Sql Server Mirroring Library

Automatic handling of mirroring in code and both check of need configuration and the setup of that. It is expected to be used from a host program or service that can load configuration, knows server state and run the library functions. The library handles multi-mirrored servers (servers that handles multiple mirrored databases for one application and all are needed for operation).

Contents

[1 Figures 3](#_Toc446925661)

[2 Vocabulary/Definitions 4](#_Toc446925662)

[3 Limitations 5](#_Toc446925663)

[4 Installation 5](#_Toc446925664)

[4.1 Prerequisites 5](#_Toc446925665)

[4.1.1 SMO 5](#_Toc446925666)

[4.1.2 Sql Server 2014 5](#_Toc446925667)

[4.1.3 Existing database on the Primary Role server 5](#_Toc446925668)

[4.2 First run 6](#_Toc446925669)

[5 Main Achitecture 6](#_Toc446925670)

[6 Primary Role States 7](#_Toc446925671)

[6.1 Startup State [Degraded] 7](#_Toc446925672)

[6.1.1 Next states 7](#_Toc446925673)

[6.1.2 Action: Instance Ready for Mirroring 7](#_Toc446925674)

[6.1.3 Primary Role 7](#_Toc446925675)

[6.1.4 Individual databases 8](#_Toc446925676)

[6.1.5 Change state 10](#_Toc446925677)

[6.2 Running State [Full Mirroring] 10](#_Toc446925678)

[6.2.1 Next states 10](#_Toc446925679)

[6.3 Forced Running State [Degraded] 10](#_Toc446925680)

[6.3.1 Next states 10](#_Toc446925681)

[6.3.2 Use 10](#_Toc446925682)

[6.4 Shutting Down State [Degraded] 10](#_Toc446925683)

[6.4.1 Next states 10](#_Toc446925684)

[6.4.2 Use 10](#_Toc446925685)

[6.5 Shutdown State [Degraded] 10](#_Toc446925686)

[6.5.1 Next states 10](#_Toc446925687)

[6.6 Maintenance State [Full Mirroring] 10](#_Toc446925688)

[6.6.1 Next states 10](#_Toc446925689)

[6.6.2 Use 10](#_Toc446925690)

[6.7 Manual Failover State [Full Mirroring] 11](#_Toc446925691)

[6.7.1 Next states 11](#_Toc446925692)

[6.7.2 Use 11](#_Toc446925693)

[6.8 Force Manual Failover State [Degraded] 11](#_Toc446925694)

[6.8.1 Next states 11](#_Toc446925695)

[6.8.2 Use 11](#_Toc446925696)

[7 Secondary Role States 11](#_Toc446925697)

[7.1 SecondaryShutdown State [Degraded] 11](#_Toc446925698)

[7.1.1 Next states 11](#_Toc446925699)

[7.2 Secondary Startup State [Degraded] 11](#_Toc446925700)

[7.2.1 Next states 11](#_Toc446925701)

[7.2.2 Action: Instance Ready for Mirroring 11](#_Toc446925702)

[7.2.3 Action: Setup of Secondary Role 11](#_Toc446925703)

[7.3 Individual databases 12](#_Toc446925704)

[7.3.1 Action: Add Database to Mirroring 12](#_Toc446925705)

[7.3.2 Action: Remove Database from Mirroring 13](#_Toc446925706)

[7.3.3 Change state 13](#_Toc446925707)

[7.4 Secondary Running State [Full Mirroring] 14](#_Toc446925708)

[7.4.1 Next states 14](#_Toc446925709)

[7.4.2 Use 14](#_Toc446925710)

[7.4.3 Database state 14](#_Toc446925711)

[7.5 SecondaryRunning No Primary State [Degraded] 14](#_Toc446925712)

[7.5.1 Next states 14](#_Toc446925713)

[7.5.2 Use 14](#_Toc446925714)

[7.5.3 Database state 14](#_Toc446925715)

[7.6 Secondary Shutting Down State [Degraded] 14](#_Toc446925716)

[7.6.1 Next states 14](#_Toc446925717)

[7.6.2 Use 14](#_Toc446925718)

[7.7 Secondary Maintenance State [Full Mirroring] 14](#_Toc446925719)

[7.7.1 Next states 14](#_Toc446925720)

[7.7.2 Use 14](#_Toc446925721)

[7.7.3 Database state 15](#_Toc446925722)

[7.8 Secondary Manual Failover State [Full Mirroring] 15](#_Toc446925723)

[7.8.1 Next states 15](#_Toc446925724)

[7.8.2 Use 15](#_Toc446925725)

[7.9 Secondary Force Manual Failover State [Degraded] 15](#_Toc446925726)

[7.9.1 Next states 15](#_Toc446925727)

[7.9.2 Use 15](#_Toc446925728)

[8 Troubleshooting 15](#_Toc446925729)

[8.1 Basic Introduction 15](#_Toc446925730)

[8.2 Basic Monitoring 15](#_Toc446925731)

[8.2.1 TSQL 16](#_Toc446925732)

[8.2.2 Database Mirroring Monitoring 17](#_Toc446925733)

[8.3 Startup Problems 17](#_Toc446925734)

[8.3.1 Stops at directory creation 17](#_Toc446925735)

[8.3.2 Stops at share setup 18](#_Toc446925736)

[8.3.3 Stops under creation of endpoint 19](#_Toc446925737)

[8.3.4 Stops when trying to access remote server 19](#_Toc446925738)

[8.3.5 Secondary Role server does not exit SECONDARY\_STARTUP\_STATE 23](#_Toc446925739)

[8.3.6 Reports unknown server name in Log 23](#_Toc446925740)

[8.4 Disable Mirroring from database and redo 23](#_Toc446925741)

[8.5 Need to manually failover databases in SSMS 24](#_Toc446925742)

[8.6 Delete mirrored database 24](#_Toc446925743)

[8.7 Mirroring problems 24](#_Toc446925744)

[8.8 Switch-over for multiple databases 24](#_Toc446925745)

# Figures

[Figure 1: State diagram. In code the states have prepended the Server Role and appended \_STATE which fx makes FORCED\_RUNNING into PRIMARY\_FORCED\_RUNNING\_STATE. 6](#_Toc446925746)

[Figure 2: Action: Primary Role – Create Dir/Share 8](#_Toc446925747)

[Figure 3: Action: Backup database and move to remote share 9](#_Toc446925748)

[Figure 4: Action: Secondary Role – Create Dir/Share 12](#_Toc446925749)

[Figure 5: Action: Move backup from remote directory and restore database 13](#_Toc446925750)

[Figure 6: Example result from Primary Role Server displaying some of the inner states 16](#_Toc446925751)

[Figure 7: Example result from Secondary Role Server displaying some of the inner states 16](#_Toc446925752)

[Figure 8: Database Mirroring Monitor (SSMS buildin monitor) 17](#_Toc446925753)

[Figure 9: Startup Problems: Stops at directory creation 18](#_Toc446925754)

[Figure 10: Startup Problems: Stops at share creation - Share Tab 18](#_Toc446925755)

[Figure 11: Startup Problems: Stops at share creation - Advanced Sharing 19](#_Toc446925756)

[Figure 12: Startup Problems: Stops at share creation – Permissions 19](#_Toc446925757)

[Figure 13: Startup Problems: Stops at failover - Select firewall 20](#_Toc446925758)

[Figure 14: Startup Problems: Stops at failover - Firewall 20](#_Toc446925759)

[Figure 15: Startup Problems: Stops at failover – Inbound Rules 20](#_Toc446925760)

[Figure 16: Startup Problems: Stops at failover – Port 21](#_Toc446925761)

[Figure 17: Startup Problems: Stops at failover – Port Number 21](#_Toc446925762)

[Figure 18: Startup Problems: Stops at failover – Action Allow 22](#_Toc446925763)

[Figure 19: Startup Problems: Stops at failover – Active Profile 22](#_Toc446925764)

[Figure 20: Startup Problems: Stops at failover – Name firewall rule 23](#_Toc446925765)

[Figure 21: Script to correct reporting from @@SERVERNAME 23](#_Toc446925766)

[Figure 22: Remove mirroring and delete database in TSQL 24](#_Toc446925767)

# Vocabulary/Definitions

Documentation is tried kept in Microsoft Sql Server Mirroring vocabulary.

MirrorLib: Reference to the SQL Server Mirroring Library.

SMO: Default short-name for Sql Server Management Objects. SMO is used to manage Sql Server from code and automatically handles starting up and shutting down connection when needed.

SSMS: Sql Server Management Studio which is normally installed when the Sql Server is installed. It is expected to be running on the server to verify states. It is not needed for MirrorLib functionality.

Database Server Instance: Named installation version of the SQL Server software. First installation typically have the default name SQLSERVER.

Principal Role: Database Role in mirroring which is the active server. This is the server that the external applications connect to and run queries. This is the database that is backed up to truncate the log and make recovery faster.

Mirror Role: Database Role in mirroring which is the mirroring server. The database is the one ready to fail over and is in restoring state to make additions faster.

Multi-mirror server: Server that has several mirrored databases that need to switch over at the same time.

Primary Role: Server Role where the server runs its mirrored databases in Principal Role. When the server shifts role from Primary to Secondary all the mirrored databases shift role from Principal to Mirrored.

Secondary Role: Server Role where the server runs its mirrored database in Mirror Role. When the server shifts role from Secondary to Primary all the mirrored databases shift role from Mirrored to Principal.

Library Host: The program or service that is running the library. It is required to be able to evaluate if the server runs in Primary Role or Secondary Role. The action in the library can be handled via scripting via the console test application (which is the running as an intermediary Library Host) or via direct use of the MirrorLib in code. The Library Host needs to be able to identify a server role and possible shifts and send that information to the MirrorLib.

Action: A task that handled by MirrorLib and communicated to the log so that in a debugging situation so it is possible to see what the server is doing. An action can consist of sub-actions if several things need to be done in an action. Actions on individual databases are done as sub-threads as they are able to be done in parallel.

Switch-over: The change of server role between primary/secondary.

Degraded state: State where the mirrored databases are not ready for another failover. The mirrored databases might be running. This state can be due to a switch-over or change in configuration.

# Limitations

1. Automatic switchover with witness server has not been implemented
2. Does not have its own monitoring service that can change the role of the server.
3. It is expected that mirrored databases on the Sql Server instance is surposed to be handled by MirrorLib.
4. A database name is only allowed to consist of letters, numbers and \_.
5. SHUTDOWN\_STATE calls Environment.Exit(0) and thus shuts down the application which is running MirrorLib which might be correct for some applications but limit others.
6. Tested on
   1. Sql Server 2014 Standard Edition
   2. Windows Server 2012 R2 Standard Edition
   3. AD User for running the mirroing that has rights to
      1. backup and restore databases.
      2. setting up mirroring which is access to the master database on each server.
      3. has the right to create endpoints on the database server instance \*.
      4. has the right to start services \*.
      5. creating needed directories \*.
      6. setting security for directories \*.
      7. creation of shares \*.
      8. setting security for shares \*.

\* This can be handled manually if the installer does not want to give the rights to the user. The state of each is tested on startup and startup fails if not configured.

# Configuration

MirrorLib has a configuration for the instance and per database for mirroring. The configuration has been made with the fail fast checks on input so it fails in assigning for configuration instead of in runtime.

## ConfigurationForInstance

The configuration consists of a number of settings.

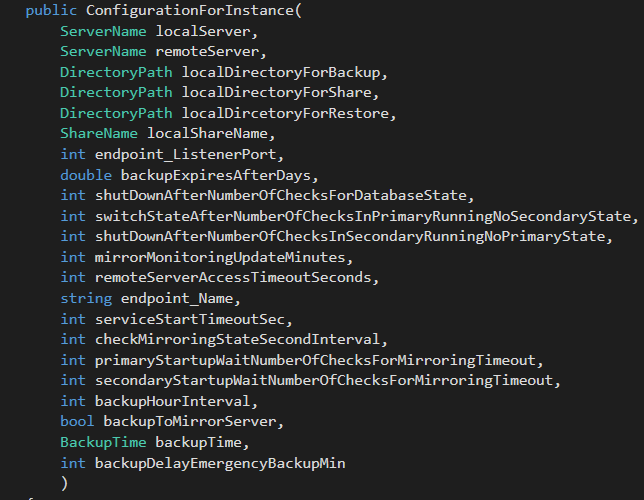


Figure : ConfigurationForInstance constructor

### ServerName localServer

Name of the local server. The ServerName class is used to make sure the name follows the expected format.

### ServerName remoteServer

Name of the remote mirrored server.

### DirectoryPath localDirectoryForBackup

Directory where the backup is generated and will be stored until they are too old.

### DirectoryPath localDirectoryForShare

Directory that will be used as the share root when the share is created.

### DirectoryPath localDirectoryForRestore

Databases are restored from this path into the database. It is where the mirror server will transport the newest backup file for restoring.

### ShareName localShareName

Name for the local share. The ShareName class is validating that the shareName is valid.

### int endpoint\_ListenerPort

The listener port that needs to be open between the servers for the mirroring communication. Setup for Microsoft Firewall is described in 9.3.4

### double backupExpiresAfterDays

The number of days a backup should be valid and should stay on the drives.

### int shutDownAfterNumberOfChecksForDatabaseState

NumberOfChecks that a database is allowed to be in an unexpected state. When the databases are in a wrong state the MirrorLib should signal that by switching to SHUTTING\_DOWN\_STATE. If 0 is specified MirrorLib will not shift to SHUTTING\_DOWN\_STATE. Suggestion is to have this at 12 which is one minute with a database in wrong state.

### int switchStateAfterNumberOfChecksInPrimaryRunningNoSecondaryState

NumberOfChecks that a server is allowed to run in PRIMARY\_RUNNING\_NO\_SECONDARY\_STATE after which state is shifted to PRIMARY\_FORCED\_RUNNING\_STATE for which a switch back to PRIMARY\_RUNNING\_STATE is not allowed. If 0 is specified the countdown is disabled. Suggestion is to use 0.

### int shutDownAfterNumberOfChecksInSecondaryRunningNoPrimaryState

NumberOfChecks that MirrorLib runs in SECONDARY\_RUNNING\_NO\_PRIMARY\_STATE before it switches to SECONDARY\_SHUTTING\_DOWN\_STATE. If 0 is specified the countdown is disabled. Suggestion is to use 12 which is 1 minute. To take over the Primary Role server MirrorLib needs to restart in Primary mode.

### int mirrorMonitoringUpdateMinutes

Number of minutes where SSMS Database Mirroring Mionitor is updating. Suggestion is to use the default 1.

### int remoteServerAccessTimeoutSeconds

Seconds for the remote server connection to timeout. Unless there is a high latency on the network this should be low. Suggestion is to use 1.

### string endpoint\_Name

Name of the endpoint for mirroring. Call it something saying as “Mirroring\_Endpoint”.

### int serviceStartTimeoutSec

Timeout for the startup of services when trying to start or stop services when setting up MirrorLib. Suggestion is to use 120 seconds.

### int checkMirroringStateSecondInterval

The interval for check of server state (you can call it MirrorLib’s heartbeat) which the NumberOfChecks is related to. Suggestion is to have it at 5 seconds.

### int primaryStartupWaitNumberOfChecksForMirroringTimeout

NumberOfChecks where server waits in PRIMARY\_STARTUP\_STATE for the Mirror Server to be ready for mirroring. Suggestion is to use 60 which is 5 minutes as it takes some time to restore databases and after that start the mirroring.

### int secondaryStartupWaitNumberOfChecksForMirroringTimeout

NumberOfChecks where server waits in SECONDARY\_STARTUP\_STATE for the Primary role server to have delivered the databases for restoring. Suggestion is to use either the same as the primary evrsion or using half of it.

### int backupHourInterval

The interval after the backup time that the next daily backup should be made. Suggestion is to use 24 to have a backup every 24 hours.

### bool backupToMirrorServer

Specifies if the daily backup should be tried to transport for the mirror server. Suggestion is to use true in case of a total failure of the primary server a backup is at least there for restoring.

### BackupTime backupTime

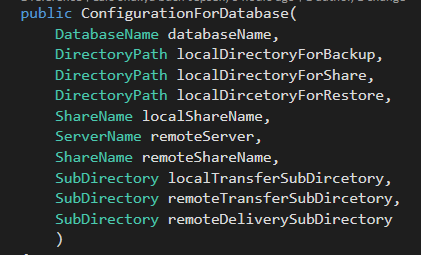
The time of day for start of the daily backup using backupHourInterval for next backup after the first. Suggestion is to place this where there are the least pressure on the server.

### int backupDelayEmergencyBackupMin

When the primary server enters PRIMARY\_RUNNING\_NO\_SECONDARY\_STATE a backup is triggered in case there are need for one to reestablish the secondary server.

## ConfigurationForDatabase

Configuration consists of several settings



### DatabaseName databaseName

The database that is surposed to be mirrored. DatabaseName class has the validation for expected name for databases.

### DirectoryPath localDirectoryForBackup

Expected to be the same as ConfigurationForInstance.

### DirectoryPath localDirectoryForShare

Expected to be the same as ConfigurationForInstance.

### DirectoryPath localDirectoryForRestore

Expected to be the same as ConfigurationForInstance.

### ShareName localShareName

Expected to be the same as ConfigurationForInstance.

### ServerName remoteServer

Expected to be the same as ConfigurationForInstance.

### ShareName remoteShareName

It is good to keep the shares either named the same or following the same pattern but it is not needed for the MirrorLib to work.

### SubDirectory localTransferSubDirectory

LocalTransfer is the share subdirectory expected to be on the server. It is used to transport backup to the mirrored server in directory RemoteTransfer.

### SubDirectory remoteTransferSubDirectory

RemoteTransfer is the share subdirectory expected on the server. It is used to transport backup from the primary server to this.

### SubDirectory remoteDeliverySubDirectory

RemoteDelivery is the share subdirectory expected on the server. It is used by the primary to transport backup from RemoteTransfer to this.

# Installation

## Prerequisites

### SMO

Servers need to have “Shared Management Objects” (SMO) installed which is included for Sql Server 2014 as redistributable packages in the sub-directory “External” for both 32bit and 64bit.

### Sql Server 2014

The database instance needs to have been installed before trying to setup mirroring.

### Existing database on the Primary Role server

The databases need to be existing on the Primary Role server and the Secondary Role server is being set up via restore of a backup from Primary Role server.

## First run

First run should be done with elevated privileges as needed directories and shares are created. This can though be done manually creating directories for LocalBackup, LocalRestore and LocalShare with full control for Authenticated Users and by sharing with Authenticated Users which has full control to the share (see 9.3.1 and 9.3.2).

# Main Achitecture

The Sql Server Mirroring Library (MirrorLib) is mainly a state machine that has both automatic and manual transitions between the different states. The reason for the use is to effectively communicate the state of the mirroring of the server. It is done through code using Sql Server Management Objects (SMO) so partly it can be run easily from scripts or code and partly to isolate from the complexities of mirroring in a transparent way. The initial steps to start mirroring and to setup later if a server had to be reinstalled.

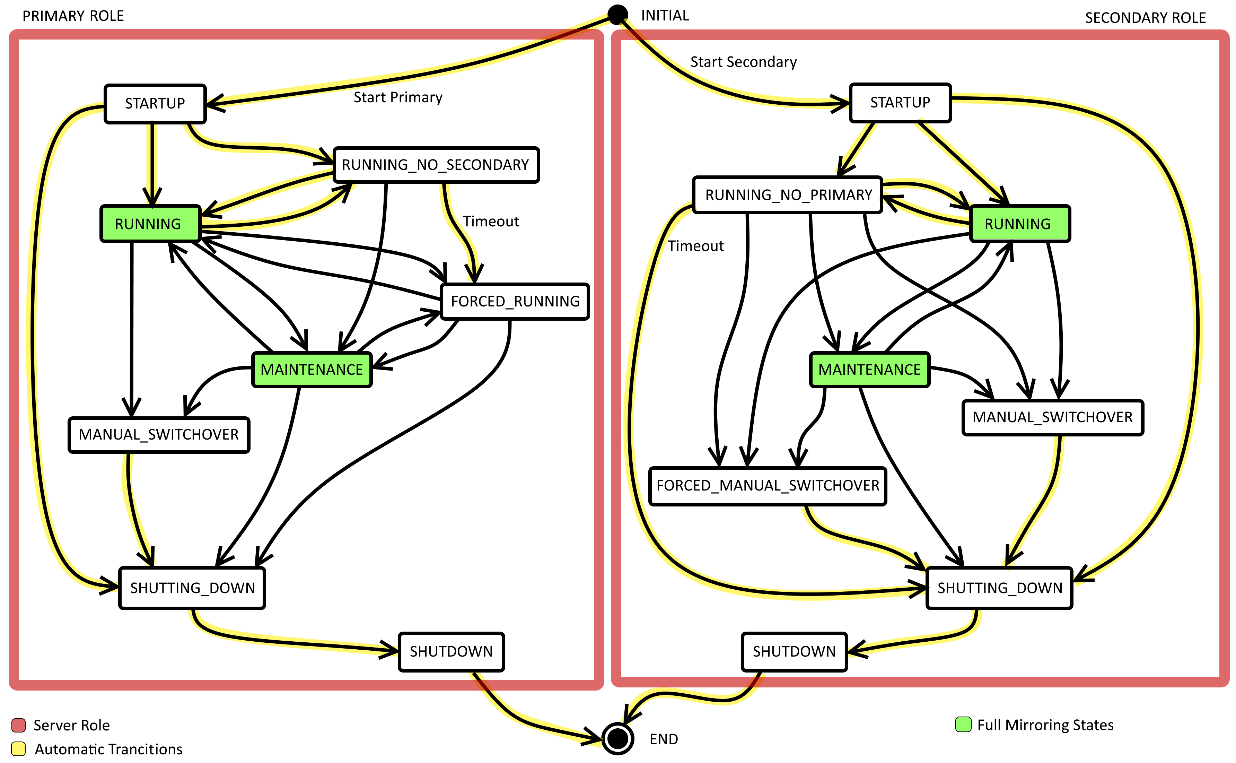


Figure 2: State diagram. In code the states have prepended the Server Role and appended \_STATE which fx makes FORCED\_RUNNING into PRIMARY\_FORCED\_RUNNING\_STATE.

The main handling in MirrorLib is done in the SqlServerInstance class where the instance needs to be running all the time to be able to report on the server and switch states.

The MirrorLib has methods with the naming standard of

* prepended **Information\_** when the method supplies information only. These methods are safe to call during runtime without changing states etc.
* prepended **Action\_** when the method modifies data or states. These methods are for modifying states and data so they are not safe/free to call.

To use the code some things need to be done:

* Handle creation of **ConfiguredDatabaseForMirroring** which are the configuration object expected by MirrorLib.
* Call to **Action\_StartPrimary()** or **Action\_StartSecondary()** when monitoring service needs MirrorLib to run with that specific Server Role from startup. If the MirrorLib needs to be switched while running calls to . If the Server Role is not the one it has been used the server will shift to that Server Role during INITIAL\_STATE if possible using a manual switchover akin in MANUAL\_SWITCHOVER\_STATE and if not successful will enter the SHUTTING\_DOWN\_STATE.
* Implementation of the **ILogger** interface to enable logging from MirrorLib.
* Timed calls to the method **Action\_CheckServerState()** in where much of the state checking is done. The suggestion is to call the method every second as it is used to check the server state and make the automatic transitions between the states.
* Ability to call methods at runtime for manual handling.

# Primary Role States

## Startup State [Degraded]

Startup is different depending on Server Role but have a common check for if the Sql Server Instance is ready for mirroring via **Action: Instance Ready form Mirroring**.

### Next states

Can switch to **Shutting Down State** or **Running State**.

### Action: Instance Ready for Mirroring

This action consists of several sub-actions which are run in sequence.

#### Action: Ready shares and directories

### Primary Role

#### Action: Setup of Primary Role

The action consist of several sub-actions. They are listed in the sequence they are called.

##### Action: Primary Role – Create Dir/Share

The process creates local directories, creates local share and checks access to directories locally via aread/write test. The read/write test is done by test creating a file to the location, reading it back and deleting it afterwards. In the configuration the different folders are names. Local share is “Share”. Local directories are “Local Share Directory”, “Local Restore”, “Local Backup” which both are absolute paths and “Local Transfer”, “Remote Transfer” and “Remote Delivery” which are placed under “Local Share Directory”. Read/write access to remote share “Remote Share” sub-directories “Local Transfer”, “Remote Transfer” and “Remote Delivery” are tested. Failure might be because of missing setup on Secondary Role server so if the read/write test fails it is noted as a warning in the log to avoid a “the hen or the egg” problem.

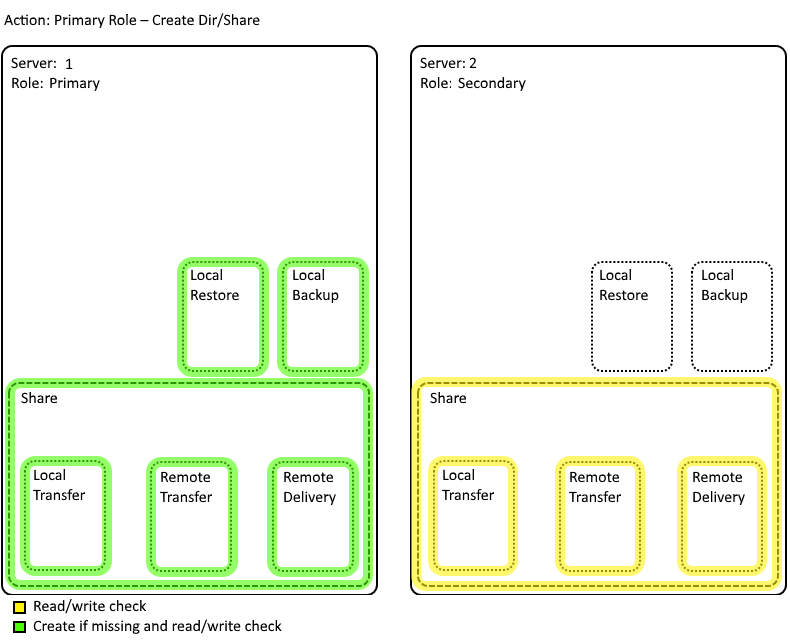


Figure 3: Action: Primary Role – Create Dir/Share

### Individual databases

This part is mostly the same for both server Roles. Where they differs are in the Backup/Restore action.

If a database from the configuration is identified as being not set up for mirroring then the **Action: Add Database to Mirroring** is called. If a database on the instance database is identified as being set up for mirroring but is not in the configuration then the **Action: Remove Database from Mirroring**. Each of these actions are handled in a sub-thread as they can be run in parallel.

#### Action: Add Database to Mirroring

This action consist of a few sub-actions which is listed in the sequence they are done.

##### Create local database sub-directories

Create local database sub-directories for each local directory with database name to host the specific databases.

##### Primary Role server only

###### Action: Backup database and move to remote share

Creates a backup of the Principal database for setting up the Mirror database on the Secondary Role server if needed.

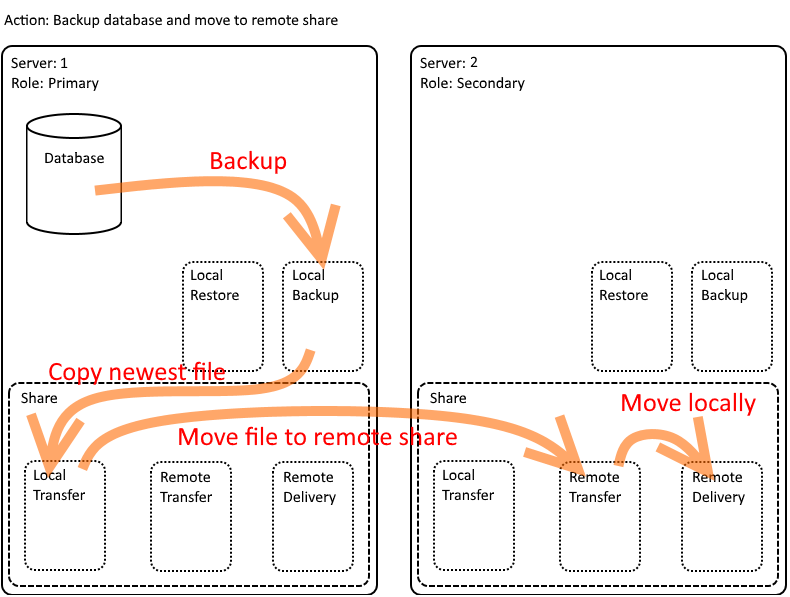


Figure 4: Action: Backup database and move to remote share

Action: Backup

A backup is made into the “Local Backup” sub-directory with the database name.

Action: Copy Backup to Local Transfer

Clears the “Local Transfer” database sub-directory and copies the newest Backup from “Local Backup” sub-directory to “Local Transfer” sub-directory.

Action: Move Backup to Remote Transfer

Clears the “Remote Transfer” database sub-directory and moves the newest Backup from “Local Transfer” sub-directory to “Remote Transfer” sub-directory on the “Remote Server”. If this fails it is only written to the log as warning to avoid “the hen and the egg” problem when setting up the servers where the Secondary Role server is not available because it is not set up or have failed.

Action: Move Backup to Remote Delivery

Clears the “Remote Delivery” database sub-directory and moves the newest Backup from “Remote Transfer” sub-directory to “Remote Delivery” sub-directory both on the “Remote Server”. This step is because a move locally on a shared drive is guaranteed to be atomic.

##### Action: Check Readiness for Mirroring

Checks database for Full Recovery Model and active Service Broker.

##### Action: Activate Mirroring on Database

1. Shifts database to full recovery mode if needed.
2. Activates Service Broker if needed.
3. Sets up Endpoint for Mirroring

#### Action: Remove Database from Mirroring

This action consist of a few sub-actions which is listed in the sequence they are done.

##### Action: Disable Mirroring for Database.

##### Action: Remove Endpoint

##### Action: Remove database specific sub-directories

### Change state

Change into **Running State** if everything has run correct and into **Shutting Down State** if some steps fails.

## Running State [Full Mirroring]

### Next states

Can switch to **Shutting Down State**, **Maintenance State**, **Forced Running State**, **Manual Failover State** or **Force Manual Failover State**.

## Forced Running State [Degraded]

### Next states

Can switch to **Shutting Down State**, **Maintenance State** and **Running State**.

### Use

Used to force the Primary Role server to run without the ability to use the **Manual Failover State** or **Force Manual Failover State**, which will move it into **Shutting Down State** instead. This is used if the Secondary Role server is running in **Maintenance State** for upgrades.

## Shutting Down State [Degraded]

### Next states

Can switch to **Shutdown State**.

### Use

Automatic state when preparing to shut down. Shifts automatic to **Shutdown State** when finished with preparations.

## Shutdown State [Degraded]

### Next states

Can switch to **Startup State** or **Maintainance State**.

## Maintenance State [Full Mirroring]

As such this state is as Running State but it is expected that external services are disabled. When shifted to this state from Running State on Primary Role server a backup is triggered on each of the mirrored databases.

### Next states

Can switch to **Shutdown State**, **Startup State**, **Manual Failover State** or **Force Manual Failover State**.

### Use

The state is used to update software which is always done on the Secondary Role server unless the database schema and data needs to be updated in which case use Primary Role server for update as there is no access to the database. If a database update fails all databases need to be restored. The benefit of updating on the Secondary Role server is that it minimizes downtime of the database as the Primary Role server can be left in **Forced Running State** where it cannot switch over. When the Secondary Role server has been upgraded switch over to the other server so it gets into the Secondary Role so it can be upgraded.

## Manual Failover State [Full Mirroring]

### Next states

Can switch to **Shutting Down State**, **Maintenance State** or **Running State**.

### Use

Used to switch server between Primary and Secondary Role and manually fail over servers

## Force Manual Failover State [Degraded]

### Next states

Can switch to **Shutting Down State**.

### Use

The use if this state is if something makes it ok to switch-over and allowing data loss in the process. This is a last resort switch-over. When command to force manual switch-over has been sent it automatically switches to **Shutting Down State**.

# Secondary Role States

## SecondaryShutdown State [Degraded]

### Next states

Can switch to **Secondary Startup State**.

## Secondary Startup State [Degraded]

Startup is different depending on Server Role but have a common check for if the Sql Server Instance is ready for mirroring via **Action: Instance Ready form Mirroring**.

### Next states

Can switch to **Secondary Shutting Down State**, **Secondary Running State** or **Secondary Running No Primary State**.

### Action: Instance Ready for Mirroring

This action consists of several sub-actions which are run in sequence.

#### Action: Ready shares and directories

### Action: Setup of Secondary Role

The action consist of several sub-actions. They are listed in the sequence they are called.

#### Action: Secondary Role – Create Dir/Share

The process creates local directories, creates local share and checks access to directories locally via a read/write test. The read/write test is done by test creating a file to the location, reading it back and deleting it afterwards. In the configuration the different folders are names. Local share is “Share”. Local directories are “Local Share Directory”, “Local Restore”, “Local Backup” which both are absolute paths and “Local Transfer”, “Remote Transfer” and “Remote Delivery” which are placed under “Local Share Directory”. Read/wirte access to remote share “Remote Share” sub-directories “Local Transfer”, “Remote Transfer” and “Remote Delivery” are tested. If the read/write test fails the startup fails as the connection is needed for mirroring.

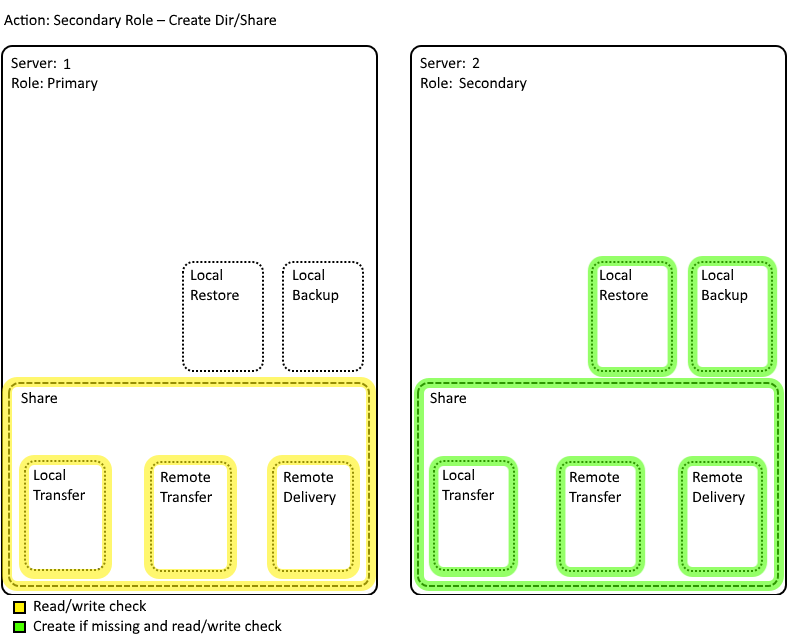


Figure 5: Action: Secondary Role – Create Dir/Share

## Individual databases

This part is mostly the same for both server Roles. Where they differs are in the Backup/Restore action.

If a database from the configuration is identified as being not set up for mirroring then the **Action: Add Database to Mirroring** is called. If a database on the instance database is identified as being set up for mirroring but is not in the configuration then the **Action: Remove Database from Mirroring**. Each of these actions are handled in a sub-thread as they can be run in parallel.

### Action: Add Database to Mirroring

This action consist of a few sub-actions which is listed in the sequence they are done.

#### Create local database sub-directories

Create local database sub-directories for each local directory with database name to host the specific databases.

#### Action: Move backup from remote directory and restore database

Check the existence of a Backup on remote share.

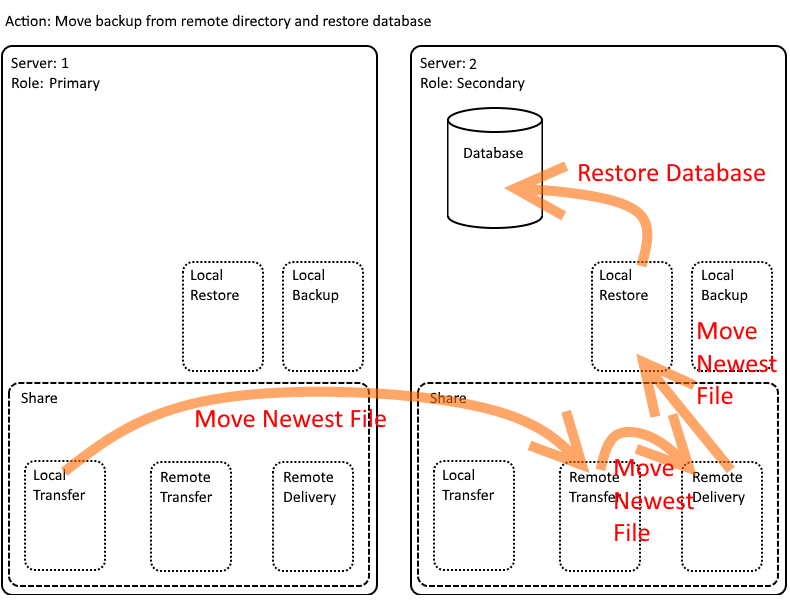


Figure 6: Action: Move backup from remote directory and restore database

##### Action: Check the newest Backup

Checks the remote share “Local Transfer” sub-directory, the local “Remote Transfer”, “Remote Delivery” and “Local Restore” sub-directories for the newest Backup.

##### Action: Delete all other files than the newest Backup file

Deletes all other files than the latest Backup file.

##### Action: Move newest file to Local Restore

Move newest file to “Remote Delivery” sub-directory in steps from the remote share “Local Transfer” sub-directory via the local “Remote Transfer” via the local “Remote Delivery” sub-directory.

##### Action: Restore database Backup from Local Restore

Action: Restore database Backup from “Local Restore”

#### Action: Check Readiness for Mirroring

Checks database for Full Recovery Model and active Service Broker.

#### Action: Activate Mirroring on Database

1. Shifts database to full recovery mode if needed.
2. Activates Service Broker if needed.
3. Sets up Endpoint for Mirroring

### Action: Remove Database from Mirroring

This action consist of a few sub-actions which is listed in the sequence they are done.

#### Action: Disable Mirroring for Database.

#### Action: Remove Endpoint

#### Action: Remove database specific sub-directories

### Change state

* Change into **Secondary Running State** if everything has run correct and has contact with Primary Role server
* if no contact with Primary Role server **Secondary Running No Primary State**
* if some steps fails into **Secondary Shutting Down State**.

## Secondary Running State [Full Mirroring]

### Next states

Can switch to **Secondary Shutting Down State**, **Secondary Maintenance State**, **Secondary Manual Failover State, Secondary Running No Primary State** or **Secondary Force Manual Failover State**.

### Use

Main mode for Secondary Role server

### Database state

All mirrored databases are in restoring state

## SecondaryRunning No Primary State [Degraded]

### Next states

Can switch to **Secondary Shutting Down State**, **Secondary Force Manual Failover State** or **Secondary Running State**.

### Use

A waiting state to see if connection to Primary Role server comes up again

### Database state

All mirrored databases are in restoring state

## Secondary Shutting Down State [Degraded]

### Next states

Can switch to **Secondary Shutdown State**.

### Use

Automatic state when preparing to shut down. Shifts automatic to **Secondary Shutdown State** when finished with preparations.

## Secondary Maintenance State [Full Mirroring]

As such this state is as Secondary Running State but it is expected that external services are disabled.

### Next states

Can switch to **Secondary Shutting Down State**, **Secondary Maintenance State**, **Secondary Manual Failover State, Secondary Running State** or **Secondary Force Manual Failover State**.

### Use

The state is used to update software which is always done on the Secondary Role server unless the database schema and data needs to be updated in which case use Primary Role server for update as there is no access to the database. If a database update fails all databases need to be restored. The benefit of updating on the Secondary Role server is that it minimizes downtime of the database as the Primary Role server can be left in **Primary Forced Running State** where it cannot switch over. When the Secondary Role server has been upgraded switch over to the other server so it gets into the Secondary Role so it can be upgraded.

### Database state

All mirrored databases are in restoring state

## Secondary Manual Failover State [Full Mirroring]

### Next states

Can switch to **Secondary Shutting Down State**.

### Use

Used to switch server between Primary and Secondary Role and manually fail over servers

## Secondary Force Manual Failover State [Degraded]

### Next states

Can switch to **Secondary Shutting Down State**.

### Use

The use if this state is if something makes it ok to switch-over and allowing data loss in the process. This is a last resort switch-over. When command to force manual switch-over has been sent it automatically switches to **Shutting Down State**.

# Troubleshooting

## Basic Introduction

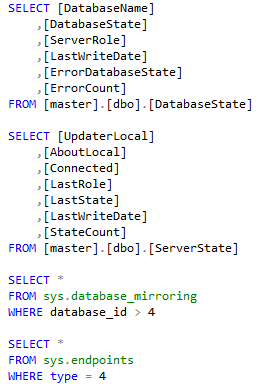
There are basic introductions on what the technology is.

Source (Videos): [Mirroring explanation](http://improve.dk/sql-server-mirroring-a-practical-approach/)

## Basic Monitoring

What happened in the database when mirroring and script startup can be monitored using TSQL or SSMS Database Mirroring Monitor (via right-clicking on database in SSMS on the Primary Role Server and selecting Database Mirroring Monitor)

### TSQL



#### Primary Role Server

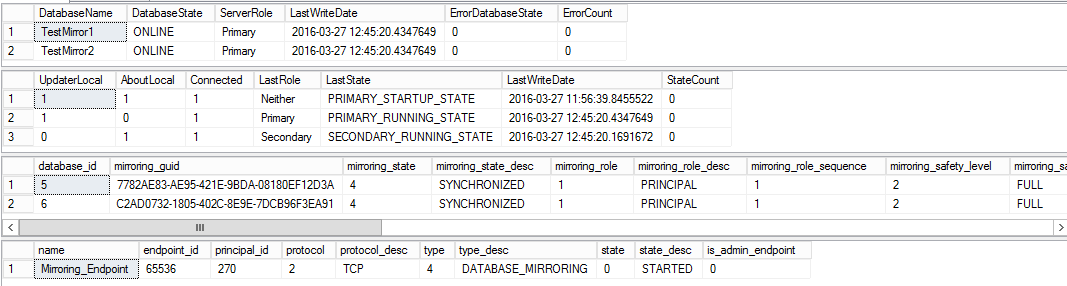


Figure 7: Example result from Primary Role Server displaying some of the inner states

#### Secondary Role Server

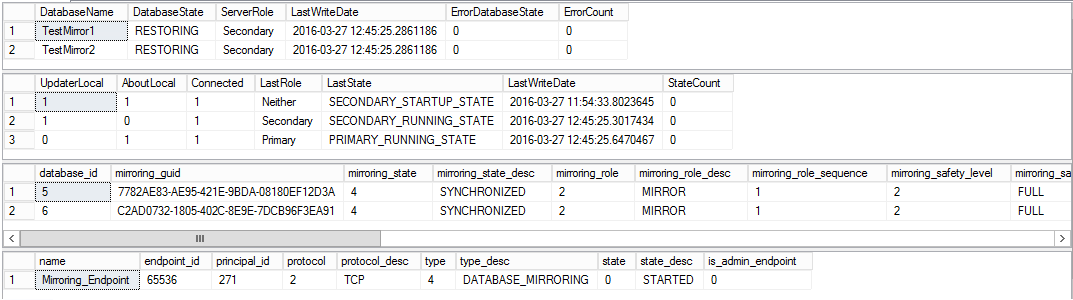


Figure 8: Example result from Secondary Role Server displaying some of the inner states

### Database Mirroring Monitoring

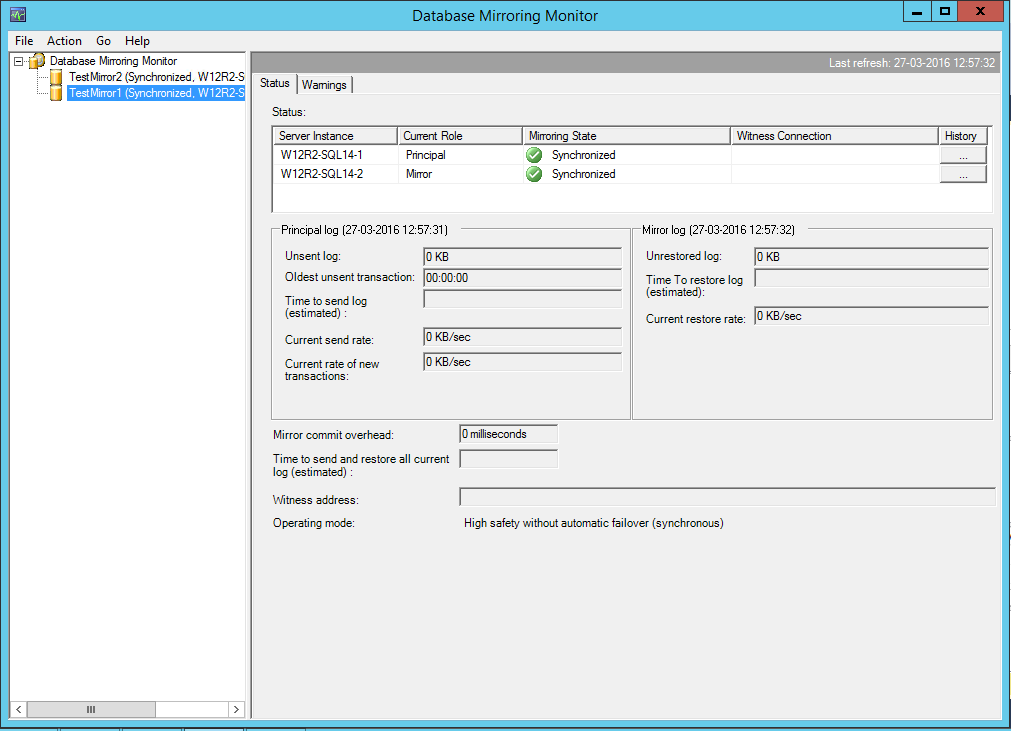


Figure 9: Database Mirroring Monitor (SSMS buildin monitor)

## Startup Problems

Check the log to see where the MirrorLib stops.

### Stops at directory creation

There might not be high enough security level for the install account to set the security for the LocalRestore, LocalBackup and LocalShare.

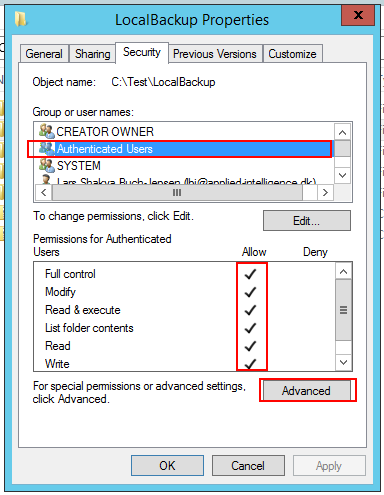


Figure 10: Startup Problems: Stops at directory creation

It can be set up manually by creating directory, right-click on directory and select **Properties**. On **Security** tab set the security access to Authenticated Users with full access which is done from the **Advanced** button.

### Stops at share setup

There might not be high enough security level for the install account to create share so making it manually and setting security to Authenticated Users with full access.

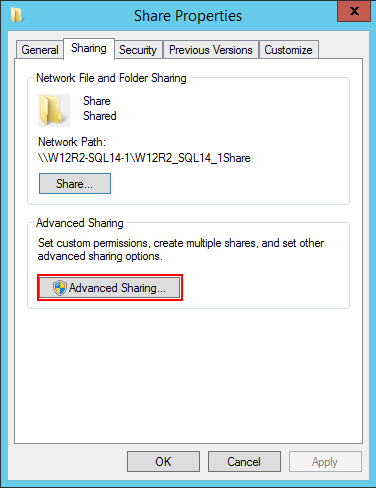


Figure 11: Startup Problems: Stops at share creation - Share Tab

It can be set up manually by creating directory, right-click on directory and select **Properties**. On **Share** tab press the **Advanced Sharing** button.

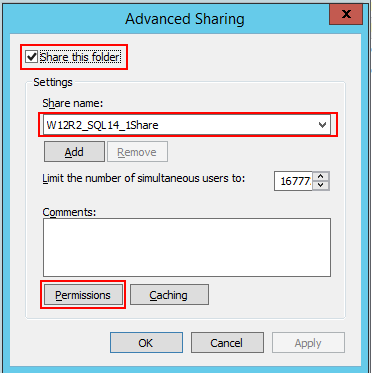


Figure 12: Startup Problems: Stops at share creation - Advanced Sharing

Check the **Share this folder** and write the share name. Click on the **Permissions** button to set up security.

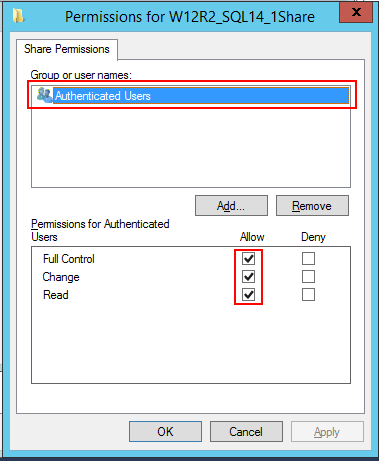


Figure 13: Startup Problems: Stops at share creation – Permissions

On the **Permissions** screen add Authenticated Users and give full access. On AD systems it might be a good thing to delete guest access as that is access for no AD users.

### Stops under creation of endpoint

This will happen as creation of endpoint throws an exception but creates and starts the endpoint which is a bug in SMO. This has been tried to avoid by checking for existence of the endpoint before allowing exception to be rethrown. It might fail some times and the workaround is to restart the MirrorLib hosted application.

### Stops when trying to access remote server

This might fail because there is no connection to remote server. It might be a missing inbound rule for the firewall to allow connection on the listener port or the normal port access between servers. Check firewall settings.

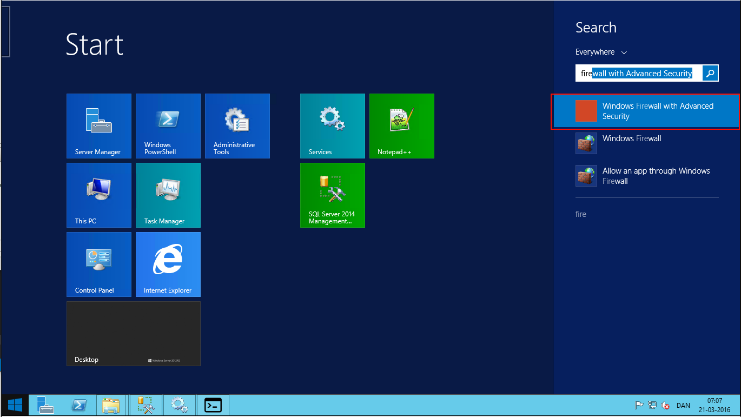


Figure 14: Startup Problems: Stops at failover - Select firewall

Select **Windows Firewall with Advanced Security** by clicking on windows button and searching for “fire” and start **As administrator** by right-clicking.

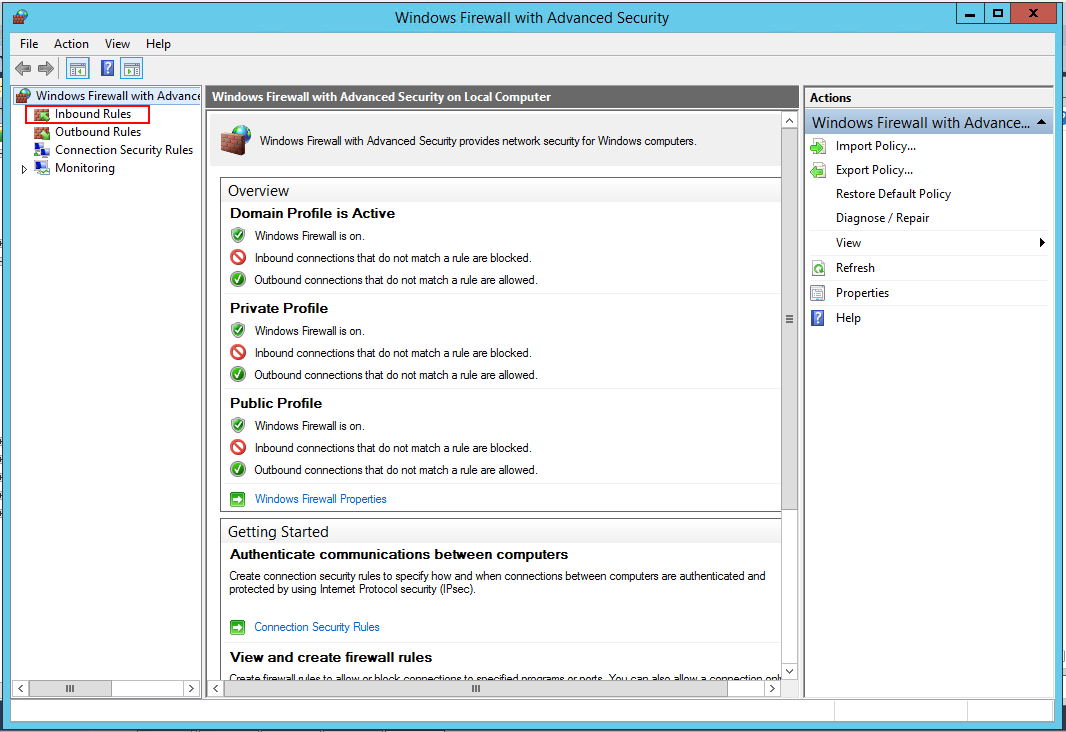


Figure 15: Startup Problems: Stops at failover - Firewall

Select **Inbound Rules** in upper left corner.



Figure 16: Startup Problems: Stops at failover – Inbound Rules

In upper right corner select **New Rule …**

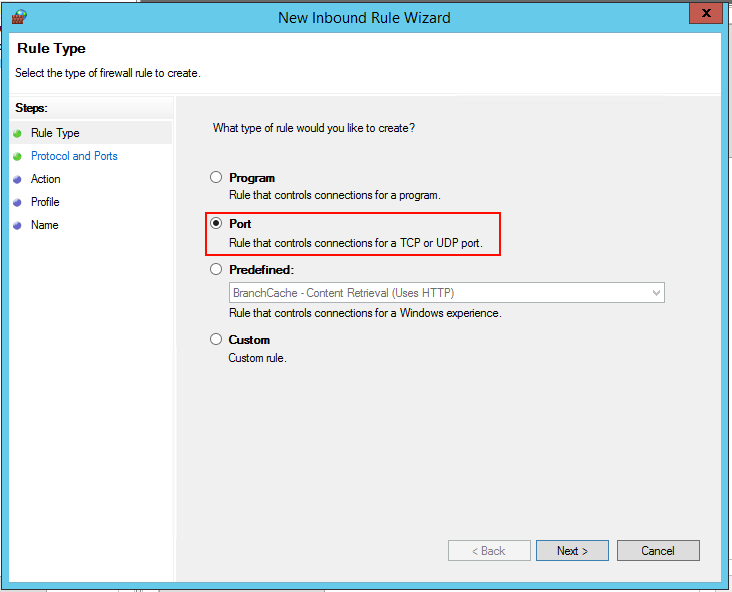


Figure 17: Startup Problems: Stops at failover – Port

Select the **Port** rule as it is the listener port we need to open for and click next.

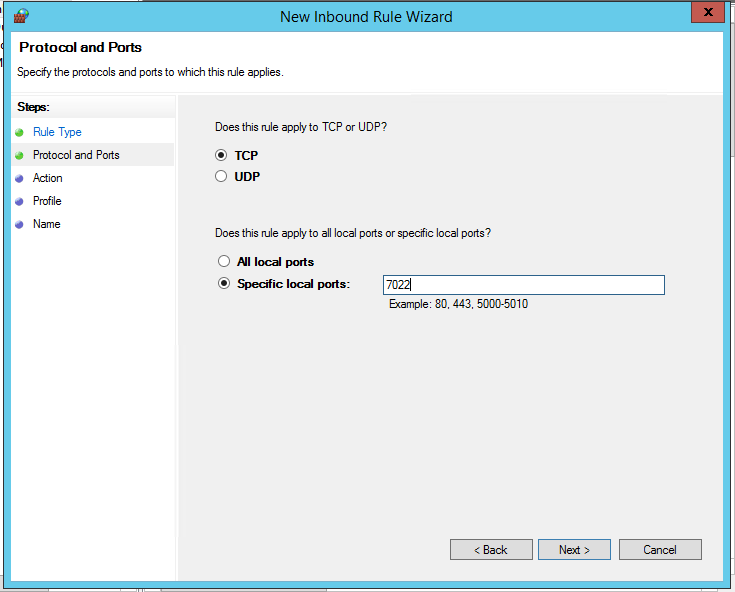


Figure 18: Startup Problems: Stops at failover – Port Number

Enter the port number and click **Next**.

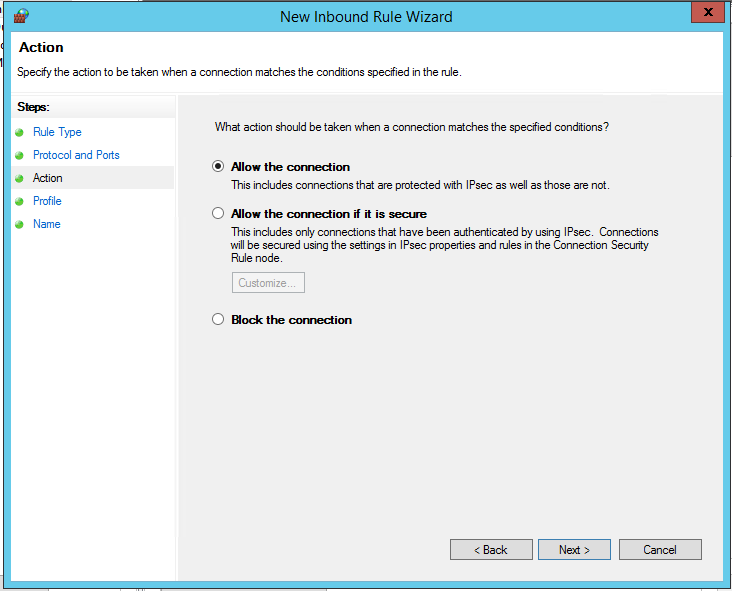


Figure 19: Startup Problems: Stops at failover – Action Allow

Take a decision or just leave it at default. Click **Next**.

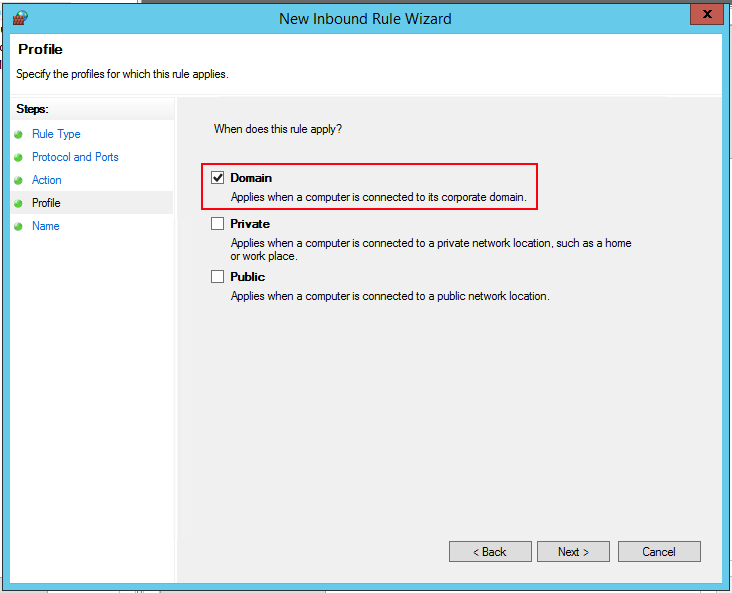


Figure 20: Startup Problems: Stops at failover – Active Profile

Check the **Domain** if in AD and click **Next**.

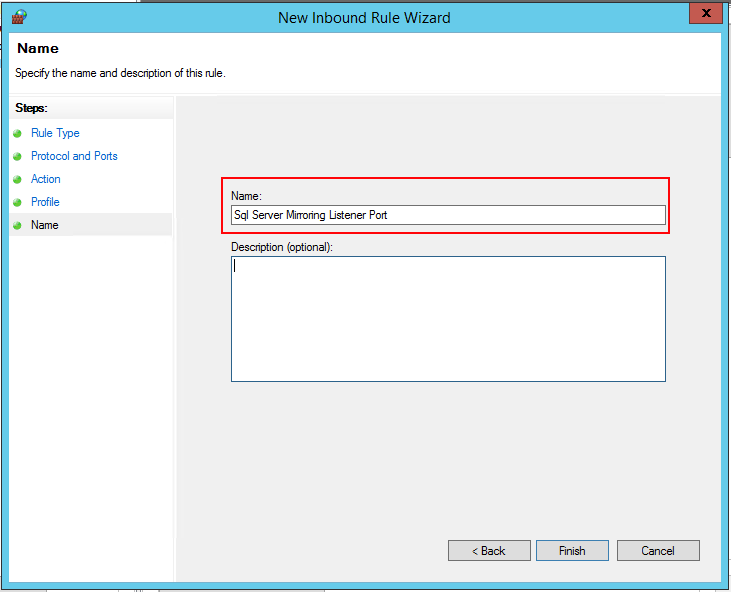


Figure 21: Startup Problems: Stops at failover – Name firewall rule

Name the rule something that makes it easy to understand and click **Finish**.

Repeat for port Sql server ports 1433 (normal Sql Server connection) and 1434 (administrative priority access).

### Secondary Role server does not exit SECONDARY\_STARTUP\_STATE

This might be due to mirroring setup error.

### Reports unknown server name in Log

The SQL Server has been renamed after the Sql Server was installed and not renamed correctly with makes the @@SERVERNAME report the wrong name as it is cached internally in Sql Server. The @@SERVERNAME is used internally in scripts fx for monitoring with Database Mirroring Monitor and for setting up mirroring.

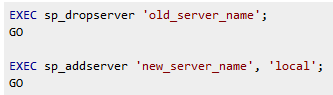


Figure 22: Script to correct reporting from @@SERVERNAME

Run the script for renaming and restart the Sql Server service. If this is the case and mirroring have been partially started the steps in 8.4 needs to be done.

## Disable Mirroring from database and redo

In order to restart mirroring if something really has gone wrong. On the Primary Role server right-click the database in SSMS and select Properties. In the Database Properties select the page Mirroring and click on the Remove Mirroring. On the Secondary Role server the databases are probably not worth having there so just delete them. When the MirrorLib start on the Primary Role server a backup is taken because of the missing mirroring and the Secondary role server will restore the new backup in the restoring state.

## Need to manually failover databases in SSMS

It is possible to make a switch between principal and mirror. On the Primary Role server right-click the database in SSMS and select Properties. In the Database Properties select the page Mirroring and click on the Failover. MirrorLib also have support for this by code.

## Delete mirrored database

It is not possible to delete the database directly if it is mirrored. Mirroring needs to be disabled before. It might be needed to do so in TSQL if 8.4 is not possible from SSMS.

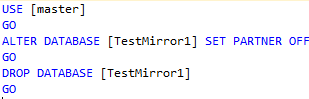


Figure 23: Remove mirroring and delete database in TSQL

## Adding server that has been down for longer periods of time

When the server is online make sure it does not continue on MIrrorLib but is restarted as the proper server role.

## Adding a new server

When adding a new server set up the server with all and let MirrorLib restore database and resume mirroring by starting server in Secondary Role.

## Misc Mirroring problems

As the server is set up as the normal Sql Server Mirroring normal diagnostic methods can be used for the normal errors.

Source: [SSMS configuration and troubleshooting](https://www.mssqltips.com/sqlservertip/2464/configure-sql-server-database-mirroring-using-ssms/)

## Switch-over for multiple databases

SMO WMI is used to make switch-over for multiple databases which has a simple debugging. This is only used if a Sql Server Witness server is set up in the system.

Source: [Switch-over for multiple databases](https://www.mssqltips.com/sqlservertip/1564/database-mirroring-automating-failover-for-multiple-sql-server-databases/)