|  |
| --- |
| **Database Mirroring Cases and Responses** |

By

Eitan Blumin  
SQL Server Consultant  
Madeira Data Solutions  
[eitan@madeiradata.com](mailto:eitan@madeiradata.com)

Copyright © 2015 Madeira Data Solutions

Version 3 (2015–03–11)

# Table of Contents

[Table of Contents 2](#_Toc413843260)

[Introduction 4](#_Toc413843261)

[Important Resources 4](#_Toc413843262)

[Standard (Normal) State 5](#_Toc413843263)

[Monitoring Tools 6](#_Toc413843264)

[MSDB Database Mirroring Monitoring Stored Procedure sp\_dbmmonitorresults 6](#_Toc413843265)

[Database Mirroring Monitor (GUI) 7](#_Toc413843266)

[SQL Server Profiler Events 8](#_Toc413843267)

[Database Mirroring Continuous Monitoring 8](#_Toc413843268)

[Using Warning Thresholds and Alerts 9](#_Toc413843269)

[Setting Up and Managing Warning Thresholds 10](#_Toc413843270)

[Warning Thresholds and Alerts Recommended for WhiteBay 11](#_Toc413843271)

[Response Tools 11](#_Toc413843272)

[Response Tool 1: Manual Failover 11](#_Toc413843273)

[Response Tool 2: Disable Witness 12](#_Toc413843274)

[Response Tool 3: Break Mirroring Session 12](#_Toc413843275)

[Response Tool 4: Re-Establish Mirroring Session 12](#_Toc413843276)

[Response Tool 5: Restarting Endpoints 13](#_Toc413843277)

[Emergency Cases 14](#_Toc413843278)

[Case 1: Principle Server is Unavailable 14](#_Toc413843279)

[Response to Case 1: Automatic Failover 14](#_Toc413843280)

[Case 2: Mirror Server is Unavailable 14](#_Toc413843281)

[Response A to Case 2: None 15](#_Toc413843282)

[Response B to Case 2: Disable Witness 15](#_Toc413843283)

[Response C to Case 2: Break Mirroring 15](#_Toc413843284)

[Case 3: Witness Server is Unavailable 15](#_Toc413843285)

[Response A to Case 3: Disable Witness 16](#_Toc413843286)

[Response B to Case 3: Manual Failover 16](#_Toc413843287)

[Case 4: Principle Isolated (Partner and Witness Servers Unavailable) 16](#_Toc413843288)

[Response A to Case 4: Restart Mirroring Endpoints 16](#_Toc413843289)

[Response B to Case 4: Break Mirroring Session 16](#_Toc413843290)

[Case 5: Threshold Alerts Fired 17](#_Toc413843291)

[Response A to Case 5: System Performance Checks 17](#_Toc413843292)

[Response B to Case 5: Change Mode to High Performance 17](#_Toc413843293)

[Response C to Case 5: Other Extreme Measures 18](#_Toc413843294)

# Introduction

This document includes basic explanation about Database Mirroring, specifically between **PRDSRV1** and **PRDSRV2** as an example. This document also contains common "emergency" scenarios, and ways to troubleshoot them.

This document assumes the reader has a basic understanding of Database Mirroring in SQL Server, basic understanding of the SQL Server environment, and basic understanding of TSQL.

For basic introductory information regarding Database Mirroring in SQL Server, please refer to: <https://msdn.microsoft.com/en-us/library/ms189852%28v=sql.100%29.aspx>

## Important Resources

This document is based on the following official Microsoft knowledge base sources:

* + - * Quorum: How a Witness Affects Database Availability:  
        <https://msdn.microsoft.com/en-us/library/ms189902%28v=sql.100%29.aspx>
      * Possible Failures During Database Mirroring:  
        <https://msdn.microsoft.com/en-us/library/ms190913%28v=sql.100%29.aspx>
      * sp\_dbmmonitorresults (Transact-SQL):  
        <https://msdn.microsoft.com/en-us/library/ms366320%28v=sql.100%29.aspx>
      * Using Warning Thresholds and Alerts on Mirroring Performance Metrics:  
        <http://msdn.microsoft.com/en-us/library/ms408393%28v=sql.100%29.aspx>

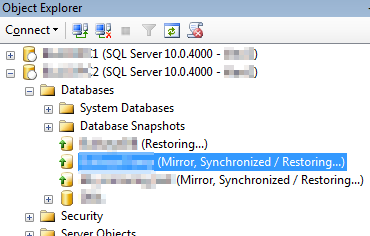
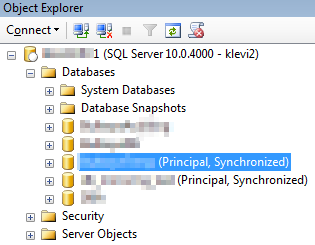
# Standard (Normal) State

The "Normal" state that we'll be assuming for our mirroring session is a configuration of the type "synchronous with witness (High Safety mode with Automatic Failover)". Which is:

* + - 1. PRDSRV1 as the Principle.
      2. PRDSRV2 as the Mirror (Partner).
      3. PRDWITNESS as the Witness.

**With all 3 servers maintaining connectivity with each other.**We will consider a database mirroring session as "Normal" when we can see in the SQL Server Management Studio, under the "Databases" tree, that next to the mirrored database we can see the words "(Principle, Synchronized)" in PRDSRV1, and "(Mirror, Synchronized / Restoring…)" in PRDSRV2.

Illustrated here:



We can also monitor the mirroring status by connecting to PRDSRV1 or PRDSRV2 and running the command: exec msdb..sp\_dbmmonitorresults 'MyProductionDB'  
We can then look at the columns **role**, **mirroring\_state**, and **witness\_status** to see the current database mirroring status:

|  |  |
| --- | --- |
| Column | Description |
| role | Current mirroring role of the server instance: 1 = Principal 2 = Mirror |
| mirroring\_state | State of the database: 0 = Suspended 1 = Disconnected 2 = Synchronizing  3 = Pending Failover 4 = Synchronized |
| witness\_status | Connection status of the witness in the database mirroring session of the database, can be: 0 = Unknown 1 = Connected 2 = Disconnected |

# Monitoring Tools

The following monitoring and alert tools are available to us.

## MSDB Database Mirroring Monitoring Stored Procedure sp\_dbmmonitorresults

This procedure allows us to monitor and track the changes in the DB Mirroring states, as well as the latency and performance of the synchronization.

The data presented by this procedure is collected by a scheduled job called "**Database Mirroring Monitor Job**". You can change the schedule of this job in order to modify the sampling rate, and therefore granularity, of the data.

**Example for how to use the monitoring results procedure:**

exec msdb..sp\_dbmmonitorresults 'MyProductionDB', 0

**Syntax:**

sp\_dbmmonitorresults *database\_name, rows\_to\_return, update\_status*

**Arguments:**

*database\_name*

Specifies the database for which to return mirroring status.

*rows\_to\_return*

Specifies the quantity of rows returned. Possible values:

* 0 = Last row
* 1 = Rows last two hours
* 2 = Rows last four hours
* 3 = Rows last eight hours
* 4 = Rows last day
* 5 = Rows last two days
* 6 = Last 100 rows
* 7 = Last 500 rows
* 8 = Last 1,000 rows
* 9 = Last 1,000,000 rows

*update\_status*

Specifies that before returning results the procedure:

* 0 = Does not update the status for the database. The results are computed using just the last two rows, the age of which depends on when the status table was refreshed.
* 1 = Updates the status for the database by calling **sp\_dbmmonitorupdate** before computing the results. However, if the status table has been updated within the previous 15 seconds, or the user is not a member of the **sysadmin** fixed server role, **sp\_dbmmonitorresults** runs without updating the status.

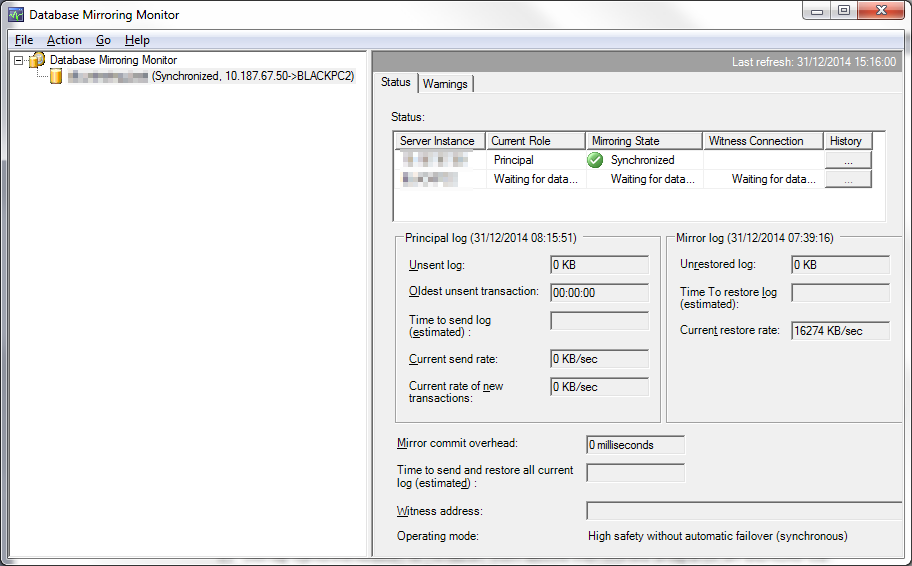
**Results Set:**

The procedure returns the requested number of rows of history status for the specified database. Each row contains the following information (interesting columns marked in yellow):

|  |  |  |
| --- | --- | --- |
| **Column name** | **Data type** | **Description** |
| **database\_name** | sysname | Name of a mirrored database. |
| **role** | int | Current mirroring role of the server instance: 1 = Principal 2 = Mirror |
| **mirroring\_state** | int | State of the database: 0 = Suspended 1 = Disconnected 2 = Synchronizing  3 = Pending Failover 4 = Synchronized |
| **witness\_status** | int | Connection status of the witness in the database mirroring session of the database, can be: 0 = Unknown 1 = Connected 2 = Disconnected |
| **log\_generation\_rate** | int | Amount of log generated since preceding update of the mirroring status of this database in kilobytes/sec. |
| **unsent\_log** | int | Size of the unsent log in the send queue on the principal in kilobytes. |
| **send\_rate** | int | Send rate of log from the principal to the mirror in kilobytes/sec. |
| **unrestored\_log** | int | Size of the redo queue on the mirror in kilobytes. |
| **recovery\_rate** | int | Redo rate on the mirror in kilobytes/sec. |
| **transaction\_delay** | int | Total delay for all transactions in milliseconds. |
| **transactions\_per\_sec** | int | Number of transactions that are occurring per second on the principal server instance. |
| **average\_delay** | int | Average delay on the principal server instance for each transaction because of database mirroring. In high-performance mode (that is, when the SAFETY property is set to OFF), this value is generally 0. |
| **time\_recorded** | datetime | Time at which the row was recorded by the database mirroring monitor. This is the system clock time of the principal. |
| **time\_behind** | datetime | Approximate system-clock time of the principal to which the mirror database is currently caught up. This value is meaningful only on the principal server instance. |
| **local\_time** | datetime | System clock time on the local server instance when this row was updated. |

## Database Mirroring Monitor (GUI)

This is the graphical interface version of the above stored procedure.  
You can open it by **Right-clicking** on the mirrored database and selecting **"Tasks" > "Launch Database Mirroring Monitor…".** This graphical interface also allows easy configuration of the Warning Thresholds and Alerts.



## SQL Server Profiler Events

We can use these events to track *changes* in the database mirroring state (automatic/manual failover, losing/regaining connection and synchronization, etc.).

* Database Mirroring Connection
* Database Mirroring State Change

Unlike the monitoring procedure and the Database Mirroring Monitor GUI described in previous sections, there is no scheduled sampling by some job that needs to run. Profiler Events are reported **immediately** when the event happens (Real-time).

So by recording these SQL Server Profiler Events (to a file or a table), we can know *exactly when* mirroring state changes happened, down to the millisecond.

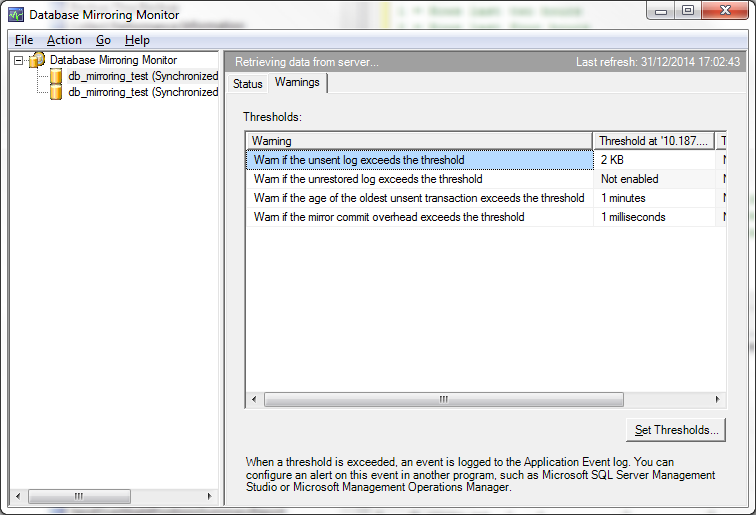
We probably won't need it in our case, but it's good to know that this option is also available.

## Database Mirroring Continuous Monitoring

The "**Valinor - DB Mirroring State Change**" job sends us an **e-mail** whenever a database changes its mirroring state. This will let us know of failures, disconnections, failovers etc.

## Using Warning Thresholds and Alerts

We can set up "Warning Thresholds and Alerts" using either the "Database Mirroring Monitor" GUI or the stored procedures [as explained here](http://msdn.microsoft.com/en-us/library/ms408393%28v=sql.100%29.aspx). Here we can notify ourselves, through the **Windows Event Log**, if and when there's a synchronization delay, lots of unsent transactions, and so on:  
  
Note that if we want to be notified about these alerts, IT will need to set up alerts on the Windows Event Log (based on the Error IDs [as detailed here](http://msdn.microsoft.com/en-us/library/ms408393%28v=sql.100%29.aspx)):



|  |  |
| --- | --- |
| **Performance metric** | **Event ID** |
| Unsent log | 32042 |
| Unrestored log | 32043 |
| Oldest unsent transaction | 32040 |
| Mirror commit overhead | 32044 |

The following table lists the performance metrics for which warnings can be configured, describes the corresponding warning threshold, and lists the corresponding Database Mirroring Monitor label.

|  |  |  |
| --- | --- | --- |
| **Performance metric** | **Warning threshold** | **Database Mirroring Monitor label** |
| Unsent log | Specifies how many kilobytes (KB) of unsent log generate a warning on the principal server instance. This warning helps measure the potential for data loss in terms of KB and is especially relevant for high-performance mode. However, the warning is also **relevant for high-safety mode when mirroring is paused or suspended because the partners become disconnected**. | Warn if the unsent log exceeds the threshold |
| Unrestored log | Specifies how many KB of unrestored log generate a warning on the mirror server instance. This warning helps measure failover time. Failover time consists mainly of the time that the former mirror server requires to roll forward any log remaining in its redo queue, plus a short additional time.  Note For an automatic failover, the time for the system to notice the error is independent of the failover time. For more information, see [Estimating the Interruption of Service During Role Switching](https://msdn.microsoft.com/en-us/library/ms187465%28v=sql.100%29.aspx). | Warn if the unrestored log exceeds the threshold |
| Oldest unsent transaction | Specifies the number of minutes worth of transactions that can accumulate in the send queue before a warning is generated on the principal server instance. This warning helps measure the potential for data loss in terms of time and is especially relevant for high-performance mode. However, the warning is also **relevant for high-safety mode when mirroring is paused or suspended because the partners become disconnected**. | Warn if the age of the oldest unsent transaction exceeds the threshold |
| Mirror commit overhead | Specifies the number of milliseconds of average delay per transaction that are tolerated before a warning is generated on the principal server. This delay is the amount of overhead incurred while the principal server instance waits for the mirror server instance to write the transaction's log record into the redo queue. **This value is relevant only in high-safety mode**. | Warn if the mirror commit overhead exceeds the threshold |

### Setting Up and Managing Warning Thresholds

Example:

EXEC sp\_dbmmonitorchangealert 'MyProductionDB', 1, 5, 1

Syntax:

sp\_dbmmonitorchangealert *database\_name, alert\_id, alert\_threshold, enabled*

Arguments:

*database\_name*

Specifies the database for which to add or change the specified warning threshold.

*alert\_id*

An integer value that identifies the warning to be added or changed. Specify one of the following values:

|  |  |  |
| --- | --- | --- |
| **Value** | **Performance metric** | **Warning threshold** |
| 1 | Oldest unsent transaction | Specifies the number of minutes worth of transactions that can accumulate in the send queue before a warning is generated on the principal server instance. This warning helps measure the potential for data loss in terms of time, and it is particularly relevant for high-performance mode. However, the warning is also **relevant for high-safety mode when mirroring is paused or suspended because the partners become disconnected.** |
| 2 | Unsent log | Specifies how many kilobytes (KB) of unsent log generate a warning on the principal server instance. This warning helps measure the potential for data loss in terms of KB, and it is particularly relevant for high-performance mode. However, the warning is also **relevant for high-safety mode when mirroring is paused or suspended because the partners become disconnected.** |
| 3 | Unrestored log | Specifies how many KB of unrestored log generate a warning on the mirror server instance. This warning helps measure failover time. Failover time consists mainly of the time that the former mirror server requires to roll forward any log remaining in its redo queue, plus a short additional time. |
| 4 | Mirror commit overhead | Specifies the number of milliseconds of average delay per transaction that are tolerated before a warning is generated on the principal server. This delay is the amount of overhead incurred while the principal server instance waits for the mirror server instance to write the transaction's log record into the redo queue. **This value is relevant only in high-safety mode.** |
| 5 | Retention period | Metadata that controls how long rows in the database mirroring status table are preserved. |

*alert\_threshold*

The threshold value for the warning. If a value above this threshold is returned when the mirroring status is updated, an entry is entered into the Windows event log. This value represents KB, minutes, or milliseconds, depending on the performance metric.

*enabled*

Is the warning enabled?

0 = Warning is disabled.

1 = Warning is enabled.

### Warning Thresholds and Alerts Recommended for High Safety

Since the chosen Mirroring mode in this use-case is "high-safety" (synchronous), the 3 metrics "Oldest unsent transaction", "Unsent log" and "Unrestored log" are irrelevant in day-to-day activities. These metrics would be relevant only if the mirroring session is **paused or suspended** (and we already have alerts set up for that).  
This leaves only "**Mirror commit overhead**" which is relevant *only* to "high-safety" mode.

Therefore, based on monitoring data of Database Mirroring performance between PRDSRV1 and PRDSRV2, I recommend the following threshold configuration:

|  |  |
| --- | --- |
| **Performance metric** | **Warning threshold** |
| Mirror commit overhead | **100** milliseconds |

This is based on the performance data where the maximum transaction delay was 69 ms, and the maximum average was 17 milliseconds. As time goes by, we can decide to adjust these thresholds as needed.

To see what sort of responses you should follow in the event of these alerts firing, please refer to section [Case 5: Threshold Alerts Fired](#_Case_5:_Threshold_Alerts Fired).

# Response Tools

As part of maintaining the database mirroring session, we will need to be familiar with the specific set of "tools" at our disposal. We will need to know how to use these "tools" when required doing so as part of our response to various possible cases and emergencies.

## Response Tool 1: Manual Failover

In the case of when a manual failover is required, you can run the following command **at the Principle** server:

ALTER DATABASE MyProductionDB SET PARTNER FAILOVER

After you do this, the mirroring partners will switch roles: The Mirror server will now become the Principle, and the Principle server will become the Mirror.

## Response Tool 2: Disable Witness

In the case of when you wish to disconnect the **witness** server from the mirroring session, you can run the following command **at the Principle** server:

ALTER DATABASE MyProductionDB SET WITNESS OFF

**Note: This will turn off automatic failover.**

## Response Tool 3: Break Mirroring Session

In the case of when you need to turn off the mirroring session completely, you can run the following command on **either the Principle, or the Mirror** server:

ALTER DATABASE MyProductionDB SET PARTNER OFF

And then, in order to bring the database on the **mirror** server online, you need to run the command:

RESTORE LOG MyProductionDB WITH RECOVERY

## Response Tool 4: Re-Establish Mirroring Session

If, after breaking the mirroring session, you need to re-establish it, you'll need to perform the following steps (assuming PRDSRV1 is the Principle, and PRDSRV2 is the Mirror):

* If between the time of turning off the mirroring session and until now, there was no transaction log backup performed on the principle database, and if the mirror database is still in "Restoring" mode, then you can **skip to step 5**.
* If there were transaction log backups performed on the principle database since the mirroring session was turned off, and if the mirror database is still in "Restoring" mode, then you'll need to first restore these backups (**skip to step 4**).
* If the principle and mirror databases were de-synchronized somehow (for example, the mirror database was/is online after the mirroring session was turned off), then you'll need to do the whole thing **starting with step 1**.

1. Backup the *MyProductionDB* database from the *Principle* server (**full backup**).
2. Backup the *MyProductionDB* database from the *Principle* server (**transaction log** backup).
3. Restore the **full backup** on the *Mirror* server with the NORECOVERY option. For example:  
   RESTORE DATABASE [MyProductionDB] FROM DISK = N'D:\DB\_Mirror\MyProductionDB\MyProductionDB.bak' WITH NORECOVERY
   1. If the database already exists on the *Mirror* server, you can restore the backup with the REPLACE option in order to overwrite it. For example:  
      RESTORE DATABASE [MyProductionDB] FROM DISK = N'D:\DB\_Mirror\MyProductionDB\MyProductionDB.bak' WITH NORECOVERY, REPLACE
4. Restore **the transaction log** backup on the *Mirror* server with the NORECOVERY option. For example:  
   RESTORE LOG [MyProductionDB] FROM DISK = N'D:\DB\_Mirror\MyProductionDB\MyProductionDB.trn' WITH NORECOVERY
   1. If there's more than one transaction log backup performed after the latest full backup, you'll need to restore each and every one of them in the correct order. All of them with the NORECOVERY option. The following script can help:
5. Run the following command on the **Mirror** server (PRDSRV2):  
   ALTER DATABASE [MyProductionDB] SET PARTNER = 'TCP://[PRDSRV2\_FQDN\_or\_IP]:7024'
6. Run the following command on the **Principle** server (PRDSRV1):  
   ALTER DATABASE [MyProductionDB] SET PARTNER = 'TCP://[PRDSRV1\_FQDN\_or\_IP]:7024'
7. In order to connect the witness (PRDWITNESS) to the mirroring session, run the following command on the **Principle** server (PRDSRV1):  
   ALTER DATABASE [MyProductionDB] SET WITNESS = 'TCP://[PRDWITNESS\_FQDN\_or\_IP]:7024'

## Response Tool 5: Restarting Endpoints

In some cases, it's possible that the mirroring session doesn't "fix" itself automatically. For example, if one of the databases was in an "isolated" mode (outside the quorum), and/or if there were serious network issues. In these scenarios, even though all the servers are up and running (principle, partner and witness), it's possible that the database mirroring still doesn't resume and no one takes the role of principle automatically.

Sometimes, executing a **manual failover** (on the principle server) can solve the problem (refer to "[Response Tool 1: Manual Failover"](#_Response_Tool_1:_Manual Failover) for instructions). But sometimes it's not enough.  
In this case, it's usually possible to fix the problem by "restarting" the database mirroring endpoints on one or more of the servers.

**Important Note:** In order for this to take effect, the mirroring session must already be established. If the mirroring session is broken / disabled, then you'll need to re-establish it (refer to "[Response Tool 4: Re-Establish Mirroring Session](#_Response_Tool_4:_Re-Establish Mirro)").

To restart the database mirroring endpoints on a server, run the following commands:

USE master  
GO  
ALTER ENDPOINT [Endpoint\_Mirroring] STATE=STOPPED;  
ALTER ENDPOINT [Endpoint\_Mirroring] STATE=STARTED;

Try to run these commands on the witness first, then if after a moment still nothing changes, run them on the partner server, then if after a moment still nothing changes, run them on the principle as well.

If after restarting the endpoints on all three servers, still nothing changes, then try restarting the SQL Server services (in the same order as above: witness, partner, principle). If that still doesn't help (or isn't possible for whatever reason), then you may need to re-establish the mirroring session (refer to "[Response Tool 4: Re-Establish Mirroring Session](#_Response_Tool_4:_Re-Establish Mirro)").

# Emergency Cases

These are the various possible emergency scenarios that could happen during a SQL Server Database Mirroring session, and what should be done in each case.

## Case 1: Principle Server is Unavailable

This is when the Principle server (PRDSRV1) is unavailable for whatever reason, and specifically, loses connectivity with PRDSRV2 and **PRDWITNESS**, or the database **MyProductionDB** itself becomes unresponsive.

### Response to Case 1: Automatic Failover

Assuming, of course, that both the partner (PRDSRV2) and the witness (PRDWITNESS) are still available and maintain connectivity with each other, the database mirroring session will perform an automatic failover, thus turning the partner (PRDSRV2) into the Principle server. If the fault was caused by network problems, the failover will start immediately. But if there's a "hang" due to some other problem, then it'll take SQL Server **10 seconds** (by default) before it decides to initiate the failover.

There is nothing special to be done in the database manually, other than making sure the database indeed performed an automatic failover, and applications are connecting to PRDSRV2.

## Case 2: Mirror Server is Unavailable

If the mirror server (PRDSRV2) becomes unavailable, that means the Principle database is in an "exposed" mode: The principle server is now in danger because its failover partner is unavailable.

Until the mirroring server becomes available again, the log "send queue" will fill up on the principle server. When the mirror server becomes available, the unsent log is automatically delivered from the principle, and is rolled forward on the mirroring partner.

**Please note** that until the partner becomes available again, the log on the principle will keep growing, and may pose a **serious threat to disk space** if left for too long in this state.

### Response A to Case 2: None

If you expect the mirroring partner to become available again soon, then there's no need for you to do anything. The database mirroring session will send all the unsent log data to the partner once it becomes available, and bring it up to speed on any changes.

### Response B to Case 2: Disable Witness

If you suspect, for whatever reason, that **the witness server is about to become unavailable as well**, then it's best to disconnect the witness from the mirroring session (refer to the "[Response Tool 2: Disable Witness](#_Response_Tool_2:_Disable Witness)" section for instructions).

If you don't, and the witness becomes unavailable, then the principle database will have "lost quorum" (it won't be able to see neither the partner, nor the witness). In such a case, the **principle database** will **automatically become unavailable too**. This is an automatic safety mechanism that's meant to protect the database servers from becoming de-synchronized, and/or avoid a situation where both databases become active principles. If this happens, and you can't make the witness/partner servers available again, then you will need to *break the mirroring session* in order to bring the principle database back online. For instructions on how to break mirroring, refer to section "[Response Tool 3: Break Mirroring Session](#_Response_Tool_3:_Break Mirroring Se)".

### Response C to Case 2: Break Mirroring

If you expect the mirroring server to become permanently unavailable, and/or you need to reinstall the mirroring server and/or revert it to an earlier snapshot/image, then it's best to disable the mirroring session altogether. After the mirroring partner becomes available again, then you can take the steps necessary to re-establish the mirroring session. But until then, there's no use in having the principle database have its "send queue" fill up, and its transaction log bloated.

For instructions on how to break mirroring, refer to section "[Response Tool 3: Break Mirroring Session](#_Response_Tool_3:_Break Mirroring Se)".

## Case 3: Witness Server is Unavailable

This case talks about a scenario where the witness server (PRDWITNESS) becomes unavailable, and specifically loses connectivity with both the principle (PRDSRV1) as well as the mirror (PRDSRV2).

In this scenario, the principle and mirror servers are reduced to a quorum of 2 out of 3.  
This is a dangerous situation where automatic failover not only becomes unavailable, but in this scenario, if one of the partners becomes unavailable too (either one), then the other partner becomes automatically unavailable as well. This is an automatic safety mechanism that's meant to protect SQL Server from a situation of two active principles per one mirroring session.

### Response A to Case 3: Disable Witness

If you expect one of the partner servers to become unavailable as well, then it's best to disconnect the witness from the mirroring session (refer to section "[Response Tool 2: Disable Witness](#_Response_Tool_2:_Disable Witness)" for instructions). This will turn off the quorum safety mechanism, and thus prevent from both partners to become unavailable when only one of them is truly unavailable. Please note, though, that **this will turn off automatic failover**.

### Response B to Case 3: Manual Failover

In addition, **if you expect the Principle database to become unavailable as well**, then it's probably best to initiate a **manual failover** in order to switch the roles in advance. If you manage to do this in time, then you can save yourself the trouble of having to break the mirroring session and re-establish it later (refer to section "[Response Tool 1: Manual Failover](#_Response_Tool_1:_Manual Failover)" for instructions).

Alternatively, if you expect the **Mirror** database to become unavailable in addition to the witness, then there's no real need for you to do anything (other than disabling the witness). Once the mirror server becomes unavailable, the principle server will start collecting its unsent data in its "send queue" until the mirror becomes available again.  
If the Mirror database becomes unavailable in addition to the witness, then it's the same as "[Case 2: Mirroring Partner is Unavailable](#_Case_2:_Mirror_Server is Unavailabl)" (only without the witness, obviously).

## Case 4: Principle Isolated (Partner and Witness Servers Unavailable)

In this scenario, the principle database has "lost quorum" because it cannot see either of the other two servers.

If you didn't disconnect the witness from the mirroring session in time, this would mean that SQL Server will have automatically turned off the principle database as well.

### Response A to Case 4: Restart Mirroring Endpoints

If, after the principle server has been "isolated", the witness and/or partner servers become available again, but the principle database is still in the "In Recovery" mode (and therefore unavailable), then you'll need to **restart the mirroring endpoints**.

For instructions on how to restart the endpoints, refer to section "[Response Tool 5: Restart Endpoints](#_Response_Tool_5:_Restarting Endpoin)".

### Response B to Case 4: Break Mirroring Session

If for whatever reason, you don't expect the partner or witness servers to become available again soon, then all you have left to do is to **break the mirroring session**.

For instructions on how to break mirroring, refer to section "[Response Tool 3: Break Mirroring Session](#_Response_Tool_3:_Break Mirroring Se)".  
If after turning off the mirroring, the principle database is still in the "In Recovery" or "Restoring" mode, then you need to run the following command to bring it online:  
RESTORE LOG MyProductionDB WITH RECOVERY

Then, similarly to Case 2, after the other servers become available again, you can re-establish the database mirroring session.

For instructions on how re-establish the mirroring session, refer to section "[Response Tool 4: Re-Establish Mirroring Session](#_Response_Tool_4:_Re-Establish Mirro)".

## Case 5: Threshold Alerts Fired

For basic important information about Warning Thresholds and Alerts, please [refer to the relevant section](#_Using_Warning_Thresholds_and Alerts).

Remember that since the chosen mirroring mode for this use-case is "High Safety" (synchronous), there's only one performance metric relevant for us, which is "Mirror commit overhead" (a.k.a. Transaction Delay).

In the event of such an alert being fired, it would mean the Mirroring partner is having trouble keeping up with the transactions on the Principle server. This could be caused by an exceedingly high database activity, or by network problems, or by performance issues on the partner server.

### Response A to Case 5: System Performance Checks

* Look for, and accordingly take care of, possible performance problems on the **Principle** server (long-running DML commands, commands with high "write" counters, exceptionally large volumes of data being loaded, etc.).
* Look for, and accordingly take care of, possible performance problems on the **Mirroring** partner (hardware problems – storage / RAM, stress caused by processes other than mirroring on the server, etc.).
* Look for, and accordingly take care of possible **network** problems between the two servers (check ping response time, etc.).

### Response B to Case 5: Change Mode to High Performance

In the extreme case where all other options are exhausted, it's possible to do changes to the Mirroring session itself.  
One thing that should *directly* solve the "Mirror commit overhead" problem, is **changing the mirroring mode to "High Performance"**. This can be done by running the following command on the Principle server:

ALTER DATABASE MyProductionDB SET SAFETY OFF

**Warning: This will disable *both* automatic *and* manual failovers.**

To return back to "**High Safety**" mode, you should run the following command on the Principle server:

ALTER DATABASE MyProductionDB SET SAFETY FULL

This will once again allow automatic and manual failovers.

### Response C to Case 5: Other Extreme Measures

If after the above the problem still persists (which is highly unlikely), that could mean there's a serious problem with the Mirroring session.

In such a case you can try to:

1. [Restart the Endpoints](#_Response_Tool_5:_Restarting Endpoin).
2. [Re-Establish the Mirroring Session](#_Response_Tool_4:_Re-Establish Mirro).
3. [Break the Mirroring Session](#_Response_Tool_3:_Break Mirroring Se) altogether and call for an expert.