availability group, returns either zero rows indicating that no network name is associated with the availability group, or returns a row for each availability-group listener configuration in the WSFC cluster.

Column names: group_id, listener_id, dns_name, port, is_conformant, ip_configuration_string_from_cluster

sys.dm_tcp_listener_states

Returns a row containing dynamic-state information for each TCP listener.

Column names: listener_id, ip_address, is_ipv4, port, type, type_desc, state, state_desc, start_time

Note

For more information about using Transact-SQL to monitor your AlwaysOn Availability Groups environment, see Monitor Availability Groups (Transact-SQL).

Related Tasks

- Create or Configure an Availability Group Listener (SQL Server)
- Remove an Availability Group Listener (SQL Server)

See Also

AlwaysOn Availability Groups (SQL Server)

Availability Group Listeners, Client Connectivity, and Application Failover (AlwaysOn Availability Groups)

Monitor Availability Groups (Transact-SQL)

AlwaysOn Availability Groups: Interoperability

This topic documents interoperability of AlwaysOn Availability Groups with other SQL Server features in SQL Server 2012.

In This Topic:

- Features That Interoperate with AlwaysOn Availability Groups
- Features that Do Not Interoperate with AlwaysOn Availability Groups

Features That Interoperate with AlwaysOn Availability Groups

The following table lists SQL Server features that interoperate with AlwaysOn Availability Groups in SQL Server 2012. A link in the **More Information** column indicates that interoperability considerations exist for a given feature.

Feature	More Information
Change data capture	Replication, Change Tracking, Change Data Capture, and AlwaysOn Availability Groups (SQL Server)
Change tracking	Replication, Change Tracking, Change Data Capture, and AlwaysOn Availability Groups (SQL Server)
Contained databases	Contained Databases with AlwaysOn Availability Groups (SQL Server)
Database encryption	Encrypted Databases with AlwaysOn

Feature	More Information
	Availability Groups (SQL Server)
Database snapshots	Database Snapshots with AlwaysOn Availability Groups (SQL Server)
FILESTREAM and FileTable	FILESTREAM and FileTable with AlwaysOn Availability Groups (SQL Server)
Full-text search	**Note** Full-Text indexes are synchronized with AlwaysOn secondary databases.
Log shipping	Log Shipping and AlwaysOn Availability Groups (SQL Server)
Remote Blob Store (RBS)	
Replication	• Configure Replication for AlwaysOn Availability Groups • Maintaining an AlwaysOn Publication Database • Replication, Change Tracking, Change Data Capture, and AlwaysOn Availability Groups (SQL Server) • Replication Subscribers and AlwaysOn
Analysis Services	Analysis Services with AlwaysOn Availability Groups
Reporting Services	Utilize read only secondary replicas as a reporting data source and reduce the load on your primary read-write replica. Reporting Services with AlwaysOn Availability Groups (SQL Server)
Service Broker	Service Broker with AlwaysOn Availability Groups (SQL Server) 881c20e5-1c99-44eb-b393-09fc5ea0f122
SQL Server Agent	

Features that Do Not Interoperate with AlwaysOn Availability Groups

AlwaysOn Availability Groups does not interoperate with the following features:

- Cross-database transactions/distributed transactions

  For information about why such transactions are not supported, see Cross-Database Transactions Not Supported For Database Mirroring or AlwaysOn Availability Groups (SQL Server).

- Database mirroring

See Also

Overview of AlwaysOn Availability Groups

Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups (SQL Server)

Contained Databases with AlwaysOn Availability Groups

This topic contains information about the using a contained database with AlwaysOn Availability Groups in SQL Server 2012.

In this Topic:

- Prerequisites
- Related Tasks

Prerequisites

- Before adding a contained database to an availability group, ensure that the **contained database authentication** server option is set to **1** on every server instance that hosts an availability replica for the availability group. For more information, see contained database authentication Option and Set Server Configuration Options.

Related Tasks

- Set Server Configuration Options

See Also

Overview of AlwaysOn Availability Groups

Contained Databases

Cross-Database Transactions Not Supported For Database Mirroring or AlwaysOn Availability Groups

Cross-database transactions and distributed transactions are not supported by AlwaysOn Availability Groups or by database mirroring. This is because transaction atomicity/integrity cannot be guaranteed for the following reasons:

- For cross-database transactions: Each database commits independently. Therefore, even for databases in a single availability group, a failover could occur after one database commits a transaction but before the other database does. For database mirroring this issue is compounded because after a failover, the mirrored database is typically on a different server instance from the other database, and even if both databases are mirrored between the same two partners, there is no guarantee that both databases will fail over at the same time.

- For distributed transactions: After a failover, the new principal server/primary replica is unable to connect to the distributed transaction coordinator on the previous principal server/primary replica. Therefore, the new principal server/primary replica cannot obtain the transaction status.

The following database mirroring example illustrates how a logical inconsistency could occur. In this example, an application uses a cross-database transaction to insert two rows of data: one row is inserted into a table in a mirrored database, A, and the other row is inserted into a table in another database, B. Database A is being mirrored in high-safety mode with automatic failover. While the transaction is being committed, database A becomes unavailable, and the mirroring session automatically fails over to the mirror of database A.

After the failover, the cross-database transaction might be successfully committed on database B but not on the failed-over database. This would occur if the original principal server for database A had not sent the log for the cross-database transaction to the mirror server before the failure. After the failover, that transaction would not exist on the new principal server. Databases A and B would become inconsistent, because the data inserted in database B remains intact, but the data inserted in database A has been lost.

A similar scenario can occur while using a MS DTC transaction. For example, after failover, the new principal contacts MS DTC. But MS DTC has no knowledge of the new principal server, and it terminates any transactions that are "preparing to commit," which are considered committed in other databases.

Database Snapshots with AlwaysOn Availability Groups

You can create a database snapshot on an primary or secondary database in an availability group. The replica role must be either PRIMARY or SECONDARY, not in the RESOLVING state.

We recommend that the database synchronization state be SYNCHRONIZING or SYNCHRONIZED when you create a database snapshot. However, database snapshots can be created when the database synchronization state is NOT SYNCHRONIZING.

A database snapshot on a secondary replica should continue to work if the replica is DISCONNECTED from the primary replica.

Some AlwaysOn Availability Groups conditions cause both the source database and its database snapshots to be restarted, temporarily disconnecting users. These conditions are as follows:

- The primary replica changes roles, whether because the current primary replica goes off line and comes back online on the same server instance or because the availability group fails over.
- The database enters the secondary role.

If the availability replica that hosts database snapshots is failed over, the database snapshots remain on the server instance where they were created. Users can continue to use the snapshots after the failover.If performance is a concern in your environment, we recommend that you create database snapshots only on secondary databases that are hosted by a secondary replica that is configured for manual failover mode. If you ever manually fail over the availability group to this secondary replica, you can create a new set of database snapshots on another secondary replica, redirect clients to the new database snapshots, and drop all of the database snapshots from the now primary databases.

See Also

AlwaysOn Availability Groups (SQL Server)

Database Snapshots (SQL Server)

Encrypted Databases with AlwaysOn Availability Groups

This topic contains information about the using currently encrypted or recently decrypted databases with AlwaysOn Availability Groups in SQL Server 2012.

In this Topic:

- Limitations and Restrictions
- Related Tasks

Limitations and Restrictions

- If a database is encrypted or even contains a Database Encryption Key (DEK), you cannot use the New Availability Group Wizard or Add Database to Availability Group Wizard to add the database to an availability group. Even if an encrypted database has been decrypted, its log backups might contain encrypted data. In this case, full initial data synchronization could fail on the database. This is because the restore log operation might require the certificate that was used by the database encryption keys (DEKs), and that certificate might be unavailable.

  To make a decrypted database eligible to add to an availability group using the wizard:

  a. Create a log backup of the primary database.

  b. Create a full database backup of the primary database.

  c. Restore the database backup on the server instance that hosts the secondary replica.

  d. Create a new log backup from primary database.

  e. Restore this log backup on the secondary database.

Related Tasks

- Manually Prepare a Secondary Database for an Availability Group (SQL Server)
- Use the New Availability Group Wizard (SQL Server Management Studio)
- Use the Add Database to Availability Group Wizard (SQL Server Management Studio)

See Also

Overview of AlwaysOn Availability Groups (SQL Server)

Transparent Data Encryption (TDE)

FILESTREAM and FileTable with AlwaysOn Availability Groups

This topic contains information about the using the FILESTREAM and FileTable features with AlwaysOn Availability Groups in SQL Server 2012.

In this Topic:

- Prerequisites
- Using Virtual Network Names (VNNs) for FILESTREAM and FileTable Access
- Related Tasks
- Related Content

Prerequisites

- Before adding a database that uses FILESTREAM, with or without FileTable, to an availability group, ensure that FILESTREAM is enabled on every server instance that hosts an availability replica for the availability group. For more information, see Enable and Configure FILESTREAM.

Using Virtual Network Names (VNNs) for FILESTREAM and FileTable Access

When you enable FILESTREAM on an instance of SQL Server, an instance-level share is created to provide access to the FILESTREAM data. You access this share by using the computer name in the following format:

```
\\<computer_name>\<filestream_share_name>
```

In an AlwaysOn availability group, however, the name of the computer is virtualized by using a Virtual Network Name, or VNN. When the computer is the primary replica in an availability group, and databases in the availability group contain FILESTREAM data, then a VNN-scoped share is also created to provide access to the FILESTREAM data. This does not affect Transact-SQL access to FILESTREAM data. However applications that use file system APIs have to use the VNN-scoped share, which has a path in the following format:

```
\\<VNN>\<filestream_share_name>
```

This VNN-scoped share is created when one of the following events occurs.

- You add a database that contains FILESTREAM data to an AlwaysOn availability group on the primary replica. In this case, the share `\\<computer_name>\<filestream_share_name>` already exists. The share `\\<VNN>\<filestream_share_name>` is created.
- You enable FILESTREAM for file i/o streaming access on a primary replica that has availability groups. The following shares are created:
    a. `\\<computer_name>\<filestream_share_name>`
    b. `\\<VNN1>\<filestream_share_name>` for availability group 1.
    c. `\\<VNN2>\<filestream_share_name>` for availability group 2.

These VNN-scoped shares are also propagated to all secondary replicas.

When the database that contains FILESTREAM or FileTable data belongs to an AlwaysOn availability group:

- The FILESTREAM and FileTable functions accept or return virtual network names (VNNs) instead of computer names. For more information about these functions, see Filestream and FileTable Functions (Transact-SQL).
- All access to FILESTREAM or FileTable data through the file system APIs should use VNNs instead of computer names. For more information, see FILESTREAM and FileTable with AlwaysOn Availability Groups (SQL Server).

If your application tries to access the share by using the computer name in the format `\\<computer_name>\<filestream_share_name>` when the database is part of an availability group, then an error is raised.

If your application tries to access the share by using a VNN-scoped path when the database is not part of an availability group, then the request may succeed. In this case, the virtual network name is resolved to the computer name. However this usage is strongly discouraged, since the VNN-scoped path will stop working if the availability group is dropped.

Related Tasks

- Enable and Configure FILESTREAM
- Enable the Prerequisites for FileTable

Related Content

None.

See Also

Overview of AlwaysOn Availability Groups

Prerequisites for Migrating from Log Shipping to AlwaysOn Availability Groups

This topic describes the prerequisites for converting a log shipping primary database along with one or more of its secondary databases to an AlwaysOn primary database and secondary database(s).

In This Topic:

- Availability Group Prerequisites
- Log Shipping Prerequisites
- Related Tasks
- Related Content

Availability Group Prerequisites

To allow backup jobs to run on the primary replica of the availability group, use the following AlwaysOn Availability Groups backup settings:

Property	Setting
Automated backup preference of availability group	Only on the primary replica
Backup priority of the primary replica.	>0

For more information:

View Availability Group Properties (SQL Server)

Configure Backup on Availability Replicas (SQL Server)

Log Shipping Prerequisites

- The log shipping primary database must reside on the instance of SQL Server that hosts the initial/current primary replica of the availability group.
- For a given log shipping secondary database to be converted to an AlwaysOn secondary database, it must:
    - Use the same name as the primary database.
    - Reside on a server instance that hosts a secondary replica for the availability group.

Once the backup job has run on the primary database, disable the backup job, and once the restore job has run on a given secondary database, disable the restore job.

Note
After you have created all the secondary databases for the availability group, if you want to perform backups on secondary replicas, you need to re-configure the automated backup preference of the availability group.

For more information:

Converting a logshipping configuration to Availability Group (a SQL Server blog)

Related Tasks

Log shipping

- Upgrade Log Shipping to SQL Server 2012 (Transact-SQL)
- Remove Log Shipping

AlwaysOn Availability Groups

- Use the New Availability Group Wizard (SQL Server Management Studio)
- Use the New Availability Group Dialog Box (SQL Server Management Studio)
- Create an Availability Group (Transact-SQL)
- Create and Configure an Availability Group (SQL Server PowerShell)

- Join a Secondary Database to an Availability Group (SQL Server)
- Configure Backup on Availability Replicas (SQL Server)

Related Content

- Converting a logshipping configuration to Availability Group
- Add a Log Shipping Primary Database and Secondary Database(s) to an Existing Availability Group
- Microsoft SQL Server AlwaysOn Solutions Guide for High Availability and Disaster Recovery
- SQL Server AlwaysOn Team Blog: The official SQL Server AlwaysOn Team Blog

See Also

Log Shipping Overview

Overview of AlwaysOn Availability Groups

Monitoring of Availability Groups (SQL Server)

Configure Replication for AlwaysOn Availability Groups

Configuring replication and AlwaysOn availability groups involves seven steps. Each step is described in more detail in the following sections.

1. Configure the database publications and subscriptions.
2. Configure the AlwaysOn availability group.
3. Insure that all secondary replica hosts are configured for replication.
4. Configure the secondary replica hosts as replication publishers.
5. Redirect the original publisher to the Availability Group Listener Name.
6. Run the validation stored procedure to verify the configuration.
7. Add the original publisher to Replication Monitor.

Steps 1 and 2 can be performed in either order.

1. Configure the Database Publications and Subscriptions

Configure the distributor

The distributor should not be a host for any of the current (or intended) replicas of the availability group that the publishing database is (or will become) a member of.

1. Configure distribution at the distributor. If stored procedures are being used for configuration, run **sp_adddistributor**. Use the @password parameter to identify the password that will be used when a remote publisher connects to the distributor. The password will also be needed at each remote publisher when the remote distributor is set up.

   ```
   USE master;

   GO

   EXEC sys.sp_adddistributor
   ```

```
        @distributor = 'MyDistributor',
        @password = '**Strong password for distributor**';
```

2. Create the distribution database at the distributor. If stored procedures are being used for configuration, run **sp_adddistributiondb**.

```
    USE master;
    GO
    EXEC sys.sp_adddistributiondb
        @database = 'distribution',
        @security_mode = 1;
```

3. Configure the remote publisher. If stored procedures are being used to configure the distributor, run **sp_adddistpublisher**. The @security_mode parameter is used to determine how the publisher validation stored procedure that is run from the replication agents, connects to the current primary. If set to 1 Windows authentication is used to connect to the current primary. If set to 0, SQL Server authentication is used with the specified @login and @password values. The login and password specified must be valid at each secondary replica for the validation stored procedure to successfully connect to that replica.

   Note
   If any modified replication agents run on a computer other than the distributor, use of Windows authentication for the connection to the primary will require Kerberos authentication to be configured for the communication between the replica host computers. Use of a SQL Server login for the connection to the current primary will not require Kerberos authentication.

```
    USE master;
    GO
    EXEC sys.sp_adddistpublisher
        @publisher = 'AGPrimaryReplicaHost',
        @distribution_db = 'distribution',
        @working_directory = '\\MyReplShare\WorkingDir',
        @login = 'MyPubLogin',
        @password = '**Strong password for publisher**';
```

For more information, see sp_adddistpublisher (Transact-SQL).

Configure the publisher at the original publisher

1. Configure remote distribution. If stored procedures are being used to configure the publisher, run **sp_adddistributor**. Specify the same value for @password as that used when **sp_adddistrbutor** was run at the distributor to set up distribution.

```
    exec sys.sp_adddistributor
```

```
        @distributor = 'MyDistributor',
        @password = 'MyDistPass'
```

2. Enable the database for replication. If stored procedures are being used to configure the publisher, run **sp_replicationdboption**. If both transactional and merge replication are to be configured for the database, each must be enabled.

```
    USE master;
    GO
    EXEC sys.sp_replicationdboption
        @dbname = 'MyDBName',
        @optname = 'publish',
        @value = 'true';

    EXEC sys.sp_replicationdboption
        @dbname = 'MyDBName',
        @optname = 'merge publish',
        @value = 'true';
```

3. Create the replication publication, articles, and subscriptions. For more information about how to configure replication, see Publishing Data and Database objects.

2. Configure the AlwaysOn Availability Group

At the intended primary, create the availability group with the published (or to be published) database as a member database. If using the Availability Group Wizard, you can either allow the wizard to initially synchronize the secondary replica databases or you can perform the initialization manually by using backup and restore.

Create a DNS listener for the availability group that will be used by the replication agents to connect to the current primary. The listener name that is specified will be used as the target of redirection for the original publisher/published database pair. For example, if you are using DDL to configure the availability group, the following code example can be used to specify an availability group listener for an existing availability group named `MyAG`:

```
ALTER AVAILABILITY GROUP 'MyAG'
    ADD LISTENER 'MyAGListenerName' (WITH IP (('10.120.19.155',
'255.255.254.0')));
```

For more information, see Creation and Configuration of Availability Groups (SQL Server).

3. Insure that all of the Secondary Replica Hosts are Configured for Replication

At each secondary replica host, verify that SQL Server has been configured to support replication. The following query can be run at each secondary replica host to determine whether replication is installed:

```
USE master;

GO

DECLARE @installed int;

EXEC @installed = sys.sp_MS_replication_installed;

SELECT @installed;
```

If @installed is 0, replication must be added to the SQL Server installation.

4. Configure the Secondary Replica Hosts as Replication Publishers

A secondary replica cannot act as a replication publisher or republisher but replication must be configured so that the secondary can take over after a failover. At the distributor, configure distribution for each secondary replica host. Specify the same distribution database and working directory as was specified when the original publisher was added to the distributor. If you are using stored procedures to configure distribution, use **sp_adddistpublisher** to associate the remote publishers with the distributor. If @login and @password were used for the original publisher, specify the same values for each when you add the secondary replica hosts as publishers.

```
EXEC sys.sp_adddistpublisher

    @publisher = 'AGSecondaryReplicaHost',

    @distribution_db = 'distribution',

    @working_directory = '\\MyReplShare\WorkingDir',

    @login = 'MyPubLogin',

    @password = '**Strong password for publisher**';
```

At each secondary replica host, configure distribution. Identify the distributor of the original publisher as the remote distributor. Use the same password as that used when **sp_adddistributor** was run originally at the distributor. If stored procedures are being used to configure distribution, the @password parameter of **sp_adddistributor** is used to specify the password.

```
EXEC sp_adddistributor

    @distributor = 'MyDistributor',

    @password = '**Strong password for distributor**';
```

At each secondary replica host, make sure that the push subscribers of the database publications appear as linked servers. If stored procedures are being used to configure the remote publishers, use **sp_addlinkedserver** to add the subscribers (if not already present) as linked servers to the publishers.

```
EXEC sys.sp_addlinkedserver

    @server = 'MySubscriber';
```

5. Redirect the Original Publisher to the AG Listener Name

At the distributor, in the distribution database, run the stored procedure **sp_redirect_publisher** to associate the original publisher and the published database with the availability group listener name of the availability group.

```
USE distribution;

GO

EXEC sys.sp_redirect_publisher

@original_publisher = 'MyPublisher',

    @publisher_db = 'MyPublishedDB',

    @redirected_publisher = 'MyAGListenerName';
```

6. Run the Replication Validation Stored Procedure to Verify the Configuration

At the distributor, in the distribution database, run the stored procedure **sp_validate_replica_hosts_as_publishers** to verify that all replica hosts are now configured to serve as publishers for the published database.

```
USE distribution;

GO

DECLARE @redirected_publisher sysname;

EXEC sys.sp_validate_replica_hosts_as_publishers

    @original_publisher = 'MyPublisher',

    @publisher_db = 'MyPublishedDB',

    @redirected_publisher = @redirected_publisher output;
```

The stored procedure **sp_validate_replica_hosts_as_publishers** should be run from a login with sufficient authorization at each availability group replica host to query for information about the availability group. Unlike **sp_validate_redirected_publisher**, it uses the credentials of the caller and does not use the login retained in msdb.dbo.MSdistpublishers to connect to the availability group replicas.

Note

- **sp_validate_replica_hosts_as_publishers** will fail with the following error when validating secondary replica hosts that do not allow read access, or require read intent to be specified.
- Msg 21899, Level 11, State 1, Procedure **sp_hadr_verify_subscribers_at_publisher**, Line 109

- The query at the redirected publisher 'MyReplicaHostName' to determine whether there were sysserver entries for the subscribers of the original publisher 'MyOriginalPublisher' failed with error '976', error message 'Error 976, Level 14, State 1, Message: The target database, 'MyPublishedDB', is participating in an availability group and is currently not accessible for queries. Either data movement is suspended or the availability replica is not enabled for read access. To allow read-only access to this and other databases in the availability group, enable read access to one or more secondary availability replicas in the group. For more information, see the **ALTER AVAILABILITY GROUP** statement in SQL Server Books Online.'.
- One or more publisher validation errors were encountered for replica host 'MyReplicaHostName'.

This is expected behavior. You must verify the presence of the subscriber server entries at these secondary replica hosts by querying for the sysserver entries directly at the host.

7. Add the Original Publisher to Replication Monitor

At each availability group replica, add the original publisher to Replication Monitor.

Related Tasks

Replication

- Maintaining An AlwaysOn Publication Database
- Replication, Change Tracking, Change Data Capture, and AlwaysOn Availability Groups
- Administration (Replication)

To create and configure an availability group

- Use the New Availability Group Wizard (SQL Server Management Studio)
- Use the New Availability Group Dialog Box (SQL Server Management Studio)
- Create an Availability Group (Transact-SQL)
- Create and Configure an Availability Group (SQL Server PowerShell)
- Specify the Endpoint URL When Adding or Modifying an Availability Replica (AlwaysOn Availability Groups)
- Create a Database Mirroring Endpoint for AlwaysOn Availability Groups (SQL Server PowerShell)
- Join a Secondary Replica to an Availability Group (SQL Server)
- Prepare a Secondary Database for an Availability Group (SQL Server)
- Join a Secondary Database to an Availability Group (SQL Server)
- Create or Configure an Availability Group Listener (SQL Server)

See Also

Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups (SQL Server)

Overview of AlwaysOn Availability Groups

AlwaysOn Availability Groups: Interoperability

SQL Server Replication

Maintaining an AlwaysOn Publication Database

This topic discusses special considerations for maintaining a publication database when you use AlwaysOn availability groups.

In This Topic:

- Maintaining a Published Database in an Availability Group
- Removing a Published Database from an Availability Group
- Related Tasks

Maintaining a Published Database in an Availability Group

Maintaining an AlwaysOn publication database is basically the same as maintaining a standard publication database, with the following considerations:

- Administration must occur at the primary replica host. In SQL Server Management Studio, publications appear under the **Local Publications** folder for the primary replica host and also for readable secondary replicas. After failover, you might have to manually refresh Management Studio for the change to be reflected if the secondary that was promoted to primary was not readable.
- Replication Monitor always displays publication information under the original publisher. However, this information can be viewed in Replication Monitor from any replica by adding the original publisher as a server.
- When using stored procedures or Replication Management Objects (RMO) to administer replication at the current primary, for cases in which you specify the Publisher name, you must specify the name of the instance on which the database was enabled for replication (the original publisher). To determine the appropriate name, use the **PUBLISHINGSERVERNAME** function. When a publishing database joins an availability group, the replication metadata stored in the secondary database replicas is identical to that at the primary. Consequently, for publication databases enabled for replication at the primary, the publisher instance name stored in system tables at the secondary is the name of the primary, not the secondary. This affects replication configuration and maintenance if the publication database fails over to a secondary. For example, if you are configuring replication with stored procedures at a secondary after failover, and you want a pull subscription to a publication database that was enabled at a different replica, you must specify the name of the original publisher instead of the current publisher as the @publisher parameter of **sp_addpullsubscription** or **sp_addmergepulllsubscription**. However, if you enable a publication database after failover, the publisher instance name stored in the

system tables is the name of the current primary host. In this case, you would use the host name of the current primary replica for the @publisher parameter.

Note
For some procedures, such as **sp_addpublication**, the @publisher parameter is supported only for publishers that are not instances of SQL Server; in these cases, it is not relevant for SQL Server AlwaysOn.

- To synchronize a subscription in Management Studio after a failover, synchronize pull subscriptions from the subscriber and synchronize push subscriptions from the active publisher.

Removing a Published Database from an Availability Group

Consider the following issues if a published database is removed from an availability group, or if an availability group that has a published member database is dropped.

- If the publication database at the original publisher is removed from an availability group primary replica, you must run **sp_redirect_publisher** without specifying a value for the @redirected_publisher parameter in order to remove the redirection for the publisher/database pair.

```
EXEC sys.sp_redirect_publisher
    @original_publisher = 'MyPublisher',
    @published_database = 'MyPublishedDB';
```

  The database will be left in the recovering state at the primary and must be restored. Once you do this, replication should work unchanged against the original Publisher.

- If the publication database fails over from the original publisher to a replica and the database is removed from the availability group primary replica, use the stored procedure **sp_redirect_publisher** to explicitly redirect the original publisher to the new publisher. The database will be left in the recovering state and must be restored. Once you do this, replication should continue to work as it did under the availability group.

```
EXEC sys.sp_redirect_publisher
    @original_publisher = 'MyPublisher',
    @published_database = 'MyPublishedDB',
    @redirected_publisher = 'MyNewPublisher';
```

  Do not remove the remote server for the original publisher from the distributor, even if the server can no longer be accessed. The server metadata for the original publisher is needed at the distributor to satisfy publication metadata queries.

- If a complete availability group is removed, the behavior with regard to a member replicated database is the same as when a published database is removed from an availability group. Replication can be resumed from the last primary as soon as the database has been restored and the redirection has been modified. If the database is restored at its original publisher, redirection should be removed. If the database is restored at a different host, redirection should be explicitly directed to the new host.

Note
When an availability group is removed that has published member databases, or a published database is removed from an availability group, all copies of the published databases will be left in the recovering state. If restored, each will appear as a published database. Only one copy should be retained with publication metadata. To disable replication for a published database copy, first remove all subscriptions and publications from the database.

Run **sp_dropsubscription** to remove publication subscriptions. Make sure to set the parameter @ignore_distributributor to 1 to preserve the metadata for the active publishing database at the distributor.

```
USE MyDBName;

GO


EXEC sys.sp_dropsubscription

    @subscriber = 'MySubscriber',

    @publication = 'MyPublication',

    @article = 'all',

    @ignore_distributor = 1;
```

Run **sp_droppublication** to remove all publications. Again, set the parameter @ignore_distributor to 1 to preserve the metadata for the active publishing database at the distributor.

```
EXEC sys.sp_droppublication

    @publication = 'MyPublication',

    @ignore_distributor = 1;
```

Run **sp_replicationdboption** to disable replication for the database.

```
EXEC sys.sp_replicationdboption

    @dbname = 'MyDBName',

    @optname = 'publish',

    @value = 'false';
```

At this point, the copy of the published database can be retained or dropped.

Related Tasks

- Configuring Replication for AlwaysOn Availability Groups

- Replication, Change Tracking, Change Data Capture, and AlwaysOn Availability Groups (SQL Server)
- Administration (Replication)
- Replication Subscribers and AlwaysOn

See Also

Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups (SQL Server)

Overview of AlwaysOn Availability Groups

AlwaysOn Availability Groups: Interoperability

SQL Server Replication

Replication Subscribers and AlwaysOn

When an AlwaysOn availability group containing a database that is a replication subscriber fails over to an AlwaysOn secondary availability group, the replication subscription will fail. To resume replication, a replication administrator must manually reconfigure the subscriber. This topic provides a high-level overview of the process.

What is Supported

SQL Server replication supports the automatic failover of the publisher, and the manual failover of the subscriber. The failover of a distributor on an AlwaysOn availability group is not supported.

Cause of Failure

Failover of a Pull Subscription

A pull subscription fails upon AlwaysOn failover, because pull agent cannot find the jobs stored in the **msdb** database of the AlwaysOn primary; which is not available because the primary has failed.

Failover of a Push Subscription

A push subscription fails upon AlwaysOn failover, because the push agent can no longer connect to original subscription database on original subscriber.

How to Recover the Subscription

How to resume the merge and distribution agents after subscriber availability group failover?

1. Execute **sp_subscription_cleanup** to remove the old subscription on the subscriber. Perform this action on the new primary AlwaysOn availability group (which was formerly the secondary AlwaysOn availability group).
2. Recreate the subscription in one of the following two ways:
    - Remove the subscribed tables on the subscriber, and create a new subscription, beginning with a new snapshot.
    - For transactional replication, you can rely on the existing data as a starting point and execute the following two steps.

i. Execute the following statement on the secondary availability group subscriber to get **LSN** of last transaction received.

SELECT transaction_timestamp FROM MSreplication_subscriptions;

ii. Create a new subscription, using the `@sync_type = 'initialize from LSN'` and `@subscriptionlsn =` LSN arguments. The following example demonstrates this statement.

```
EXEC sp_addsubscription
@publication = 'MyPublication',
@subscriber = 'ALWAYSON-SRV5',
@destination_db = 'DB_Subscriber',
@subscription_Type = 'Push',
@sync_Type = 'initialize from LSN',
@subscriptionlsn = <insert the lsn value from the previous step>,
@article = 'all',
@update_mode = 'read only',
@subscriber_type = 0;
```

Important

- For merge replication, changes performed on the subscriber between after the failover and before the recreation of the subscription will not be replicated.
- For transactional replication, if the subscription is recreated within the retention period, the changes performed between the failover and the recreation of the subscription will be replicated.

Note

The current support is inconvenient for merge replication subscribers however the main scenario for merge replication is disconnected users (desktops, laptops, handset devices) which will not use AlwaysOn on the subscriber.

Examples

This lengthy example creates a replication environment on servers with AlwaysOn availability groups and then demonstrates the failover actions. Your server names and parameter values, such as lsn values, will be different.

The replication topology for this example is:

Publisher	(primary)	(secondary)
(server names)	**ALWAYSON-SRV2**	**ALWAYSON-SRV3**
--------------	--------------	--------------
Distributor		

(server and instance name)	**ALWAYSON-SRV8\distributor**	
--------------	--------------	--------------
Subscriber	(primary)	(secondary)
(server names)	**ALWAYSON-SRV4**	**ALWAYSON-SRV5**

Configure Replication on AlwaysOn, Configure the Published Database on an AlwaysOn AG, and Failover the Primary Publisher to its Secondary

The first portion of the example creates the replication environment.

```
------- BEGIN: Configure schema on Publisher -------
:Connect ALWAYSON-SRV2
CREATE DATABASE DB_Publisher;
GO
USE DB_Publisher;
GO
CREATE TABLE T1 (col1 int PRIMARY KEY, col2 varchar(256));
GO
INSERT INTO T1 VALUES (1, 'Row 1');
GO
SELECT * FROM DB_Publisher..T1;
------- END: Configure schema on Publisher -------

------- BEGIN: Configure schema on Subscriber -------
:Connect ALWAYSON-SRV4
CREATE DATABASE DB_Subscriber;
------- END: Configure schema on Subscriber -------

------- BEGIN: on Distributor, configure the distributor
--   as a remote distributor for the primary publisher -------
:connect ALWAYSON-SRV8\distributor
-- configure the remote publisher
USE master;
GO
EXEC sys.sp_adddistpublisher
```

```
@publisher = 'alwayson-srv2',
@distribution_db='distribution',
@working_directory='C:\Program Files\Microsoft SQL
Server\MSSQL11.DISTRIBUTOR\MSSQL\ReplData',
@security_mode=1;
------- END: on Distributor, configure the distributor
--   as a remote distributor for the primary publisher -------

------- BEGIN: Configure Replication on Publisher -------
:connect ALWAYSON-SRV2
-- Configure the remote distributor
USE master;
EXEC sp_adddistributor @distributor = N'ALWAYSON-SRV8\distributor', @password
= N'SQLServer2012';
GO
-- configure the publication database
USE master;
GO
EXEC sys.sp_replicationdboption @dbname='DB_Publisher', @optname='publish',
@value='true';
GO
-- create the publication for transactionnal replication
USE [DB_Publisher];
EXEC sp_addpublication @publication = N'MyPublication', @description =
N'Transactional publication of database ''DB_Publisher'' from Publisher
''ALWAYSON-SRV2''.', @sync_method = N'concurrent', @retention = 0,
@allow_push = N'true', @allow_pull = N'true', @allow_anonymous = N'true',
@enabled_for_internet = N'false', @snapshot_in_defaultfolder = N'true',
@compress_snapshot = N'false', @ftp_port = 21, @allow_subscription_copy =
N'false', @add_to_active_directory = N'false', @repl_freq = N'continuous',
@status = N'active', @independent_agent = N'true', @immediate_sync = N'true',
@allow_sync_tran = N'false', @allow_queued_tran = N'false', @allow_dts =
N'false', @replicate_ddl = 1, @allow_initialize_from_backup = N'false',
@enabled_for_p2p = N'false', @enabled_for_het_sub = N'false';
GO
```

```
EXEC sp_addpublication_snapshot @publication = N'MyPublication',
@frequency_type = 1, @frequency_interval = 1, @frequency_relative_interval =
1, @frequency_recurrence_factor = 0, @frequency_subday = 8,
@frequency_subday_interval = 1, @active_start_time_of_day = 0,
@active_end_time_of_day = 235959, @active_start_date = 0, @active_end_date =
0, @job_login = null, @job_password = null, @publisher_security_mode = 1;

USE [DB_Publisher];
EXEC sp_addarticle @publication = N'MyPublication', @article = N'T1',
@source_owner = N'dbo', @source_object = N'T1', @type = N'logbased',
@description = null, @creation_script = null, @pre_creation_cmd = N'drop',
@schema_option = 0x000000000803509F, @identityrangemanagementoption =
N'manual', @destination_table = N'T1', @destination_owner = N'dbo',
@vertical_partition = N'false', @ins_cmd = N'CALL sp_MSins_dboT1', @del_cmd =
N'CALL sp_MSdel_dboT1', @upd_cmd = N'SCALL sp_MSupd_dboT1';
GO
EXEC sp_changepublication @publication = 'MyPublication', @property =
'allow_initialize_from_backup', @value = 'true';

-- Start the snapshot agent

-- create push subscription
:Connect to SRV2
USE [DB_Publisher];
EXEC sp_addsubscription @publication = N'MyPublication', @subscriber =
N'ALWAYSON-SRV4', @destination_db = N'DB_Subscriber', @subscription_type =
N'Push', @sync_type = N'automatic', @article = N'all', @update_mode = N'read
only', @subscriber_type = 0;
EXEC sp_addpushsubscription_agent @publication = N'MyPublication',
@subscriber = N'ALWAYSON-SRV4', @subscriber_db = N'DB_Subscriber', @job_login
= null, @job_password = null, @subscriber_security_mode = 1, @frequency_type
= 1, @frequency_interval = 0, @frequency_relative_interval = 0,
@frequency_recurrence_factor = 0, @frequency_subday = 0,
@frequency_subday_interval = 0, @active_start_time_of_day = 0,
@active_end_time_of_day = 0, @active_start_date = 0, @active_end_date =
19950101, @enabled_for_syncmgr = N'False', @dts_package_location =
N'Distributor';
```

```
GO

-- Sync the distribution agent

-- Connect to the subscriber and check that data is there
:connect ALWAYSON-SRV4
SELECT * FROM DB_Subscriber.dbo.T1;

--Check replication is running properly
--Connect to publisher and change data
:connect ALWAYSON-SRV2
USE DB_Publisher;
GO
UPDATE t1 SET col2 = 'Something else' WHERE col1 = 1;
GO
SELECT * FROM DB_Publisher..T1;
GO

-- Run distribution agent and check that data is there
:connect ALWAYSON-SRV4
SELECT * FROM DB_Subscriber..T1;
------- BEGIN: Configure Replication on Publisher -------

------- BEGIN: Configure AlwaysOn on the Publisher -------
-- Connect to primary publisher
:connect ALWAYSON-SRV2
BACKUP DATABASE DB_Publisher TO DISK = '\\alwayson-
srv8\MyBackups\DB_Publisher_Full';

-- From the UI create an AG with the wizard

-- Connect to publisher secondary to insure
-- it is configured for Replication
:connect ALWAYSON-SRV3
```

```
SELECT @@SERVERNAME;
USE master;
GO
DECLARE @installed int;
EXEC @installed = sys.sp_MS_replication_installed;
SELECT @installed;

-- Declare the publisher secondary as
-- a potential failover publisher
:connect ALWAYSON-SRV8\distributor
USE master;
GO
EXEC sys.sp_adddistpublisher
@publisher = 'ALWAYSON-SRV3',
@distribution_db='distribution',
@working_directory='C:\Program Files\Microsoft SQL
Server\MSSQL11.DISTRIBUTOR\MSSQL\ReplData',
@security_mode=1;

-- Connect to publisher secondary to configure
-- the remote distributor
:connect ALWAYSON-SRV3
USE master;
EXEC sp_adddistributor @distributor = N'ALWAYSON-SRV8\distributor', @password
= N'SQLServer2012';
GO
-- ensure each push subscriber appear as linked servers
EXEC sys.sp_addlinkedserver @server = 'ALWAYSON-SRV4';
GO
EXEC sys.sp_addlinkedserver @server = 'ALWAYSON-SRV5';
GO

-- Redirect the original publisher
-- to the publisher AG listener name
```

```
:connect ALWAYSON-SRV8\distributor
USE distribution;
GO
EXEC sys.sp_redirect_publisher
@original_publisher = 'ALWAYSON-SRV2',
@publisher_db = 'DB_Publisher',
@redirected_publisher = 'PublisherAGList';
GO
-- Show the new metadata table that has been created
-- and filled to persist this configuration
SELECT * FROM distribution..MSredirected_publishers;

-- Run the Replication validation stored procedure
-- to verify that primary and secondary publishers
-- can serve as publishers for the published database
USE distribution;
GO
DECLARE @redirected_publisher sysname;
EXEC sys.sp_validate_replica_hosts_as_publishers @original_publisher =
'ALWAYSON-SRV2', @publisher_db = 'DB_Publisher', @redirected_publisher =
@redirected_publisher OUTPUT;
SELECT @redirected_publisher;
------- END: Configure AlwaysOn on the publisher -------
```

Subscriber Failover and Recovery Steps of a Subscription for Transactional Replication

This portion of the example demonstrates the failover actions.

```
------- BEGIN: Configure AlwaysOn on the Subscriber -------
:Connect to SRV4
BACKUP DATABASE DB_Subscriber TO DISK = '\\alwayson-
srv8\MyBackups\DB_Subscriber_Full';
-- From the UI create an AG with the wizard
------- END: Configure AlwaysOn on the Subscriber -------

------- BEGIN: demo subscriber failover and recovery steps -------
```

```
-- Failover the subscriber AG
:connect to ALWAYSON-SRV5
ALTER AVAILABILITY GROUP SubscriberAG FAILOVER;

-- Get the LSN from the last transaction received
-- by the subscriber
:Connect to SRV5
SELECT transaction_timestamp, * FROM
DB_Subscriber..MSreplication_subscriptions;
--0x00000023000000E60003000000000000

-- Cleanup old subscription from new subscription DB
EXEC sp_subscription_cleanup @publisher = 'ALWAYSON-SRV2', @publisher_db =
'DB_Publisher';

-- Create new subscription to the new Subscriber Primary based
-- on the LSN of the last successful sync
:Connect to PublisherAGList
USE DB_Publisher;
GO
EXEC sp_addsubscription -- past the LSN before executing
@publication = 'MyPublication',
@subscriber = 'ALWAYSON-SRV5',
@destination_db = 'DB_Subscriber',
@subscription_Type = 'Push',
@sync_Type = 'initialize from LSN',
@subscriptionlsn = 0x00000023000000E60003000000000000,
@article = 'all',
@update_mode = 'read only',
@subscriber_type = 0;

-- Note that the distribution agent is running
-- Note that data is replicated from publisher


------- END: demo subscriber failover and recovery steps -------
```

Replication, Change Tracking, Change Data Capture, and AlwaysOn Availability Groups

Replication, Change Data Capture (CDC), and Change Tracking (CT) are supported on AlwaysOn Availability Groups. AlwaysOn helps provide high availability and additional database recovery capabilities.

In this Topic:

- Overview of Replication on AlwaysOn Availability Groups
    - Publisher Redirection
    - Changes to Replication Agents to Support AlwaysOn Availability Groups
    - Stored Procedures Supporting AlwaysOn
    - Change Data Capture
    - Change Tracking
- Prerequisites, Restrictions, and Considerations for Using Replication with AlwaysOn Availability Groups
- Related Tasks

Overview of Replication on AlwaysOn Availability Groups

Publisher Redirection

When a published database is AlwaysOn aware, the distributor that provides agent access to the publishing database is configured with redirected_publishers entries. These entries redirect the originally configured publisher/database pair, making use of an availability group listener name to connect to the publisher and publishing database. Established connections through the availability group listener name will fail on failover. When the replication agent restarts after failover, the connection will automatically be redirected to the new primary.

In an AlwaysOn Availability Group an AlwaysOn secondary cannot be a publisher. Republishing is not supported when replication is combined with AlwaysOn.

If a published database is a member of an AlwaysOn availability group and the publisher is redirected, it must be redirected to an availability group listener name associated with the availability group. It may not be redirected to an explicit node.

Note

After failover to a secondary, Replication Monitor is unable to adjust the name of the publishing instance of SQL Server and will continue to display replication information under the name of the original primary instance of SQL Server. After failover, a tracer token cannot be entered by using the Replication Monitor, however a tracer token entered on the new publisher by using Transact-SQL, is visible in Replication Monitor.

(top)

General Changes to Replication Agents to Support AlwaysOn Availability Groups

Three replication agents were modified to support AlwaysOn Availability Groups. The Log Reader, Snapshot, and Merge agents were modified to query the distribution database for the redirected publisher and to use the returned availability group listener name, if a redirected publisher was declared, to connect to the database publisher.

By default, when the agents query the distributor to determine whether the original publisher has been redirected, the suitability of the current target or redirection will be verified prior to returning the redirected host to the agent. This is recommended behavior. However, if agent start up occurs very frequently the overhead associated with the validation stored procedure may be deemed too costly. A new command line switch, BypassPublisherValidation, has been added to the Logreader, Snapshot, and Merge agents. When the switch is used, the redirected publisher is returned immediately to the agent and execution of the validation stored procedure is bypassed.

Failures returned from the validation stored procedure are logged in the agent history logs. Those errors with severity greater than or equal to 16 will cause the agents to terminate. Some retry capabilities have been built in to the agents to handle the expected disconnect from a published database when it fails over to a new primary.

(top)

Log Reader Agent Modifications

The Logreader Agent has the following changes.

- **Replicated Database Consistency**

  When a published database is a member of an AlwaysOn availability group, by default the log reader will not process log records that have not already been hardened at all availability group secondary replicas. This insures that on failover, all rows replicated to a subscriber also are present at the new primary.

  When the publisher has only two AlwaysOn replicas (one primary and one secondary) and a failover happens, the original primary remains down because the logreader does not move forward until all secondaries are brought back online or until the failing AlwaysOn replicas are removed from the availability group. The logreader, now running against the secondary, will not proceed forward since AlwaysOn cannot harden any changes to any secondary. To allow the logreader to proceed further and still have disaster recovery capacity, remove the original primary from the Availability Group using **ALTER AVAILABITY GROUP** <group_name> **REMOVE REPLICA**. Then add another secondary to the availability group.

- **Trace flag 1448**

  Trace flag 1448 enables the replication log reader to move forward even if the async secondaries have not acknowledged the reception of a change. Even with this trace flag enabled the log reader always waits for the sync secondaries. The log reader will not go beyond the min ack of the sync secondaries. This trace flag applies to the instance of SQL Server, not just an availability group, an availability database, or a log reader instance. Takes effect immediately without a restart. This trace flag can be activated ahead of time or when an async secondary fails.

(top)

Stored Procedures Supporting AlwaysOn

- **sp_redirect_publisher**

  The stored procedure **sp_redirect_publisher** is used to specify a redirected publisher for an existing publisher/database pair. If the publisher database belongs to an AlwaysOn availability group, the redirected publisher is the availability group listener name.

- **sp_get_redirected_publisher**

  The stored procedure **sp_get_redirected_publisher** is used by replication agents to query a distributor to determine whether a publisher/database pair has a defined redirected publisher. This stored procedure serves two purposes. First, it allows the agent to determine whether the original publisher has been redirected. Second, it may also initiate a validation stored procedure run at the distributor (**sp_validate_redirected_publisher**) that verifies the suitability of the target node of the redirection to serve as a publisher for the named database.

  To execute this stored procedure the caller must either be a member of the **sysadmin** server role, the **db_owner** database role for the distribution database, or a member of a **Publication Access List** for a defined publication associated with the publisher database.

- **sp_validate_redirected_publisher**

  This stored procedure attempts to validate that the current publisher is capable of hosting the published database. It can be called at any time to verify that the current host for the published database is capable of supporting replication.

- **sp_validate_replicate_hosts_as_publishers**

  While it is useful for the agents to insure that the current primary can function as the replication publisher for a publisher database, a more general validation capability is needed to establish the validity of an entire AlwaysOn replication topology. The stored procedure **sp_validate_replica_hosts_as_publishers** is designed to fill this need.

  This stored procedure is always run manually. The caller must either be **sysadmin** at the distributor, **dbowner** of the distribution database, or a member of the **Publication Access List** of a publication of the publisher database. In addition, the login of the caller must be a valid login for all of the availability group replica hosts, and have select privileges on the availability group database associated with the publisher database.

(top)

Change Data Capture

Databases enabled for Change Data Capture (CDC) are able to leverage AlwaysOn Availability Groups in order to insure not only that the database remains available in the event of failure, but that changes to the database tables continue to be monitored and deposited in the CDC change tables. The order in which CDC and AlwaysOn Availability Groups are configured is not important. CDC enabled databases can be added to AlwaysOn Availability Groups, and databases that are members of an AlwaysOn Availability Groups can be enabled for CDC. In both cases, however, CDC configuration is always performed on the current or intended AlwaysOn primary replica.

- **Harvesting Changes for Change Data Capture Without Replication**

  If CDC is enabled for a database, but replication is not, the capture process used to harvest changes from the log and deposit them in CDC change tables runs at the CDC host as its own SQL Agent job.

  In order to resume the harvesting of changes after failover, the stored procedure **sp_cdc_add_job** must be run at the new primary to create the local capture job.

  The following example creates the capture job.

  ```
  EXEC sys.sp_cdc_add_job @job_type = 'capture';
  ```

- **Harvesting Changes for Change Data Capture With Replication**

  If both CDC and replication are enabled for a database, the log reader handles the population of the CDC change tables. In this case, the techniques used by replication to leverage AlwaysOn availability groups will insure that changes continue to be harvested from the log and deposited in CDC change tables after failover. Nothing additional needs to be done for CDC in this configuration to insure that the change tables are populated.

- **Change Data Capture Cleanup**

  To insure that appropriate cleanup occurs at the new primary, a local cleanup job should always be created. The following example creates the cleanup job.

  ```
  EXEC sys.sp_cdc_add_job @job_type = 'cleanup';
  ```

  Note
  You should create the jobs at all of the possible failover targets before failover, and mark them as disabled until the replica at a host becomes primary. The CDC jobs running at the old primary should be also disabled when the local database becomes a secondary replica. To disable and enable jobs, use the @enabled option of sp_update_job (Transact-SQL). For more information about creating CDC jobs, see sys.sp_cdc_add_job (Transact-SQL).

- **Adding CDC Roles to an AlwaysOn Primary Database Replica**

  When a table is enabled for CDC, it is possible to associate a database role with the capture instance. If a role is specified, the user wishing to use the CDC table-valued functions to access changes for the table must not only have select access to the tracked table columns, but must also be a member of the named role. If the specified role does not already exist, the role will be created. When database roles are automatically added to a database that is a member of a primary replica, the roles are also propagated to the availability group secondary replica databases.

- **Client Applications Accessing CDC Change Data and Always On**

Client applications that use the table-valued functions (TVFs) or linked servers to access change table data also need the ability to locate an appropriate CDC host after failover. The availability group listener name is the mechanism provided by AlwaysOn to transparently allow a connection to be retargeted to a different host. Once an availability group listener name is associated with an availability group, it is available to be used in TCP connection strings. Two different connection scenarios are supported through the availability group listener name.

- One insures that connection requests are always directed to the current primary.
- One insures that connection requests are directed to a read-only secondary.

If used to locate a read-only secondary, a read-only routing list must also be defined for the availability group. For more information on routing access to readable secondaries, see To Configure Availability Replicas for Read-Only Routing.

Note
There is some propagation delay associated with the creation of an availability group listener name and its use by client applications to access an availability group database replica.

Use the following query to determine whether an availability group listener name has been defined for the availability group hosting a CDC database. The query will return the availability group listener name if one has been created.

```
SELECT dns_name
FROM sys.availability_group_listeners AS l
INNER JOIN sys.availability_databases_cluster AS d
    ON l.group_id = d.group_id
WHERE d.database_name = N'MyCDCDB';
```

- **Redirecting the Query Load to a Readable Secondary**

While in many cases a client application will always want to connect to the current primary replica that is not the only way to leverage AlwaysOn availability groups. If an availability group is configured to support readable secondary replicas, change data can also be gathered from secondary nodes.

When an availability group is configured, the **ALLOW_CONNECTIONS** attribute associated with the **SECONDARY_ROLE** is used to specify the type of secondary access supported. If configured as **ALL**, all connections to the secondary will be allowed, but only those requiring read only access will succeed. If configured as **READ_ONLY**, it is necessary to specify read only intent when making the connection to the secondary database in order for the connection to succeed.

The following query can be used to determine whether read-only intent is needed to connect to an availability group readable secondary.

```
SELECT g.name AS AG, replica_server_name,
secondary_role_allow_connections_desc
```

```
FROM sys.availability_replicas AS r
JOIN sys.availability_groups AS g
    ON r.group_id = g.group_id
WHERE g.name = N'MY_AG_NAME;
```

Either the availability group listener name or the explicit node name can be used to locate the secondary. If the availability group listener name is used, access will be directed to any suitable secondary.

When **sp_addlinkedserver** is used to create a linked server to access the secondary, the @datasrc parameter is used for the availability group listener name or the explicit server name, and the @provstr parameter is used to specify read-only intent.

```
EXEC sp_addlinkedserver
@server = N'linked_svr',
@srvproduct=N'SqlServer',
@provider=N'SQLNCLI11',
@datasrc=N'AG_Listener_Name',
@provstr=N'ApplicationIntent=ReadOnly',
@catalog=N'MY_DB_NAME';
```

- **Client Access to CDC Change Data and Domain Logins**

  In general, you should use domain logins for client access to change data residing in databases that are members of AlwaysOn availability groups. To insure continued access to change data after failover, the domain user will need access privileges on all of the hosts supporting availability group replicas. If a database user is added to a database in a primary replica, and the user is associated with a domain login, the database user is propagated to secondary databases and continues to be associated with the specified domain login. If the new database user is associated with a SQL Server authentication login, the user at the secondary databases will be propagated without a login. While the associated SQL Server authentication login could be used to access change data at the primary where the database user was originally defined, that node is the only one where access would be possible. The SQL Server authentication login would not be able to access data from any secondary database, nor from any new primary databases other than the original database where the database user was defined.

(top)

Change Tracking

A database enabled for Change Tracking (CT) can be part of an AlwaysOn availability group. No additional configuration is needed. Change Tracking client applications that use the CDC table-valued functions (TVFs) to access change data will need the ability to locate the primary replica after failover. If the client application connects through the availability group listener name, connection requests will always be appropriately directed to the current primary replica.

Note

- Change tracking data must always be obtained from the primary replica. An attempt to access change data from a secondary replica will result in the following error:
- Msg 22117, Level 16, State 1, Line1
- For databases that are members of a secondary replica, change tracking is not supported. Run change tracking queries on the databases in the primary replica.

(top)

Prerequisites, Restrictions, and Considerations for Using Replication

This section describes considerations for deploying replication with AlwaysOn Availability Groups, including prerequisites, restrictions, and recommendations.

Prerequisites

- The Distributor, all Publishers, and merge pull Subscribers must be running SQL Server 2012 or later.
- Placing the distribution database on an availability group is not supported.
- The Publisher instances satisfy all the prerequisites required to participate in an AlwaysOn availability group. For more information see Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups (SQL Server).

Restrictions

Supported combinations of replication on AlwaysOn Availability Groups:

	Publisher	**Distributor**[3]	**Subscriber**
Transactional	Yes[1]	No	Yes[2]
P2P	No	No	No
Merge	Yes	No	Yes[2]
Snapshot	Yes	No	Yes[2]

[1] Does not include support for bi-directional and reciprocal transactional replication.

[2] Failover to the replica database is manual procedure. Automatic failover is not provided.

[3] The Distributor database is not supported for use with AlwaysOn Availability Groups or Database Mirroring.

(top)

Considerations

- The distribution database is not supported for use with AlwaysOn Availability Groups or Database Mirroring. Replication configuration is coupled to the SQL Server instance where the Distributor is configured; therefore the distribution database cannot be mirrored or replicated. To provide high availability for the Distributor, use a SQL Server failover cluster. For more information, see AlwaysOn Failover Cluster Instances (FCI).

- Subscriber failover to a secondary replica database, while supported, is a relatively complex manual procedure. The procedure is essentially identical to the method used to fail over a mirrored subscriber database. Subscribers must be running SQL Server 2012 or later to participate in an availability group.
- Metadata and objects that exist outside the database are not propagated to the secondary replicas, including logins, jobs, linked servers. If you require the metadata and objects at the new primary after failover, you must copy them manually. For more information, see Management of Logins and Jobs for the Databases of an Availability Group (SQL Server).

(top)

Related Tasks

Replication

- Configure Replication for AlwaysOn Availability Groups
- Maintaining An AlwaysOn Publication Database
- Administration (Replication)

Change data capture

- Enable and Disable Change Data Capture
- Administer and Monitor Change Data Capture
- Work with Change Data

Change tracking

- Enable and Disable Change Tracking
- Manage Change Tracking
- Work with Change Tracking

See Also

Replication Subscribers and AlwaysOn

Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups (SQL Server)

Overview of AlwaysOn Availability Groups

AlwaysOn Availability Groups: Interoperability

AlwaysOn Failover Cluster Instances (FCI)

About Change Data Capture

About Change Tracking

SQL Server Replication

Track Data Changes

sys.sp_cdc_add_job (Transact-SQL)

Analysis Services with AlwaysOn Availability Groups

An AlwaysOn availability group is a predefined collection of SQL Server relational databases that failover together when conditions trigger a failover in any one database, redirecting requests to a mirrored database on another instance in the same availability group. If you are using availability groups as your high availability solution, you can use a database in that group as a data source in an Analysis Services tabular or multidimensional solution. All of the following Analysis Services operations work as expected when using an availability database: processing or importing data, querying relational data directly (using ROLAP storage or DirectQuery mode), and writeback.

Processing and querying are read-only workloads. You can improve performance by offloading these workloads to a readable secondary replica. Additional configuration is required for this scenario. Use the checklist in this topic to ensure you follow all the steps.

Prerequisites

You must have a SQL Server login on all replicas. You must be a **sysadmin** to configure availability groups, listeners, and databases, but users only need **db_datareader** permissions to access the database from an Analysis Services client.

Use a data provider that supports the tabular data stream (TDS) protocol version 7.4 or newer, such as the SQL Server Native Client 11.0 or the Data Provider for SQL Server in .NET Framework 4.02.

(For read-only workloads). The secondary replica role must be configured for read-only connections, the availability group must have a routing list, and the connection in the Analysis Services data source must specify the availability group listener. Instructions are provided in this topic.

Checklist: Use a secondary replica for read-only operations

Unless your Analysis Services solution includes writeback, you can configure a data source connection to use a readable secondary replica. If you have a fast network connection, the secondary replica has very low data latency, providing nearly identical data as the primary replica. By using the secondary replica for Analysis Services operations, you can reduce read-write contention on the primary replica and get better utilization of secondary replicas in your availability group.

By default, both read-write and read-intent access are allowed to the primary replica and no connections are allowed to secondary replicas. Additional configuration is required to set up a read-only client connection to a secondary replica. Configuration requires setting properties on the secondary replica and running a T-SQL script that defines a read-only routing list. Use the following procedures to ensure you have performed both steps.

Note
The following steps assume an existing AlwaysOn availability group and databases. If you are configuring a new group, use the New Availability Group Wizard to create the group and join the databases. The wizard checks for prerequisites, provides guidance for each step, and performs the initial synchronization. For more information, see Use the New Availability Group Wizard (SQL Server Management Studio).

Step 1: Configure access on an availability replica

1. In Object Explorer, connect to the server instance that hosts the primary replica, and expand the server tree.

   Note
   These steps are taken from Configure Read-Only Access on an Availability Replica (SQL Server), which provides additional information and alternative instructions for performing this task.
2. Expand the **AlwaysOn High Availability** node and the **Availability Groups** node.
3. Click the availability group whose replica you want to change. Expand **Availability Replicas**.
4. Right-click the secondary replica, and click **Properties**.
5. In the **Availability Replica Properties** dialog box, change the connection access for the secondary role, as follows:
   - In the **Readable secondary** drop list, select **Read-intent only**.
   - In the **Connections in primary role** drop list, select **Allow all connections**. This is the default.
   - Optionally, in **Availability mode** drop list, select **Synchronous commit**. This step is not required, but setting it ensures that there is data parity between the primary and secondary replica.

     This property is also a requirement for planned failover. If you want to perform a planned manual failover for testing purposes, set **Availability mode** to **Synchronous commit** for both the primary and secondary replica.

Step 2: Configure read-only routing

1. Connect to the primary replica.

   Note
   These steps are taken from Configure Read-Only Routing for an Availability Group (SQL Server), which provides additional information and alternative instructions for performing this task.

2. Open a query window and paste in the following script. This script does three things: enables readable connections to a secondary replica (which is off by default), sets the read-only routing URL, and creates the routing list that prioritizes how connection requests are directed. The first statement, allowing readable connections, is redundant if you already set the properties in Management Studio, but are included for completeness.

```
ALTER AVAILABILITY GROUP [AG1]
 MODIFY REPLICA ON
N'COMPUTER01' WITH
(SECONDARY_ROLE (ALLOW_CONNECTIONS = READ_ONLY));

ALTER AVAILABILITY GROUP [AG1]
 MODIFY REPLICA ON
N'COMPUTER01' WITH
(SECONDARY_ROLE (READ_ONLY_ROUTING_URL =
N'TCP://COMPUTER01.contoso.com:1433'));

ALTER AVAILABILITY GROUP [AG1]
 MODIFY REPLICA ON
N'COMPUTER02' WITH
(SECONDARY_ROLE (ALLOW_CONNECTIONS = READ_ONLY));

ALTER AVAILABILITY GROUP [AG1]
 MODIFY REPLICA ON
N'COMPUTER02' WITH
(SECONDARY_ROLE (READ_ONLY_ROUTING_URL =
N'TCP://COMPUTER02.contoso.com:1433'));

ALTER AVAILABILITY GROUP [AG1]
MODIFY REPLICA ON
N'COMPUTER01' WITH
(PRIMARY_ROLE
(READ_ONLY_ROUTING_LIST=('COMPUTER02','COMPUTER01')));

ALTER AVAILABILITY GROUP [AG1]
```

```
MODIFY REPLICA ON
N'COMPUTER02' WITH
(PRIMARY_ROLE
(READ_ONLY_ROUTING_LIST=('COMPUTER01','COMPUTER02')));
GO
```

3. Modify the script, replacing placeholders with values that are valid for your deployment:
   - Replace 'Computer01' with the name of the server instance that hosts the primary replica.
   - Replace 'Computer02' with the name of the server instance that hosts the secondary replica.
   - Replace 'contoso.com' with the name of your domain, or omit it from the script if all computers are in the same domain. Keep the port number if the listener is using the default port. The port that is actually used by the listener is listed in the properties page in Management Studio.
4. Execute the script.

   Next, create a data source in an Analysis Services model that uses a database from the group you just configured.

Create an Analysis Services data source using an AlwaysOn availability database

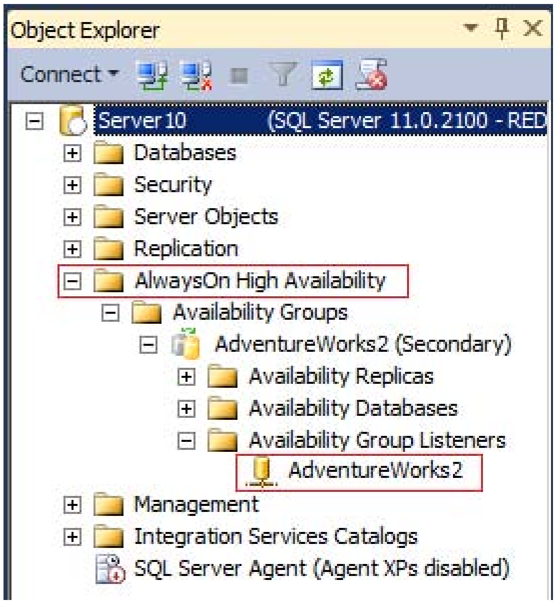
This section explains how to create an Analysis Services data source that connects to a database in an availability group. You can use these instructions to configure a connection to either a primary replica (default) or a readable secondary replica that you configured based on steps in a previous section. AlwaysOn configuration settings, plus the connection properties set in the client, will determine whether a primary or secondary replica is used.

1. In SQL Server Data Tools, in an Analysis Services Multidimensional and Data Mining Model project, right-click **Data Sources** and select **New Data Source**. Click **New** to create a new data source.

   Alternatively, for a tabular model project, click the Model menu, and then click **Import from Data Source**.
2. In Connection Manager, in Provider, choose a provider that supports the Tabular Data Stream (TDS) protocol. The SQL Server Native Client 11.0 supports this protocol.
3. In Connection Manager, in Server Name, enter the name of the *availability group listener*, and then choose a database that is available in the group.

   The availability group listener redirects a client connection to a primary replica for read-write requests or to a secondary replica if you specify read-intent in the connection string. Because replica roles will change during a failover (where the primary becomes the secondary and a secondary becomes a primary), you should always specify the listener so that the client connection is redirected accordingly.

To determine the name of the availability group listener, you can either ask a database administrator or connect to an instance in the availability group and view its AlwaysOn availability configuration. In the screenshot below, the availability group listener is **AdventureWorks2**.


Object Explorer
Connect
Server10 (SQL Server 11.0.2100 - RED
Databases
Security
Server Objects
Replication
AlwaysOn High Availability
Availability Groups
AdventureWorks2 (Secondary)
Availability Replicas
Availability Databases
Availability Group Listeners
AdventureWorks2
Management
Integration Services Catalogs
SQL Server Agent (Agent XPs disabled)

4. Still in Connection Manager, click **All** in the left navigation pane to view the property grid of data provider.

   Set **Application Intent** to **READONLY** if you are configuring a read-only client connection to a secondary replica. Otherwise, keep the **READWRITE** default to redirect the connection to the primary replica.

5. In Impersonation Information, select **Use a specific Windows user name and password**, and then enter a Windows domain user account that has a minimum of **db_datareader** permissions on the database.

   Do not choose **Use the credentials of the current user** or **Inherit**. You can choose **Use the service account**, but only if that account has read permissions on the database.

   Finish the data source and close the Data Source Wizard.

6. Add **MultiSubnetFailover=Yes** to the connection string to provide faster detection and connection to the active server. For more information about this property, see [SQL Server Native Client Support for High Availability, Disaster Recovery](#).

   This property is not visible in the property grid. To add the property, right-click the data source and choose **View Code**. Add `MultiSubnetFailover=Yes` to the connection string.

The data source is now defined. You can now proceed to build a model, starting with the data source view, or in the case of tabular models, creating relationships. When you are at a point where data must be retrieved from the availability database (for example when you are ready to process or deploy the solution), you can test the configuration to verify data is accessed from the secondary replica.

Test the configuration

After you configure the secondary replica and create a data source connection in Analysis Services, you can confirm that processing and query commands are redirected to the secondary replica. You can also perform a planned manual failover to verify your recovery plan for this scenario.

▶Step 1: Confirm the data source connection is redirected to the secondary replica

1. Start SQL Server Profiler and connect to the SQL Server instance hosting the secondary replica.

   As the trace runs, the **SQL:BatchStarting** and **SQL:BatchCompleting** events will show the queries issued from Analysis Services that are executing on the database engine instance. These events are selected by default so all you need to do is start the trace.
2. In SQL Server Data Tools, open the Analysis Services project or solution containing a data source connection you want to test. Be sure that the data source specifies the availability group listener and not an instance in the group.

   This step is important. Routing to the secondary replica will not occur if you specify a server instance name.
3. Arrange the application windows so that you can view SQL Server Profiler and SQL Server Data Tools side by side.
4. Deploy the solution, and when it completes, stop the trace.

   In the trace window, you should see events from the application **Microsoft SQL Server Analysis Services**. You should see **SELECT** statements that retrieve data from a database on the server instance that hosts the secondary replica, proving that the connection was made via the listener to the secondary replica.

▶Step 2: Perform a planned failover to test the configuration

1. In Management Studio check the primary and secondary replicas to ensure that both are configured for synchronous-commit mode and are currently synchronized.

   The following steps assume a secondary replica is configured for synchronous commit.

   To verify synchronization, open a connection to each instance that hosts the primary and secondary replicas, expand the Databases folder, and ensure that the database has **(Synchronized)** and **(Synchronizing)** appended to its name in each replica.

Note

These steps are taken from Perform a Planned Manual Failover of an Availability Group (SQL Server), which provides additional information and alternative instructions for performing this task.

2. In SQL Server Profiler, start traces for each replica and view the traces side-by-side. In the following steps, you will compare traces, confirming that the SQL queries used for processing or querying from Analysis Services switch from one replica to the other.
3. Execute a processing or query command from within Analysis Services. Because you configured the data source for a read-only connection, you should see the command execute on the secondary replica.
4. In Management Studio, connect to the secondary replica.
5. Expand the **AlwaysOn High Availability** node and the **Availability Groups** node.
6. Right-click the availability group to be failed over, and select the **Failover** command. This starts the Fail Over Availability Group Wizard. Use the wizard to choose which replica to make the new primary replica.
7. Confirm that failover succeeded:
    - In Management Studio, expand the availability groups to view the (primary) and (secondary) designations. The instance that was previously a primary replica should now be a secondary replica.
    - View the dashboard to determine if any health issues were detected. Right-click the availability group and select **Show Dashboard**.
8. Wait one or two minutes for the failover to complete on the backend.
9. Repeat the processing or query command in the Analysis Services solution, and then watch the traces side by side in SQL Server Profiler. You should see evidence of processing on the other instance, which is now the new secondary replica.

What happens after a failover occurs

During a failover, a secondary replica transitions to the primary role and the former primary replica transitions to the secondary role. All client connections are terminated, ownership of the availability group listener moves with the primary replica role to a new SQL Server instance, and the listener endpoint is bound to the new instance's virtual IP addresses and TCP ports. For more information, see About Client Connection Access to Availability Replicas (SQL Server).

If failover occurs during processing, the following error occurs in Analysis Services in the log file or output window: "OLE DB error: OLE DB or ODBC error: Communication link failure; 08S01; TPC Provider: An existing connection was forcibly closed by the remote host. ; 08S01."

This error should resolve if you wait a minute and try again. If the availability group is configured correctly for readable secondary replica, processing will resume on the new secondary replica when you retry processing.

Persistent errors are most likely due to a configuration problem. You can try re-running the T-SQL script to resolve problems with the routing list, read-only routing URLs, and read-intent on the secondary replica. You should also verify that the primary replica allows all connections.

Writeback when using an AlwaysOn availability database

Writeback is an Analysis Services feature that supports What If analysis in Excel. It is also commonly used for budgeting and forecasting tasks in custom applications.

Support for writeback requires a READWRITE client connection. In Excel, if you attempt to write back on a read-only connection, the following error will occur: "Data could not be retrieved from the external data source." "Data could not be retrieved from the external data source."

If you configured a connection to always access a readable secondary replica, you must now configure a new connection that uses a READWRITE connection to the primary replica.

To do this, create an additional data source in an Analysis Services model to support the read-write connection. When creating the additional data source, use the same listener name and database that you specified in the read-only connection, but instead of modifying **Application Intent**, keep the default that supports READWRITE connections. You can now add new fact or dimension tables to your data source view that are based on the read-write data source, and then enable writeback on the new tables.

See Also

Availability Group Listeners, Client Connectivity, and Application Failover (SQL Server)

Readable Secondary Replicas (AlwaysOn Availability Groups)

AlwaysOn Policies for Operational Issues with AlwaysOn Availability Groups (SQL Server)

Define a Data Source (SSAS - Multidimensional Models)

Enable Dimension Writeback

Reporting Services with AlwaysOn Availability Groups

This topic contains information about configuring Reporting Services to work with AlwaysOn Availability Groups (AG) in SQL Server 2012. The three scenarios for using Reporting Services and AlwaysOn Availability Groups are databases for report data sources, report server databases, and report design. The supported functionality and required configuration is different for the three scenarios.

A key benefit of using AlwaysOn Availability Groups with Reporting Services data sources is to leverage readable secondary replicas as a reporting data source while, at the same time the secondary replicas are providing a failover for a primary database.

For general information on AlwaysOn Availability Groups, see AlwaysOn FAQ for SQL Server 2012 (http://msdn.microsoft.com/en-us/sqlserver/gg508768).

In This Topic:

Requirements for Using Reporting Services and AlwaysOn Availability Groups

Report Data Sources and Availability Groups

Requirements for using Reporting Services and AlwaysOn Availability Groups

To use AlwaysOn Availability Groups with SQL Server 2012 Reporting Services, you need to download and install a hotfix for .Net 3.5 SP1. The hotfix adds support to SQL Client for AG features and support of the connection string properties **ApplicationIntent** and **MultiSubnetFailover**. If the Hotfix is not installed on each computer that hosts a report server, then users attempting to preview reports will see an error message similar to the following, and the error message will be written to the report server trace log:

Error message:

The message occurs when you include one of the AlwaysOn Availability Groups properties in the Reporting Services connection string, but the server does not recognize the property. The noted error message will be seen when you click the 'Test Connection' button in Reporting Services user interfaces and when you preview the report if remote errors are enabled on the report servers.

For more information on the required hotfix, see KB 2654347A hotfix introduces support for the AlwaysOn features from SQL Server 2012 to the .NET Framework 3.5 SP1.

For information on other AlwaysOn Availability Groups requirements, see Prerequisites, Restrictions, and Recommendations for AlwaysOn Availability Groups (SQL Server).

Note

Reporting Services configuration files such as **RSreportserver.config** are not supported as part of AlwaysOn Availability Groups functionality. If you manually make changes to a configuration file on one of the report servers, you will need to manually update the replicas.

Top

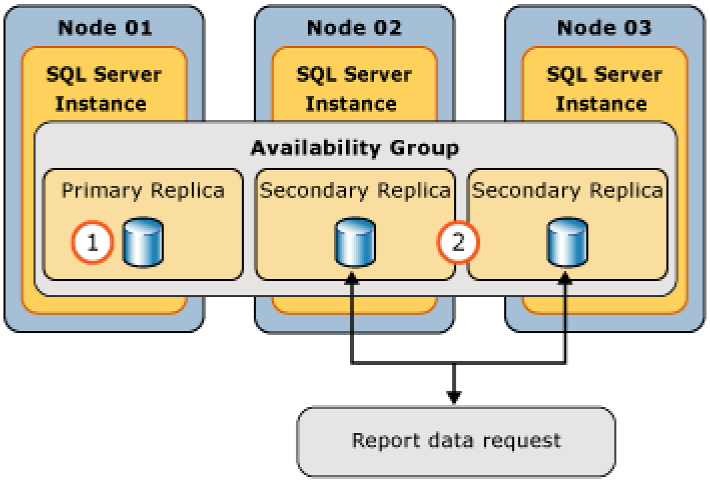
Report Data Sources and Availability Groups

The behavior of Reporting Services data sources based on AlwaysOn Availability Groups can vary depending on how your administrator has configured the AG environment.

To utilize AlwaysOn Availability Groups for report data sources you need to configure the report data source connection string is to use the availability group *Listener DNS name*. Supported data sources are the following:

- ODBC data source using SQL Native Client.
- SQL Client, with the .Net hotfix applied to the report server.

The connection string can also contain new AlwaysOn connection properties that configure the report query requests to use secondary replica for read-only reporting. Use of secondary replica for reporting requests will reduce the load on a read-write primary replica. The following illustration is an example of a three replica AG configuration where the Reporting Services data source connection strings have been configured with ApplicationIntent=ReadOnly. In this example the report query requests are sent to a secondary replica and not the primary replica.



The following is an example connection string, where the [AvailabilityGroupListenerName] is the **Listener DNS Name** that was configured when replicas were created:

```
Data Source=[AvailabilityGroupListenerName];Initial Catalog =
AdventureWorks2008R2; ApplicationIntent=ReadOnly
```

The **Test Connection** button in Reporting Services user interfaces will validate if a connection can be established but it will not validate AG configuration. For example if you include ApplicationIntent in a connection string to a server that is not part of AG, the extra parameter is ignored and the **Test Connection** button will only validate a connection can be established to the specified server.

Depending on how your reports are created and published will determine where you edit the connection string:

- **Native mode:** Use Report Manager for shared data sources and reports that are already published to a native mode report server.
- **SharePoint Mode:** Use SharePoint configuration pages within the document libraries for reports that are already published to a SharePoint server.
- **Report Design:** SQL Server Report Builder for SQL Server 2012 or SQL Server Data Tools (SSDT) when you are creating new reports. See the 'Report Design' section in this topic or more information.

Additional Resources:

Manage Report Data Sources

For more information on the available connection string properties, see Using Connection String Keywords with SQL Server Native Client.

For more information on availability group listeners, see Create or Configure an Availability Group Listener (SQL Server).

Considerations: Secondary replicas will typically experience a delay in receiving data changes from the primary replica. The following factors can affect the update latency between the primary and secondary replicas:

- The number of secondary replicas. The delay increases with each secondary replica added to the configuration.
- Geographic location and distance between the primary and secondary replicas. For example the delay is typically larger if the secondary replicas are in a different data center than if they were in the same building as the primary replica.
- Configuration of the availability mode for each replica. The availability mode determines whether the primary replica waits to commit transactions on a database until a secondary replica has written the transaction to disk. For more information, see the 'Availability Modes' section of Overview of AlwaysOn Availability Groups (SQL Server).

When using a read-only secondary as a Reporting Services data source, it is important to ensure that data update latency meets the needs of the report users.

Top

Report Design and Availability Groups

When designing reports in SQL Server Report Builder for SQL Server 2012 or a report project in SQL Server Data Tools (SSDT), a user can configure a report data source connection string to contain new connection properties provided by AlwaysOn Availability Groups. Support for the new connection properties depends on where a user previews the report.

- **Local preview:** SQL Server Report Builder for SQL Server 2012 and SQL Server Data Tools (SSDT) use the .Net framework 4.0 and support AlwaysOn Availability Groups connection string properties.
- **Remote or server mode preview:** If after publishing reports to the report server or using preview in SQL Server Report Builder for SQL Server 2012, you see an error similar to the following, it is an indication you are previewing reports against the report server and the .Net Framework 3.5 SP1 Hotfix for AlwaysOn Availability Groups has not been installed on the report server.

Error message:

Top

Report Server Databases and Availability Groups

Reporting Services offers limited support for using AlwaysOn Availability Groups with report server databases. The report server databases can be configured in AG to be part of a replica; however Reporting Services will not automatically use a different replica for the report server databases when a failover occurs.

Manual actions or custom automation scripts need to be used to complete the failover and recovery. Until these actions are completed, some features of the report server may not work correctly after the AlwaysOn Availability Groups failover.

Note
When planning failover and disaster recovery for the report server databases, it is advised you always backup a copy of the report server encryption key.

Top

Differences between SharePoint Native Mode

This section summarizes the differences between how SharePoint mode and Native mode report servers interact with AlwaysOn Availability Groups.

A SharePoint report server creates **3** databases for each Reporting Services service application you create. The connection to the report server databases in SharePoint mode is configured in SharePoint Central Administration when you create the service application. The default names of the databases include a GUID that is associated with the service application. The following are example database names, for a SharePoint mode report server:

- ReportingService_85c08ac3c8e64d3cb400ad06ed5da5d6
- ReportingService_85c08ac3c8e64d3cb400ad06ed5da5d6TempDB
- ReportingService_85c08ac3c8e64d3cb400ad06ed5da5d6_Alerting

Native mode report servers use **2** databases. The following are example database names, for a native mode report server:

- ReportServer
- ReportServerTempDB

Native mode does not support or use the Alerting databases and related features. You configure native mode report servers in the Reporting Services Configuration Manager. For SharePoint mode you configure the service application database name to be the name of the "client access point" you created as part of the SharePoint configuration. For more information on configuring SharePoint with AlwaysOn Availability Groups, see Configure and manage SQL Server availability groups for SharePoint Server (http://go.microsoft.com/fwlink/?LinkId=245165).

Note
- SharePoint mode report servers use a synchronization process between the Reporting Services service application databases and the SharePoint content databases. It is important to maintain the report server databases and content databases together. You should consider configuring them in the same availability groups so they failover and recover as a set. Consider the following scenario:

Top

Prepare Report Server Databases for Availability Groups

The following are the basic steps of preparing and adding the report server databases to an AlwaysOn Availability Groups:

- Create your Availability Group and configure a *Listener DNS name*.
- **Primary Replica:** Configure the report server databases to be part of a single availability group and create a primary replica that includes all of the report server databases. This includes
- **Secondary Replicas:** Create one or more secondary replicas. The common approach to copying the databases from the primary replica to the secondary replica(s) is to restore the databases to each secondary replica using 'RESTORE WITH NORECOVERY'. For more information on creating secondary replicas and verifying data synchronization is working, see Start Data Movement on an AlwaysOn Secondary Database (SQL Server).
- **Report Server Credentials:** You need to create the appropriate report server credentials on the secondary replicas that you created on the primary. The exact steps depend on what type of authentication you are using in your Reporting Services environment; Window Reporting Services service account, Windows user account, or SQL Server authentication. For more information, see Configure a Report Server Database Connection (Native Mode)
- Update the database connection to use the Lister DNS Name. for natve mode report servers, change the **Report Server Database Name** in ssRSnoverison configuration manager. For SharePoint mode, change the **Database server name** for the Reporting Services service application(s).

Top

Steps to complete disaster recovery of Report Server Databases

The following steps need to be completed after a AlwaysOn Availability Groups failover to a secondary replica:

1. Stop the instance of the SQL Agent service that was being used by the primary database engine hosting the Reporting Services databases.
2. Start SQL Agent service on the computer that is the new primary replica.
3. Stop the Report Server service.

   If the report server is in native mode, stop the report server Windows server using Reporting Services configuration manager.

   If the report server is configured for SharePoint mode, stop the Reporting Services shared service in SharePoint Central Administration.
4. Start the report server service or Reporting Services SharePoint service.
5. Verify that reports can run against the new primary replica.

Top

Report Server Behavior When a Failover Occurs

When report server databases failover and you have updated the report server environment to use the new primary replica, there are some operational issues that result from the failover and recovery process. The impact of these issues will vary depending on the Reporting Services load at the time of failover as well as the length of time it takes for AlwaysOn Availability Groups to failover to a secondary replica and for the report server administrator to update the reporting environment to use the new primary replica.

- The execution of background processing may occur more than once due to retry logic and the inability of the report server to mark scheduled work as completed during the failover period.
- The execution of background processing that would have normally been triggered to run during the period of the failover will not occur because SQL Server Agent will not be able to write data into the report server database and this data will not be synchronized to the new primary replica.
- After the database failover completes and after the report server service is re-started, SQL Server Agent jobs will be re-created automatically. Until the SQL agent jobs are recreated, any background executions associated with SQL Server Agent jobs will not be processed. This includes Reporting Services subscriptions, schedules, an snapshots.

Top

See Also

SQL Server Native Client Support for High Availability, Disaster Recovery

AlwaysOn Availability Groups (SQL Server)

Get Started with AlwaysOn Availability Groups (SQL Server)

Using Connection String Keywords with SQL Server Native Client

SQL Server Native Client Support for High Availability, Disaster Recovery

Client Connection Access to Availability Replicas (SQL Server)

Service Broker with AlwaysOn Availability Groups

This topic contains information about configuring Service Broker to work with AlwaysOn Availability Groups in SQL Server 2012.

In This Topic:

- Requirements for a Service in an Availability Group to Receive Remote Messages
- Requirements for Sending Messages to a Remote Service in an Availability Group

Requirements for a Service in an Availability Group to Receive Remote Messages

1. **Ensure that the availability group possesses a listener.**

   For more information, see Create or Configure an Availability Group Listener (SQL Server).

2. **Ensure that the Service Broker endpoint exists and is correctly configured.**

   On every instance of SQL Server that hosts an availability replica for the availability group, configure the Service Broker endpoint, as follows:

   - Set LISTENER_IP to 'ALL'. This setting enables connections on any valid IP address that is bound to the availability group listener.

- Set the Service Broker PORT to the same port number on all the host server instances.

  Tip
  To view the port number of the Service Broker endpoint on a given server instance, query the **port** column of the sys.tcp_endpoints catalog view, where **type_desc** = 'SERVICE_BROKER'.

The following example creates a Windows authenticated Service Broker endpoint that uses the default Service Broker port (4022) and listens to all valid IP addresses.

```
CREATE ENDPOINT [SSBEndpoint]
    STATE = STARTED
    AS TCP  (LISTENER_PORT = 4022, LISTENER_IP = ALL )
    FOR SERVICE_BROKER (AUTHENTICATION = WINDOWS)
```

For more information, see CREATE ENDPOINT (Transact-SQL).

3. **Grant CONNECT permission on the endpoint.**

   Grant CONNECT permission on the Service Broker endpoint either to PUBLIC or to a login.

   The following example grants the connection on a Service Broker endpoint named `broker_endpoint` to PUBLIC.

```
GRANT CONNECT ON ENDPOINT::[broker_endpoint] TO [PUBLIC]
```

   For more information, see GRANT (Transact-SQL).

4. **Ensure that msdb contains either an AutoCreatedLocal route or a route to the specific service.**

   Note
   By default, each user database, including **msdb**, contains the route **AutoCreatedLocal**. This route matches any service name and broker instance and specifies that the message should be delivered within the current instance. **AutoCreatedLocal** has lower priority than routes that explicitly specify a specific service that communicates with a remote instance.

   For information about creating routes, see Service Broker Routing Examples (in the SQL Server 2008 R2 version of Books Online) and CREATE ROUTE (Transact-SQL).

Requirements for Sending Messages to a Remote Service in an Availability Group

1. **Create a route to the target service.**

   Configure the route as follows:

   - Set ADDRESS to the listener IP address of availability group that hosts the service database.
   - Set PORT to the port that you specified in the Service Broker endpoint of each of the remote SQL Server instances.

The following example creates a route named `RouteToTargetService` for the `ISBNLookupRequestService` service. The route targets the availability group listener, `MyAgListener`, which uses port 4022.

```
CREATE ROUTE [RouteToTargetService] WITH

SERVICE_NAME = 'ISBNLookupRequestService',

ADDRESS = 'TCP://MyAgListener:4022';
```

For more information, see CREATE ROUTE (Transact-SQL).

2. **Ensure that msdb contains either an AutoCreatedLocal route or a route to the specific service.** (For more information, see Requirements for a Service in an Availability Group to Receive Remote Messages, earlier in this topic.)

Related Tasks

- CREATE ENDPOINT (Transact-SQL)
- CREATE ROUTE (Transact-SQL)
- GRANT (Transact-SQL)
- Create or Configure an Availability Group Listener (SQL Server).
- Creation and Configuration of Availability Groups (SQL Server)
- Set Up Login Accounts for Database Mirroring or AlwaysOn Availability Groups (SQL Server)

See Also

Overview of AlwaysOn Availability Groups

Availability Group Listeners, Client Connectivity, and Application Failover

SQL Server Service Broker

The Database Mirroring Endpoint

To participate in AlwaysOn Availability Groups or database mirroring a server instance requires its own, dedicated *database mirroring endpoint*. This endpoint is a special-purpose endpoint that is used exclusively to receive connections from other server instances. On a given server instance, every AlwaysOn Availability Groups or database mirroring connection to any other server instance uses a single database mirroring endpoint.

Database mirroring endpoints use Transmission Control Protocol (TCP) to send and receive messages between the server instances participating database mirroring sessions or hosting availability replicas. The database mirroring endpoint listens on a unique TCP port number.

Note

Client connections to a principal server or primary replica do not use the database mirroring endpoint.

Note
The database mirroring 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 database mirroring to use AlwaysOn Availability Groups instead.

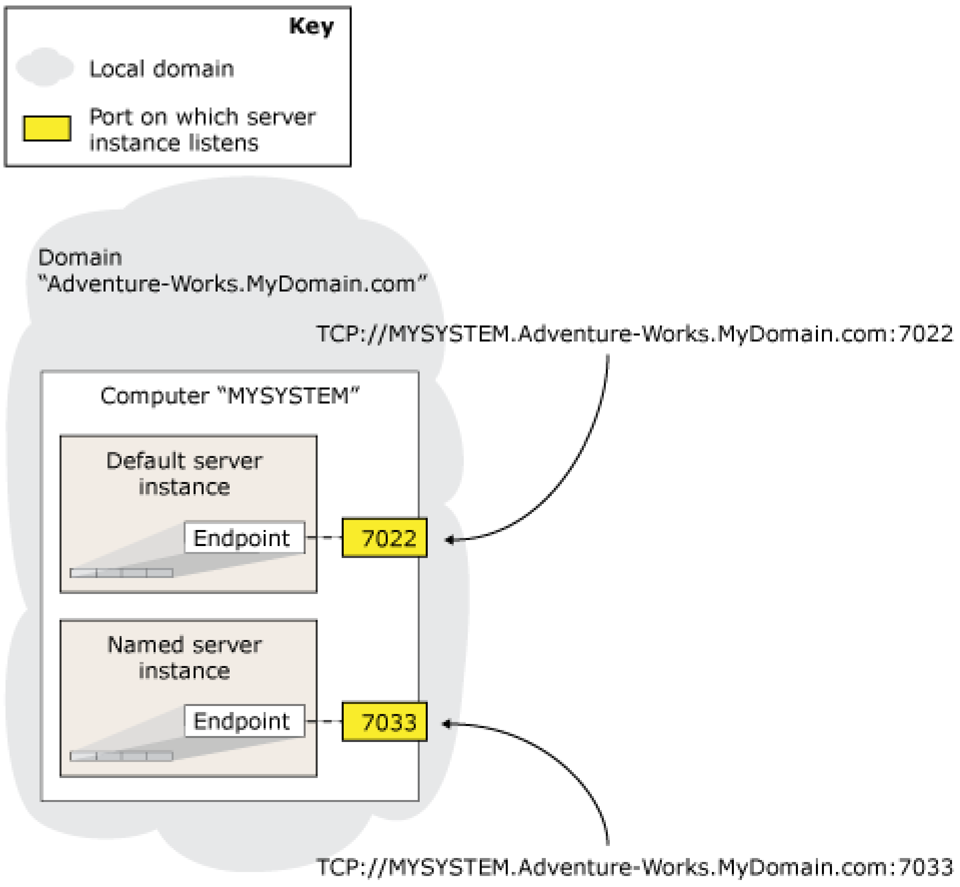
In this Topic:

- Server Network Address
- Determining the Authentication Type for a Database Mirroring Endpoint
- Related Tasks

Server Network Address

The network address of a server instance (its *server network address* or *Endpoint URL*) contains the port number of its endpoint, as well as the system and domain name of its host computer. The port number uniquely identifies a specific server instance.

The following figure illustrates how two server instances on the same server are uniquely identified. The server network addresses of both server instances contain the same system name, `MYSYSTEM`, and domain name, `Adventure-Works.MyDomain.com`. To enable the system to route connections to a server instance, a server network address includes the port number associated with the mirroring endpoint of a particular server instance.


Key
Local domain
Port on which server instance listens
Domain "Adventure-Works.MyDomain.com"
TCP://MYSYSTEM.Adventure-Works.MyDomain.com:7022
Computer "MYSYSTEM"
Default server instance
Endpoint
7022
Named server instance
Endpoint
7033
TCP://MYSYSTEM.Adventure-Works.MyDomain.com:7033

By default, an instance of SQL Server does not contain a database mirroring endpoint. These must be created manually as part of setting up a database mirroring session. The system administrator must create a separate endpoint in each server instance that is to participate in database mirroring. Note that if more than one server instance on a given computer requires a database mirroring endpoint, specify a different port number for each endpoint.

noteDXDOC112778PADS Security Note

If the computer running SQL Server has a firewall, the firewall configuration must allow both incoming and outgoing connections for the port specified in the endpoint.

For database mirroring and AlwaysOn Availability Groups, authentication and encryption are configured on the endpoint. For more information, see Database Mirroring Transport Security.

Important

Do not reconfigure an in-use database mirroring endpoint. The server instances use each other's endpoints to learn the state of the other systems. If the endpoint is reconfigured, it might restart, which can appear to be an error to the other server instances. This is particularly important for automatic failover mode, in which reconfiguring the endpoint on a partner could cause a failover to occur.

Determining the Authentication Type for a Database Mirroring Endpoint

It is important to understand that the SQL Server service accounts of your server instances determine what type of authentication you can use for your database mirroring endpoints, as follows:

- If every server instance is running under a domain service account, you can use Windows Authentication for your database mirroring endpoints. If all the server instances run as the same domain user account, the correct user logins exist automatically in both **master** databases. This simplifies the security configuration for the availability databases and is recommended.

  If any server instances that are hosting the availability replicas for an availability group run as different accounts, the login each account must be created in **master** on the other server instance. Then, that login must be granted CONNECT permissions to connect to the database mirroring endpoint of that server instance. For more information, Setting Up Login Accounts for Database Mirroring.

  If your server instances use Windows Authentication, you can create database mirroring endpoints by using Transact-SQL, PowerShell, or the New Availability Group Wizard.

  Note
  If a server instance that is to host an availability replica lacks a database mirroring endpoint, the New Availability Group Wizard can automatically create a database mirroring endpoint that uses Windows Authentication. For more information, see Use the Availability Group (New Availability Group Wizard).

- If any server instance is running under a built-in account, such as Local System, Local Service, or Network Service, or a nondomain account, you must use certificates for endpoint authentication. If you are using certificates for your database mirroring endpoints, your system administrator must configure each server instance to use certificates on both outbound and inbound connections.

  There is no automated method for configuring database mirroring security using certificates. You will need to use either CREATE ENDPOINT Transact-SQL statement or the **New-SqlHadrEndpoint** PowerShell cmdlet. For more information, see CREATE ENDPOINT (Transact-SQL). For information about enabling certificate authentication on a server instance, see Use Certificates for a Database Mirroring Endpoint.

Related Tasks

To Configure a Database Mirroring Endpoint

- Create a Mirroring Endpoint for Windows Authentication (Transact-SQL)
- Use Certificates for a Database Mirroring Endpoint
  - Allow Database Mirroring to Use Certificates for Outbound Connections (Transact-SQL)

    - Allow a Database Mirroring Endpoint to Use Certificates for Inbound Connections (Transact-SQL)
- Specify a Server Network Address (Database Mirroring)
- Specify the Endpoint URL When Adding or Modifying an Availability Replica (SQL Server)
- Use the New Availability Group Wizard (SQL Server Management Studio)

To View Information About the Database Mirroring Endpoint

- sys.database_mirroring_endpoints (Transact-SQL)

See Also

Transport Security for Database Mirroring and AlwaysOn Availability Groups (SQL Server)

Troubleshoot Database Mirroring Configuration

sys.dm_hadr_availability_replica_states (Transact-SQL)

sys.dm_db_mirroring_connections

Transport Security for Database Mirroring and AlwaysOn Availability Groups

Transport security in SQL Server 2005 and later versions involves authentication and, optionally, encryption of messages exchanged between the databases. For database mirroring and AlwaysOn Availability Groups, authentication and encryption are configured on the database mirroring endpoint. For an introduction to database mirroring endpoints, see The Database Mirroring Endpoint.

In this Topic:

- Authentication
- Data Encryption
- Related Tasks

Authentication

Authentication is the process of verifying that a user is who the user claims to be. Connections between database mirroring endpoints require authentication. Connection requests from a partner or witness, if any, must be authenticated.

The type of authentication used by a server instance for database mirroring or AlwaysOn Availability Groups is a property of the database mirroring endpoint. Two types of transport security are available for database mirroring endpoints: Windows Authentication (the Security Support Provider Interface (SSPI)) and certificate-based authentication.

Windows Authentication

Under Windows Authentication, each server instance logs in to the other side using the Windows credentials of the Windows user account under which the process is running. Windows Authentication might require some manual configuration of login accounts, as follows:

- If the instances of SQL Server run as services under the same domain account, no extra configuration is required.
- If the instances of SQL Server run as services under different domain accounts (in the same or trusted domains), the login of each account must be created in **master** on each of the other server instances, and that login must be granted CONNECT permissions on the endpoint.
- If the instances of SQL Server run as the Network Service account, the login of the each host computer account (*DomainName\ComputerName$*) must be created in **master** on each of the other servers, and that login must be granted CONNECT permissions on the endpoint. This is because a server instance running under the Network Service account authenticates using the domain account of the host computer.

Note
For an example of setting up a database mirroring session using Windows Authentication, see Example of Setting Up Database Mirroring.

Certificates

In some situations, such as when server instances are not in trusted domains or when SQL Server is running as a local service, Windows Authentication is unavailable. In such cases, instead of user credentials, certificates are required to authenticate connection requests. The mirroring endpoint of each server instance must be configured with its own locally created certificate.

The encryption method is established when the certificate is created. For more information, see How to: Allow Database Mirroring to Use Certificates for Outbound Connections (Transact-SQL). Carefully manage the certificates that you use.

A server instance uses the private key of its own certificate to establish its identity when setting up a connection. The server instance that receives the connection request uses the public key of the sender's certificate to authenticate the sender's identity. For example, consider two server instances, Server_A and Server_B. Server_A uses its private key to encrypt the connection header before sending a connection request to Server_B. Server_B uses the public key of Server_A's certificate to decrypt the connection header. If the decrypted header is correct, Server_B knows that the header was encrypted by Server_A, and the connection is authenticated. If the decrypted header is incorrect, Server_B knows that the connection request is inauthentic and refuses the connection.

Data Encryption

By default, a database mirroring endpoint requires encryption of data sent over mirroring connections. In this case, the endpoint can connect only to endpoints that also use encryption. Unless you can guarantee that your network is secure, we recommend that you require encryption for your database mirroring connections. However, you can disable encryption or make it supported, but not required. If encryption is disabled, data is never encrypted and the endpoint cannot connect to an endpoint that requires encryption. If encryption is supported, data is encrypted only if the opposite endpoint either supports or requires encryption.

Note
Mirroring endpoints created by SQL Server Management Studio are created with encryption either required or disabled. To change the encryption setting to SUPPORTED, use the ALTER ENDPOINT Transact-SQL statement. For more information, see ALTER ENDPOINT (Transact-SQL).

Optionally, you can control the encryption algorithms that can be used by an endpoint, by specifying one of the following values for the ALGORITHM option in a CREATE ENDPOINT statement or ALTER ENDPOINT statement:

ALGORITHM value	Description
RC4	Specifies that the endpoint must use the RC4 algorithm. This is the default. **Note** The RC4 algorithm is deprecated. This feature will be removed in a future version of Microsoft SQL Server. Do not use this feature in new development work, and modify applications that currently use this feature as soon as possible. We recommend that you use AES.
AES	Specifies that the endpoint must use the AES algorithm.
AES RC4	Specifies that the two endpoints will negotiate for an encryption algorithm with this endpoint giving preference to the AES algorithm.
RC4 AES	Specifies that the two endpoints will negotiate for an encryption algorithm with this endpoint giving preference to the RC4 algorithm.

If connecting endpoints specify both algorithms but in different orders, the endpoint accepting the connection wins.

Note

- The RC4 algorithm is only supported for backward compatibility. New material can only be encrypted using RC4 or RC4_128 when the database is in compatibility level 90 or 100. (Not recommended.) Use a newer algorithm such as one of the AES algorithms instead. In SQL Server 2012 material encrypted using RC4 or RC4_128 can be decrypted in any compatibility level.
- Though considerably faster than AES, RC4 is a relatively weak algorithm, while AES is a relatively strong algorithm. Therefore, we recommend that you use the AES algorithm.

For information about the Transact-SQL syntax for specifying encryption, see CREATE ENDPOINT (Transact-SQL).

Related Tasks

To configure transport security for a database mirroring endpoint

- Create a Database Mirroring Endpoint for Windows Authentication (Transact-SQL)
- How to: Configure a Database Mirroring Session (SQL Server Management Studio)
- How to: Create a Mirroring Endpoint for Windows Authentication (Transact-SQL)
- How to: Allow Database Mirroring to Use Certificates for Outbound Connections (Transact-SQL)

See Also

Choosing an Encryption Algorithm

ALTER ENDPOINT (Transact-SQL)

DROP ENDPOINT (Transact-SQL)

Security and Protection (Database Engine)

Setting up Login Accounts for Database Mirroring

The Database Mirroring Endpoint

sys.database_mirroring_endpoints (Transact-SQL)

sys.dm_database_mirroring_connections

Troubleshooting Database Mirroring Setup

Troubleshoot AlwaysOn Availability Groups Configuration (SQL Server)

Create a Database Mirroring Endpoint for Windows Authentication (Transact-SQL)

This topic describes how to create a database mirroring endpoint that uses Windows Authentication in SQL Server 2012 by using Transact-SQL. To support database mirroring or AlwaysOn Availability Groups each instance of SQL Server requires a database mirroring endpoint. A server instance can have only one database mirroring endpoint, which has a single port. A database mirroring endpoint can use any port that is available on the local system when the endpoint is created. All database mirroring sessions on a server instance listen on that port, and all incoming connections for database mirroring use that port.

Important

If a database mirroring endpoint exists and is already in use, we recommend that you use that endpoint. Dropping an in-use endpoint disrupts existing sessions.

In This Topic

- **Before you begin:** Security
- **To create a database mirroring endpoint, using:** Transact-SQL

Before You Begin

Security

The authentication and encryption methods of the server instance are established by the system administrator.

noteDXDOC112778PADS Security Note

The RC4 algorithm is deprecated. This feature will be removed in a future version of Microsoft SQL Server. Do not use this feature in new development work, and modify applications that currently use this feature as soon as possible. We recommend that you use AES.

Permissions

Requires CREATE ENDPOINT permission, or membership in the sysadmin fixed server role. For more information, see GRANT Endpoint Permissions (Transact-SQL).

Using Transact-SQL

To Create a Database Mirroring Endpoint That Uses Windows Authentication

1. Connect to the instance of SQL Server on which you want to create a database mirroring endpoint.
2. From the Standard bar, click **New Query**.
3. Determine if a database mirroring endpoint already exists by using the following statement:

```
SELECT name, role_desc, state_desc FROM
sys.database_mirroring_endpoints
```

   Important

   If a database mirroring endpoint already exists for the server instance, use that endpoint for any other sessions you establish on the server instance.

4. To use Transact-SQL to create an endpoint to use with Windows Authentication, use a CREATE ENDPOINT statement. The statement takes the following general form:

```
CREATE ENDPOINT <endpointName>
   STATE=STARTED
   AS TCP ( LISTENER_PORT = <listenerPortList>  )
   FOR DATABASE_MIRRORING
   (
      [ AUTHENTICATION = WINDOWS [ <authorizationMethod> ]
      ]
      [ [,] ENCRYPTION = REQUIRED
            [ ALGORITHM { <algorithm> } ]
      ]
      [,] ROLE = <role>
   )
```

   where

   - *<endpointName>* is a unique name for the database mirroring endpoint of the server instance.
   - STARTED specifies that the endpoint is to be started and to begin listening for connections. A database mirroring endpoint typically is created in the STARTED state. Alternatively, you can start a session in a STOPPED state (the default) or DISABLED state.
   - *<listenerPortList>* is a single port number (*nnnn*) on which you want the server to listen for database mirroring messages. Only TCP is allowed; specifying any other protocol causes an error.

     A port number can be used only once per computer system. A database mirroring endpoint can use any port that is available on the local system when the endpoint is created. To identify the ports currently being used by TCP endpoints on the system, use the following Transact-SQL statement:

     ```
     SELECT name, port FROM sys.tcp_endpoints
     ```

     Important
     Each server instance requires one and only one unique listener port.
   - For Windows Authentication, the AUTHENTICATION option is optional, unless you want the endpoint to use only NTLM or Kerberos to authenticate connections. *<authorizationMethod>* specifies the method used to authenticate connections as one of the following: NTLM, KERBEROS, or NEGOTIATE. The default, NEGOTIATE, causes the endpoint to use the Windows negotiation protocol to choose either NTLM or Kerberos. Negotiation enables connections with or without authentication, depending on the authentication level of the opposite endpoint.

- ENCRYPTION is set to REQUIRED by default. This means that all connections to this endpoint must use encryption. However, you can disable encryption or make it optional on an endpoint. The alternatives are as follows:

Value	Definition
DISABLED	Specifies that data sent over a connection is not encrypted.
SUPPORTED	Specifies that the data is encrypted only if the opposite endpoint specifies either SUPPORTED or REQUIRED.
REQUIRED	Specifies that data sent over a connection must be encrypted.

  If an endpoint requires encryption, the other endpoint must have ENCRYPTION set to either SUPPORTED or REQUIRED.
- <*algorithm*> provides the option of specifying the encryption standards for the endpoint. The value of <*algorithm*> can be one following algorithms or combinations of algorithms: RC4, AES, AES RC4, or RC4 AES.

  AES RC4 specifies that this endpoint will negotiate for the encryption algorithm, giving preference to the AES algorithm. RC4 AES specifies that this endpoint will negotiate for the encryption algorithm, giving preference to the RC4 algorithm. If both endpoints specify both algorithms but in different orders, the endpoint accepting the connection wins.

  Note
  The RC4 algorithm is deprecated. This feature will be removed in a future version of Microsoft SQL Server. Do not use this feature in new development work, and modify applications that currently use this feature as soon as possible. We recommend that you use AES.
- <*role*> defines the role or roles that the server can perform. Specifying ROLE is required. However, the role of the endpoint is relevant only for database mirroring. For AlwaysOn Availability Groups, the role of the endpoint is ignored.

  To allow a server instance to serve as one role for one database mirroring session and different role for another session, specify ROLE=ALL. To restrict a server instance to being either a partner or a witness, specify ROLE=PARTNER or ROLE=WITNESS, respectively.

  Note
  For more information about Database Mirroring options for different editions of SQL Server, see Features Supported by the Editions of SQL Server

2012 (http://go.microsoft.com/fwlink/?linkid=232473).

For a complete description of the CREATE ENDPOINT syntax, see CREATE ENDPOINT (Transact-SQL).

Note
To change an existing endpoint, use ALTER ENDPOINT.

Example: Creating Endpoints to Support for Database Mirroring (Transact-SQL)

The following example creates database mirroring endpoints for the default server instances on three separate computer systems:

Role of server instance	Name of host computer
Partner (initially in the principal role)	`SQLHOST01\.`
Partner (initially in the mirror role)	`SQLHOST02\.`
Witness	`SQLHOST03\.`

In this example, all three endpoints use port number 7022, though any available port number would work. The AUTHENTICATION option is unnecessary, because the endpoints use the default type, Windows Authentication. The ENCRYPTION option is also unnecessary, because the endpoints are all intended to negotiate the authentication method for a connection, which is the default behavior for Windows Authentication. Also, all of the endpoints require the encryption, which is the default behavior.

Each server instance is limited to serving as either a partner or a witness, and the endpoint of each server expressly specifies which role (ROLE=PARTNER or ROLE=WITNESS).

Important
Each server instance can have only one endpoint. Therefore, if you want a server instance to be a partner in some sessions and the witness in others, specify ROLE=ALL.

```
--Endpoint for initial principal server instance, which
--is the only server instance running on SQLHOST01.
CREATE ENDPOINT endpoint_mirroring
    STATE = STARTED
    AS TCP ( LISTENER_PORT = 7022 )
    FOR DATABASE_MIRRORING (ROLE=PARTNER);
GO
--Endpoint for initial mirror server instance, which
--is the only server instance running on SQLHOST02.
```

```
CREATE ENDPOINT endpoint_mirroring
    STATE = STARTED
    AS TCP ( LISTENER_PORT = 7022 )
    FOR DATABASE_MIRRORING (ROLE=PARTNER);
GO
--Endpoint for witness server instance, which
--is the only server instance running on SQLHOST03.
CREATE ENDPOINT endpoint_mirroring
    STATE = STARTED
    AS TCP ( LISTENER_PORT = 7022 )
    FOR DATABASE_MIRRORING (ROLE=WITNESS);
GO
```

Related Tasks

To Configure a Database Mirroring Endpoint

- Create a Database Mirroring Endpoint for AlwaysOn Availability Groups (SQL Server PowerShell)
- Use Certificates for a Database Mirroring Endpoint
  - Allow Database Mirroring to Use Certificates for Outbound Connections (Transact-SQL)
  - Allow a Database Mirroring Endpoint to Use Certificates for Inbound Connections (Transact-SQL)
- Specify a Server Network Address (Database Mirroring)
- Specify the Endpoint URL When Adding or Modifying an Availability Replica (SQL Server)

To View Information About the Database Mirroring Endpoint

- sys.database_mirroring_endpoints (Transact-SQL)

See Also

ALTER ENDPOINT (Transact-SQL)

Choosing an Encryption Algorithm

CREATE ENDPOINT

Specifying a Server Network Address (Database Mirroring)

Example of Setting Up Database Mirroring Using Windows Authentication

The Database Mirroring Endpoint

Set Up Login Accounts for Database Mirroring or AlwaysOn Availability Groups

For two server instances to connect to each other's database mirroring endpoint point, the login account of each instance requires access to the other instance. Also, each login account requires connect permission to the database mirroring endpoint of the other instance.

The impact of this requirement depends on whether the server instances run as the same domain user account:

- If the server instances run as the same domain user account, the correct user logins exist automatically in both **master** databases. This simplifies the security configuration the database and is recommended.
- If the server instances run as different user accounts, user logins on the server instance that hosts the principal server or primary replica must be manually reproduced on the server instance that hosts the mirror server or on every server instance that hosts a secondary replica. For more information, see Create a Login for a Different Account and Grant Connect Permission, later in this topic.

Create a Login for a Different Account

If two server instances run as different accounts, the system administrator must use the CREATE LOGIN Transact-SQL statement to create a login for the startup service account of the remote instance in the **syslogins** table of the **master** database of each server instance. For more information, see CREATE LOGIN (Transact-SQL).

Important
If you run SQL Server under a non-domain account, you must use certificates. For more information, see Use Certificates for a Database Mirroring Endpoint (SQL Server).

For example, for the server instance sqlA, which runs under loginA, to connect to the server instance sqlB, which runs under loginB, loginA must be in the **syslogins** table on sqlB, and loginB must be in the **syslogins** table on sqlA. In addition, for a database mirroring session that includes a witness server instance (sqlC) and in which the three server instances run under different domain accounts, the following logins must be created:

On instance...	Create logins for and grant connection permission to ...
sqlA	sqlB and sqlC
sqlB	sqlA and sqlC
sqlC	sqlA and sqlB

Note
It is possible to connect with the network service account by using the machine account instead of a domain user. If the machine account is used, it must be added as a user on the other server instance.

Grant Connect Permission

Once a login has been created on a server instance, the login must be granted permission to connect to the database mirroring endpoint of the server instance. The system administrator grants the connect permission using a GRANT Transact-SQL statement. For more information, see GRANT (Transact-SQL).

Related Tasks

- Create a Login (SQL Server Database Engine)
- Allow Database Mirroring Network Access Using Windows Authentication (Transact-SQL).
- Use Certificates for a Database Mirroring Endpoint (SQL Server)

See Also

Database Mirroring Endpoint

Troubleshooting Database Mirroring Setup

Troubleshoot AlwaysOn Availability Groups Configuration

Allow Network Access to a Database Mirroring Endpoint Using Windows Authentication

Using Windows Authentication for connecting the database mirroring endpoints of two instances of SQL Server requires manual configuration of login accounts under the following conditions:

- If the instances of SQL Server run as services under different domain accounts (in the same or trusted domains), the login of each account must be created in **master** on each of the remote server instances and that login must be granted CONNECT permissions on the endpoint.
- If the instances of SQL Server run as the Network Service account, the login of the each host computer account (*DomainName*\ComputerName$) must be created in **master** on each of the remote server instances and that login must be granted CONNECT permissions on the endpoint. This is because a server instance running under the Network Service account authenticates using the domain account of the host computer.

Note
Ensure that an endpoint exists for each of the server instances. For more information, see Create a Database Mirroring Endpoint for Windows Authentication (Transact-SQL).

Procedures

To configure logins for Windows Authentication

1. For the user account of each instance of SQL Server, create a login on the other instances of SQL Server. Use a CREATE LOGIN statement with the FROM WINDOWS clause.

   For more information, see Create a Login (SQL Server Database Engine.
2. Also, to ensure that the login user has access to the endpoint, use the GRANT statement to grant connect permissions on the endpoint to the login. Note that granting connect permissions to the endpoint is unnecessary if the user is an Administrator.

   For more information, see Grant a Permission to a Principal.

Example

Description

The following Transact-SQL example creates a SQL Server login for a user account named `Otheruser` that belongs to a domain called `Adomain`. The example then grants this user connect permissions to a pre-existing database mirroring endpoint named `Mirroring_Endpoint`.

Code

```
USE master;

GO

CREATE LOGIN [Adomain\Otheruser] FROM WINDOWS;

GO

GRANT CONNECT on ENDPOINT::Mirroring_Endpoint TO [Adomain\Otheruser];

GO
```

See Also

Overview of AlwaysOn Availability Groups (SQL Server)

Database Mirroring

Transport Security for Database Mirroring and AlwaysOn Availability Groups (SQL Server)

The Database Mirroring Endpoint (SQL Server)

Use Certificates for a Database Mirroring Endpoint (Transact-SQL)

To enable certificate authentication for database mirroring on a given server instance, the system administrator must configure each server instance to use certificates on both outbound and inbound connections. Outbound connections must be configured first.

Note
All mirroring connections on a server instance use a single database mirroring endpoint, and you must specify the authentication method of the server instance when you create the endpoint. Therefore, you can use only one form of authentication per server instance for database mirroring.

Configuring Outbound Connections

Follow these steps on each server instance that you are configuring for database mirroring:

1. In the **master** database, create a database master key.
2. In the **master** database, create an encrypted certificate on the server instance.
3. Create an endpoint for the server instance using its certificate.
4. Back up the certificate to a file and securely copy it to the other system or systems.

You must complete these steps for each partner and the witness, if there is one.

For more information, see Database Mirroring Endpoint.

Configuring Inbound Connections

Next, follow these steps for each partner that you are configuring for database mirroring. In the **master** database:

1. Create a login for the other system.
2. Create a user for that login.
3. Obtain the certificate for the mirroring endpoint of the other server instance.
4. Associate the certificate with the user created in step 2.
5. Grant CONNECT permission on the login for that mirroring endpoint.

If there is a witness, you must also set up inbound connections for it. This requires setting up logins, users, and certificates for the witness on both of the partners, and vice versa.

For more information, see How to: Allow Database Mirroring to Use Certificates for Inbound Connections (Transact-SQL).

Security

Unless you can guarantee that your network is secure, we recommend that you use encryption for database mirroring connections. For more information, see The Database Mirroring Endpoint.

When copying a certificate to another system, use a secure copy method. Be extremely careful to keep all of your certificates secure.

See Also

How to: Create a Database Master Key

CREATE MASTER KEY (Transact-SQL)

Database Mirroring Transport Security

Security and Protection (Database Engine)

The Database Mirroring Endpoint

Allow a Database Mirroring Endpoint to Use Certificates for Outbound Connections (Transact-SQL)

This topic describes the steps for configuring server instances to use certificates to authenticate outbound connections for database mirroring. Outbound connection configuration must be done before you can set up inbound connections.

Note
All mirroring connections on a server instance use a single database mirroring endpoint, and you must specify the authentication method of the server instance when you create the endpoint.

The process of configuring outbound connections, involves the following general steps:

1. In the **master** database, create a database Master Key.
2. In the **master** database, create an encrypted certificate on the server instance.
3. Create an endpoint for the server instance using its certificate.
4. Back up the certificate to a file and securely copy it to the other system or systems.

You must complete these steps for each partner and the witness, if there is one.

The following procedure describes these steps in detail. For each step, the procedure provides an example for configuring a server instance on a system named HOST_A. The accompanying Example section demonstrates the same steps for another server instance on a system named HOST_B.

Procedure

To configure server instances for outbound mirroring connections (On HOST_A)

1. On the **master** database, create the database Master Key, if none exists. To view the existing keys for a database, use the sys.symmetric_keys catalog view.

   To create the database Master Key, use the following Transact-SQL command:

   ```
   CREATE MASTER KEY ENCRYPTION BY PASSWORD =
   '<1_Strong_Password!>';
   ```

```
GO
```

Use a unique, strong password, and record it in a safe place.

For more information, see CREATE MASTER KEY (Transact-SQL) and How to: Create a Database Master Key.

2. In the **master** database, create an encrypted certificate on the server instance to use for its outbound connections for database mirroring.

   For example, to create a certificate for the HOST_A system.

   Important
   If you intend to use the certificate for more than one year, specify the expiry date in UTC time by using the EXPIRY_DATE option in your CREATE CERTIFICATE statement. Also, we recommend that you use SQL Server Management Studio to create a Policy-Based Management rule to alert you when your certificates are expiring. Using the Policy Management **Create New Condition** dialog box, create this rule on the **@ExpirationDate** field of the **Certificate** facet. For more information, see Administering Servers by Using Policy-Based Management and Securing SQL Server.

```
USE master;

CREATE CERTIFICATE HOST_A_cert

   WITH SUBJECT = 'HOST_A certificate for database mirroring',

   EXPIRY_DATE = '11/30/2013';

GO
```

   For more information, see CREATE CERTIFICATE (Transact-SQL).

   To view the certificates in the **master** database, you can use the following Transact-SQL statements:

```
USE master;

SELECT * FROM sys.certificates;
```

   For more information, see sys.certificates (Transact-SQL).

3. Ensure that the database mirroring endpoint exist on each of the server instances.

   If a database mirroring endpoint already exists for the server instance, you should reuse that endpoint for any other sessions you establish on the server instance. To determine whether a database mirroring endpoint exists on a server instance and to view its configuration, use the following statement:

```
SELECT name, role_desc, state_desc, connection_auth_desc,
encryption_algorithm_desc

   FROM sys.database_mirroring_endpoints;
```

   If no endpoint exists, create an endpoint that uses this certificate for outbound connections and that uses the certificate's credentials for verification on the other system. This is a server-wide endpoint that is used by all mirroring sessions in which the server instance participates.

For example, to create a mirroring endpoint for the example server instance on HOST_A.

```
CREATE ENDPOINT Endpoint_Mirroring

   STATE = STARTED

   AS TCP (

      LISTENER_PORT=7024

      , LISTENER_IP = ALL

   )

   FOR DATABASE_MIRRORING (

      AUTHENTICATION = CERTIFICATE HOST_A_cert

      , ENCRYPTION = REQUIRED ALGORITHM AES

      , ROLE = ALL

   );

GO
```

For more information, see CREATE ENDPOINT (Transact-SQL).

4. Back up the certificate and copy it to the other system or systems. This is necessary in order to configure inbound connections on the other system.

```
BACKUP CERTIFICATE HOST_A_cert TO FILE = 'C:\HOST_A_cert.cer';

GO
```

For more information, see BACKUP CERTIFICATE (Transact-SQL).

Copy this certificate using any secure method you choose. Be extremely careful to keep all of your certificates secure.

The example code in the preceding steps configure outbound connections on HOST_A.

You now need to perform the equivalent outbound steps for HOST_B. These steps are illustrated in the following Example section.

Example

Description

The following example demonstrates configuring HOST_B for outbound connections.

Code

```
USE master;

--Create the database Master Key, if needed.

CREATE MASTER KEY ENCRYPTION BY PASSWORD = '<Strong_Password_#2>';

GO
```

```
-- Make a certifcate on HOST_B server instance.
CREATE CERTIFICATE HOST_B_cert
   WITH SUBJECT = 'HOST_B certificate for database mirroring',
   EXPIRY_DATE = '11/30/2013';
GO
--Create a mirroring endpoint for the server instance on HOST_B.
CREATE ENDPOINT Endpoint_Mirroring
   STATE = STARTED
   AS TCP (
      LISTENER_PORT=7024
      , LISTENER_IP = ALL
   )
   FOR DATABASE_MIRRORING (
      AUTHENTICATION = CERTIFICATE HOST_B_cert
      , ENCRYPTION = REQUIRED ALGORITHM AES
      , ROLE = ALL
   );
GO
--Backup HOST_B certificate.
BACKUP CERTIFICATE HOST_B_cert TO FILE = 'C:\HOST_B_cert.cer';
GO
--Using any secure copy method, copy C:\HOST_B_cert.cer to HOST_A.
```

Comments

Copy the certificate to the other system using any secure method you choose. Be extremely careful to keep all of your certificates secure.

Important
After you set up outbound connections, you must configure inbound connections on each server instance for the other server instance or instances. For more information, see How to: Allow Database Mirroring to Use Certificates for Inbound Connections (Transact-SQL).

For information on creating a mirror database, including a Transact-SQL example, see Prepare a Mirror Database for Mirroring (SQL Server).

For a Transact-SQL example of establishing a high-performance mode session, see Example: Setting Up Database Mirroring Using Certificates.

Security

Unless you can guarantee that your network is secure, we recommend that you use encryption for database mirroring connections.

When copying a certificate to another system, use a secure copy method.

See Also

Choosing an Encryption Algorithm

Prepare a Mirror Database for Mirroring (SQL Server)

ALTER ENDPOINT (Transact-SQL)

Example: Setting Up Database Mirroring Using Certificates

The Database Mirroring Endpoint

Troubleshooting Database Mirroring Setup

Setting up an Encrypted Mirror Database

Allow a Database Mirroring Endpoint to Use Certificates for Inbound Connections (Transact-SQL)

This topic describes the steps for configuring server instances to use certificates to authenticate inbound connections for database mirroring. Before you can set up inbound connections, you must configure outbound connections on each server instance. For more information, see How to: Allow Database Mirroring to Use Certificates for Outbound Connections (Transact-SQL).

The process of configuring inbound connections, involves the following general steps:

1. Create a login for other system.
2. Create a user for that login.
3. Obtain the certificate for the mirroring endpoint of the other server instance.
4. Associate the certificate with the user created in step 2.
5. Grant CONNECT permission on the login for that mirroring endpoint.

If there is a witness, you must also set up inbound connections for it. This requires setting up logins, users, and certificates for the witness on both of the partners, and vice versa.

The following procedure describes these steps in detail. For each step, the procedure provides an example for configuring a server instance on a system named HOST_A. The accompanying Example section demonstrates the same steps for another server instance on a system named HOST_B.

Procedures

To configure server instances for inbound mirroring connections (on HOST_A)

1. Create a login for the other system.

   The following example creates a login for the system, HOST_B, in the **master** database of the server instance on HOST_A; in this example, the login is named `HOST_B_login`. Substitute a password of your own for the sample password.

```
USE master;
CREATE LOGIN HOST_B_login
   WITH PASSWORD = '1Sample_Strong_Password!@#';
GO
```

For more information, see CREATE LOGIN (Transact-SQL).

To view the logins on this server instance, you can use the following Transact-SQL statement:

```
SELECT * FROM sys.server_principals
```

For more information, see sys.server_principals (Transact-SQL).

2. Create a user for that login.

   The following example creates a user, `HOST_B_user`, for the login created in the preceding step.

   ```
   USE master;
   CREATE USER HOST_B_user FOR LOGIN HOST_B_login;
   GO
   ```

   For more information, see CREATE USER (Transact-SQL).

   To view the users on this server instance, you can use the following Transact-SQL statement:

   ```
   SELECT * FROM sys.sysusers;
   ```

   For more information, see sys.sysusers (Transact-SQL).

3. Obtain the certificate for the mirroring endpoint of the other server instance.

   If you have not already done so when configuring outbound connections, obtain a copy of the certificate for the mirroring endpoint of the remote server instance. To do this, back up the certificate on that server instance as described in How to: Allow Database Mirroring to Use Certificates for Outbound Connections (Transact-SQL). When copying a certificate to another system, use a secure copy method. Be extremely careful to keep all of your certificates secure.

   For more information, see BACKUP CERTIFICATE (Transact-SQL).

4. Associate the certificate with the user created in step 2.

   The following example, associates the certificate of HOST_B with its user on HOST_A.

   ```
   USE master;
   CREATE CERTIFICATE HOST_B_cert
      AUTHORIZATION HOST_B_user
      FROM FILE = 'C:\HOST_B_cert.cer'
   GO
   ```

   For more information, see CREATE CERTIFICATE (Transact-SQL).

To view the certificates on this server instance, use the following Transact-SQL statement:

```
SELECT * FROM sys.certificates
```

For more information, see sys.certificates (Transact-SQL).

5. Grant CONNECT permission on the login for the remote mirroring endpoint.

   For example, to grant permission on HOST_A to the remote server instance on HOST_B to connect to its local login—that is, to connect to `HOST_B_login`—use the following Transact-SQL statements:

   ```
   USE master;
   GRANT CONNECT ON ENDPOINT::Endpoint_Mirroring TO [HOST_B_login];
   GO
   ```

   For more information, see GRANT Endpoint Permissions (Transact-SQL).

This completes setting up certificate authentication for HOST_B to log in to HOST_A.

You now need to perform the equivalent inbound steps for HOST_A on HOST_B. These steps are illustrated in the inbound portion of the example in the Example section, below.

Example

Description

The following example demonstrates configuring HOST_B for inbound connections.

Note

This example uses a certificate file containing the HOST_A certificate that is created by a code snippet in How to: Allow Database Mirroring to Use Certificates for Outbound Connections (Transact-SQL).

Code

```
USE master;
--On HOST_B, create a login for HOST_A.
CREATE LOGIN HOST_A_login WITH PASSWORD = 'AStrongPassword!@#';
GO
--Create a user, HOST_A_user, for that login.
CREATE USER HOST_A_user FOR LOGIN HOST_A_login
GO
--Obtain HOST_A certificate. (See the note
--   preceding this example.)
--Asscociate this certificate with the user, HOST_A_user.
CREATE CERTIFICATE HOST_A_cert
   AUTHORIZATION HOST_A_user
```

```
   FROM FILE = 'C:\HOST_A_cert.cer';
GO
--Grant CONNECT permission for the server instance on HOST_A.
GRANT CONNECT ON ENDPOINT::Endpoint_Mirroring TO HOST_A_login
GO
```

Comments

If you intend to run in high-safety mode with automatic failover, you must repeat the same set up steps to configure the witness for outbound and inbound connections.

For information on creating a mirror database, including a Transact-SQL example, see Prepare a Mirror Database for Mirroring (SQL Server).

For a Transact-SQL example of establishing a high-performance mode session, see Example: Setting Up Database Mirroring Using Certificates.

Security

When copying a certificate to another system, use a secure copy method. Be extremely careful to keep all of your certificates secure.

See Also

Database Mirroring Transport Security

GRANT Endpoint Permissions (Transact-SQL)

Setting up an Encrypted Mirror Database

The Database Mirroring Endpoint

Troubleshooting Database Mirroring Setup