|  |  |
| --- | --- |
| simpsonized.jpg | SQL Server Reporting Services  Reporting Services Disaster Recovery |
| Many organizations now rely on their reporting solutions for day-to-day business as much as the underlying OLTP systems. For some organizations, being able to report on data in real time can be as important as the availability of their underlying OLTP systems. So like any important database solution, you need to plan on how to recover your reporting solution quickly and efficiently, to keep your business running smoothly. For most medium and small companies, that will mean using the tools available to you with SQL Standard Edition.  In this article I will show you how to setup a Reporting Services solution that will allow you to quickly switch from one server to another, should your primary report server / report server database fail or should you need to switch your service to allow you to carry out server maintenance tasks, reducing downtime of the live system.  **The problem:**  To implement a disaster recovery solution for **Reporting Services** using tools available within **SQL Server standard Edition,** requiring minimal involvement to restore Reporting Services to an operational state.  **The Solution:**  Using two server built to the same specification both of which have SQL Server and Reporting Services installed, we will build a reporting solution that will utilize database mirroring, that can fail over a Reporting Services solution to the mirror server quickly, efficiently and without the need for human interaction.  (To implement automatic failover, we will need a third server, with SQL 2005 express, which will act as the mirroring service witness).  **Before we begin:**  To shorten this article, I will assume the following:   1. You have server administration right and database admin rights. 2. You have configured 2 servers (ideally identical) with SQL server & Reporting Services Configured & ReportServer and ReportServerTempDB are configured on each server. 3. SQL Server Database Engine and SQL Reporting Services is started on both server. 4. I will put in some details relating to database mirroring, but it’s worth reading up on database mirroring, as your requirements may differ slightly and ultimately, it will give you a better understanding of what we are doing here.   Provided we have the above we can setup the reporting solution DR solution. Here are the key things we are going to do:   1. Initialize the Reporting Services ‘Principle’ and ‘mirror’ Servers setup. 2. Configure the ReportServer mirroring session. 3. Configure the SQL Agent jobs to handle the SSRS switch over 4. Configure the Alerts that will initiate SQL Agent jobs   In this tutorial, I’ll refer to the two servers as **SQLTEST01 (my live server)** and **SQLTEST02 (my backup/failover server).** For the proposed solution we will only need 1 server running the report server service as the report server databases. Here are the steps to setup the solution. Prepare the reporting services First off, we need to stop the reporting service on SQLTEST02 Reporting services:   1. Connect to **SQLTEST02** and open Reporting Service Manager 2. Stop the reporting service     The next, is an important step – we need to backup the Reporting Service Encryption Key   1. Connect to **SQLTEST01** 2. Open **Report Services Configuration** and connect 3. Select **Encryption Keys** 4. Backup the encryption key, by selecting **Backup** entering a file location, and specifying a backup file password.   **Considerations:**  As general rule of practice, if you’re using reporting services, you should already have a backup of the encryption key, however you may want to create a new backup file with a suitable name and place the file in a location that is backed up. Additionally, the password should be something suitable to protect any reports that need to be keep secure.  As I’m starting from scratch I’ll create my backup file **sqltest01-ssrs-encr-key.snk**, with a password **pa$$w0rd** and I’ve placed it in **C:\SQL Server Files\SSRS\** so I can easily identify the file.  Make a note of these as these will be needed later in the failover process.     1. Last of all, copy the encryption key to the initial backup/failover server.   For consistency, I would copy the encryption key backup to **SQLTEST02** and store the file in the same folder structure (i.e. c:\SQL Server Files\SSRS).. Initialize the Reporting Services ‘Principle’ and ‘mirror’ Servers  1. Open SSMS and connect **SQLTEST01** and **SQLTEST02** 2. On **SQLTEST02** expand the database node and delete **ReportServer** and **ReportServerTempDB.**   Next we need to backup the **ReportServer** and **ReportServerTempDB**, on **SQLTEST01** (which is going to be our initial principle server) and restore the database onto **SQLTEST02**  **Important:** when initializing the mirroring session, it’s important that no changes are made and no transactions logged in the database, or the mirror will fail to initialize.   1. Connect to **SQLTEST01** 2. Open a new query window and create a backup script for the databases   To make life easier I created C:\DatabaseBackups on **SQLTEST01** and **SQLTEST02**  BACKUP DATABASE ReportServer  TO DISK = 'C:\Database Backups\ReportServer\_principle.bak'  WITH FORMAT  ,INIT  , NAME = 'ReportServer DB'  ,DESCRIPTION = 'ReportServer DB backup for Principle Mirror initialisation DB'  GO  BACKUP LOG ReportServer  TO DISK = 'C:\Database Backups\ReportServer\_principle.bak'  WITH NAME = 'ReportServer Tx Log'  ,DESCRIPTION = 'ReportServer Log backup for Principle Mirror initialisation'  GO  Repeat for the Report Server Temp DB. (Although by it’s definition the report server temp DB is for temporary information, it is possible this could hold information required when the ReportServer DB is failed over, so subsequently I mirror the **ReportServerTempDB** and fail the database over when the **ReportServer** db is failed over.  BACKUP DATABASE ReportServerTempDB  TO DISK = 'C:\Database Backups\ReportServerTempDB\_principle.bak'  WITH FORMAT  ,INIT  , NAME = 'ReportServerTempDB DB'  ,DESCRIPTION = 'ReportServerTempDB DB backup for Principle Mirror initialisation DB'  GO  BACKUP LOG ReportServerTempDB  TO DISK = 'C:\Database Backups\ReportServerTempDB\_principle.bak'  WITH NAME = 'ReportServerTempDB Tx Log'  ,DESCRIPTION = 'ReportServerTempDB Log backup for Principle Mirror initialisation'  GO   1. Execute the backup commands and verify the backups are successful   Processed 328 pages for database 'ReportServer', file 'ReportServer' on file 1.  Processed 2 pages for database 'ReportServer', file 'ReportServer\_log' on file 1.  BACKUP DATABASE successfully processed 330 pages in 0.304 seconds (8.892 MB/sec).  Processed 3 pages for database 'ReportServer', file 'ReportServer\_log' on file 2.  BACKUP LOG successfully processed 3 pages in 0.034 seconds (0.572 MB/sec).  Processed 176 pages for database 'ReportServerTempDB', file 'ReportServerTempDB' on file 1.  Processed 2 pages for database 'ReportServerTempDB', file 'ReportServerTempDB\_log' on file 1.  BACKUP DATABASE successfully processed 178 pages in 0.207 seconds (7.041 MB/sec).  Processed 3 pages for database 'ReportServerTempDB', file 'ReportServerTempDB\_log' on file 2.  BACKUP LOG successfully processed 3 pages in 0.043 seconds (0.440 MB/sec).   1. Copy the backup files from ReportServer\_principle.bak and ReportServerTempDB\_principle.bak from \\**SQLTEST01\c$\Backup Databases\** to [\\**SQLTEST01\c$\Backup Databases\**](file:///\\SQLTEST01\c$\Backup%20Databases\) 2. Now we need to restore the databases to **SQLTEST02** (the initial mirror Server). **Note:** Here we need to restore the database and transaction logs for **ReportServer** and **ReportServerTempDB**, BUT we need to use **WITH NORECOVERY**   RESTORE DATABASE [ReportServer]  FROM DISK = N'C:\Database Backups\ReportServer\_principle.bak'  WITH FILE = 1  ,MOVE N'ReportServer' TO N'E:\SQL\_Data\ReportServer.mdf'  ,MOVE N'ReportServer\_log' TO N'E:\SQL\_Data\ReportServer\_log.LDF'  ,NORECOVERY  ,NOUNLOAD  ,REPLACE  GO  RESTORE LOG [ReportServer]  FROM DISK = N'C:\Database Backups\ReportServer\_principle.bak'  WITH FILE = 2  ,NORECOVERY  ,NOUNLOAD  GO  RESTORE DATABASE [ReportServerTempDB]  FROM DISK = N'C:\Database Backups\ReportServerTempDB\_principle.bak'  WITH FILE = 1  ,MOVE N'ReportServerTempDB' TO N'E:\SQL\_Data\ReportServerTempDB.mdf'  ,MOVE N'ReportServerTempDB\_log' TO N'E:\SQL\_Data\ReportServerTempDB\_log.LDF'  ,NORECOVERY  ,NOUNLOAD  ,REPLACE  GO  RESTORE LOG [ReportServerTempDB]  FROM DISK = N'C:\Database Backups\ReportServerTempDB\_principle.bak'  WITH FILE = 2  ,NORECOVERY  ,NOUNLOAD Configure the mirroring Sessions Now we can initialize the database mirroring session for **ReportServer** and **ReportServerTempDB.**  **Note:**  For this example, I will configure the Principle and Mirror servers for mirroring, but optionally you can go a step further and configure a witness user, if you want automatic failover.   1. Open a connect to **SQLTEST01** 2. Expand the database node 3. Right-click the database **ReportServer** 4. Select **Tasks > Mirror…** 5. Select **Configure Security**     **Complete the Mirroring Security Wizard as follows:**   1. Select **Next** on the wizard introduction screen 2. Select **No** for the Witness Server (if you have a witness server, you can select yes – alternatively you can configure a witness to participate in the topology at a later date. For the example I’ll leave out the witness server. 3. Acknowledge the **Principle** server settings and select **Next**. Note: if your running multiple instances on a server, you may need to change the mirroring port number if other instances also have mirrored databases. 4. For the mirror instance,    1. select the database server **SQLTEST02**    2. select **Connect**    3. Connect to the database server    4. Once you have verified you can connect, select **Next** 5. Next Specify the service accounts for the **Principle** and **Mirror** servers, if they are different. Alternatively, leave these blank if they are the same.    1. Select **Next** 6. Review the configuration details and Click **Finnish** to complete the process:   **On the principal server instance, sqltest01**   * Modify the following properties of the mirroring endpoint: * Name: Mirroring * Listener Port: 5022 * Encryption: Yes * Role: Partner   **On the mirror server instance, sqltest02**   * Modify the following properties of the mirroring endpoint: * Name: Mirroring * Listener Port: 5022 * Encryption: Yes * Role: Partner  1. On successful completion of the **Configuring Endpoints** screen, click **Close** 2. Start the mirroring session. 3. Click **OK** to exit the Database Properties Mirroring GUI.   Now Repeat the same process for **ReportServerTempDB**  By this point, we have achieved the following:   * Configured **SQLTEST01** as the live reporting server . * **SQLTEST01** Reporting Service is **Started** * **SQLTEST02** Reporting Services is **Stopped** * Mirror the report server databases from **SQLTEST01** to **SQLTEST02**  Configuring SQL Server Alert and Jobs Next we will need to configure the SQL Server alerts and Jobs that will manage the switch over process.  First off, we will configure an SQL server job to force the database role switch from Mirror to Principle.  Secondly, the alerts we configure will be used to pick up the database role change and initiate a relevant job, based on the alert condition met.  Thirdly, the relevant job executed will carry out the necessary task to start the report server service and restore the encryption keys.  Below I’ll set out what jobs we need, their purpose, when to use them (or when they are called), which side of the topology the need to operate from and how to define them. All jobs and alerts should be defined on both servers. RS Failover - Initialisation Jobs  |  |  | | --- | --- | | **Job Name:** | **Reporting Service Switch Live – Maintenance** | | **Purpose:** | Executed from the current principle server, this job forces the **ReportServer & ReportServerTempDB** database mirror partner to become the principle. | | **When to use:** | This job should be used to initiate the service switch to the partner, when maintenance tasks need to be carried out on the current principle, or other needs require that the service run from the partner. e.g. both partner servers are operating as normal, but users cannot connect to the principle, but CAN connect to the backup/failover mirror partner. | | **Definition:** | USE [msdb]  GO  /\*\*\*\*\*\* Object: Job [Switch (Live) Reporting Service – Maintenance] Script Date: 02/25/2010 16:13:02 \*\*\*\*\*\*/  BEGIN TRANSACTION  DECLARE @ReturnCode INT  SELECT @ReturnCode = 0  /\*\*\*\*\*\* Object: JobCategory [Database Mirroring] Script Date: 02/25/2010 16:13:02 \*\*\*\*\*\*/  IF NOT EXISTS (SELECT name FROM msdb.dbo.syscategories WHERE name=N'Database Mirroring' AND category\_class=1)  BEGIN  EXEC @ReturnCode = msdb.dbo.sp\_add\_category @class=N'JOB', @type=N'LOCAL', @name=N'Database Mirroring'  IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback  END  DECLARE @jobId BINARY(16)  EXEC @ReturnCode = msdb.dbo.sp\_add\_job @job\_name=N'Reporting Service Switch Live – Maintenance',  @enabled=1,  @notify\_level\_eventlog=0,  @notify\_level\_email=0,  @notify\_level\_netsend=0,  @notify\_level\_page=0,  @delete\_level=0,  @description=N'Executed from the current principle server, this job forces the ReportServer & ReportServerTempDB database mirror partner to become the principle.',  @category\_name=N'Database Mirroring',  @owner\_login\_name=N'FORTHPORTS\David.Lumley', @job\_id = @jobId OUTPUT  IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback  /\*\*\*\*\*\* Object: Step [Failover ReportServer Database] Script Date: 02/25/2010 16:13:03 \*\*\*\*\*\*/  EXEC @ReturnCode = msdb.dbo.sp\_add\_jobstep @job\_id=@jobId, @step\_name=N'Failover ReportServer Database',  @step\_id=1,  @cmdexec\_success\_code=0,  @on\_success\_action=3,  @on\_success\_step\_id=0,  @on\_fail\_action=2,  @on\_fail\_step\_id=0,  @retry\_attempts=0,  @retry\_interval=0,  @os\_run\_priority=0, @subsystem=N'TSQL',  @command=N'ALTER DATABASE ReportServer SET PARTNER FAILOVER  ',  @database\_name=N'master',  @flags=0  IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback  /\*\*\*\*\*\* Object: Step [Failover ReportServerTempDB] Script Date: 02/25/2010 16:13:04 \*\*\*\*\*\*/  EXEC @ReturnCode = msdb.dbo.sp\_add\_jobstep @job\_id=@jobId, @step\_name=N'Failover ReportServerTempDB',  @step\_id=2,  @cmdexec\_success\_code=0,  @on\_success\_action=1,  @on\_success\_step\_id=0,  @on\_fail\_action=2,  @on\_fail\_step\_id=0,  @retry\_attempts=0,  @retry\_interval=0,  @os\_run\_priority=0, @subsystem=N'TSQL',  @command=N'ALTER DATABASE ReportServerTempDB SET PARTNER FAILOVER  GO',  @database\_name=N'master',  @flags=0  IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback  EXEC @ReturnCode = msdb.dbo.sp\_update\_job @job\_id = @jobId, @start\_step\_id = 1  IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback  EXEC @ReturnCode = msdb.dbo.sp\_add\_jobserver @job\_id = @jobId, @server\_name = N'(local)'  IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback  COMMIT TRANSACTION  GOTO EndSave  QuitWithRollback:  IF (@@TRANCOUNT > 0) ROLLBACK TRANSACTION  EndSave: |  |  |  | | --- | --- | | **Job Name:** | **Reporting Services Force Live – DR** | | **Purpose:** | Executed from backup/failover server, when the Principle server fails, and the mirror partner must be forced live it forces the **ReportServer & ReportServerTempDB** databases on the current backup/failover mirror server to become the principle | | **When to use:** | When the Principle server fails, and the mirror partner must be forced live.  **Note:** it’s important to be aware that iIf your using database mirroring in High-Performance mode, then forcing the service (switching the mirror to become the principle database), could potentially restore the mirror database, with transaction missing, due to not having received the transaction on the current principle.  If your using database mirroring in High-Safety mode, then this is unlikely to be an issue as all transactions should have been written to the mirror, before being committed at principle \*provided that the databases were synchronized\*.  It’s also important to remember that when the failed server is restore, you will have to resume the mirroring sessions or reconfigure the database mirroring session for the databases. (the databases on the failed server, will automatically become the mirror when the server is restored). | | **Definition:** | USE [msdb]  GO  /\*\*\*\*\*\* Object: Job [Switch (Live) Reporting Service – Maintenance] Script Date: 02/25/2010 16:13:02 \*\*\*\*\*\*/  BEGIN TRANSACTION  DECLARE @ReturnCode INT  SELECT @ReturnCode = 0  /\*\*\*\*\*\* Object: JobCategory [Database Mirroring] Script Date: 02/25/2010 16:13:02 \*\*\*\*\*\*/  IF NOT EXISTS (SELECT name FROM msdb.dbo.syscategories WHERE name=N'Database Mirroring' AND category\_class=1)  BEGIN  EXEC @ReturnCode = msdb.dbo.sp\_add\_category @class=N'JOB', @type=N'LOCAL', @name=N'Database Mirroring'  IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback  END  DECLARE @jobId BINARY(16)  EXEC @ReturnCode = msdb.dbo.sp\_add\_job @job\_name=N'Reporting Services Force Live – DR',  @enabled=1,  @notify\_level\_eventlog=0,  @notify\_level\_email=0,  @notify\_level\_netsend=0,  @notify\_level\_page=0,  @delete\_level=0,  @description=N'Executed from the current principle server, this job forces the ReportServer & ReportServerTempDB database mirror partner to become the principle.',  @category\_name=N'Database Mirroring',  @owner\_login\_name=N'FORTHPORTS\David.Lumley', @job\_id = @jobId OUTPUT  IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback  /\*\*\*\*\*\* Object: Step [Failover ReportServer Database] Script Date: 02/25/2010 16:13:03 \*\*\*\*\*\*/  EXEC @ReturnCode = msdb.dbo.sp\_add\_jobstep @job\_id=@jobId, @step\_name=N'Failover ReportServer Database',  @step\_id=1,  @cmdexec\_success\_code=0,  @on\_success\_action=3,  @on\_success\_step\_id=0,  @on\_fail\_action=2,  @on\_fail\_step\_id=0,  @retry\_attempts=0,  @retry\_interval=0,  @os\_run\_priority=0, @subsystem=N'TSQL',  @command=N'ALTER DATABASE ReportServer SET PARTNER FORCE\_SERVICE\_ALLOW\_DATA\_LOSS  ',  @database\_name=N'master',  @flags=0  IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback  /\*\*\*\*\*\* Object: Step [Failover ReportServerTempDB] Script Date: 02/25/2010 16:13:04 \*\*\*\*\*\*/  EXEC @ReturnCode = msdb.dbo.sp\_add\_jobstep @job\_id=@jobId, @step\_name=N'Failover ReportServerTempDB',  @step\_id=2,  @cmdexec\_success\_code=0,  @on\_success\_action=1,  @on\_success\_step\_id=0,  @on\_fail\_action=2,  @on\_fail\_step\_id=0,  @retry\_attempts=0,  @retry\_interval=0,  @os\_run\_priority=0, @subsystem=N'TSQL',  @command=N'ALTER DATABASE ReportServerTempDB SET PARTNER FORCE\_SERVICE\_ALLOW\_DATA\_LOSS  GO',  @database\_name=N'master',  @flags=0  IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback  EXEC @ReturnCode = msdb.dbo.sp\_update\_job @job\_id = @jobId, @start\_step\_id = 1  IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback  EXEC @ReturnCode = msdb.dbo.sp\_add\_jobserver @job\_id = @jobId, @server\_name = N'(local)'  IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback  COMMIT TRANSACTION  GOTO EndSave  QuitWithRollback:  IF (@@TRANCOUNT > 0) ROLLBACK TRANSACTION  EndSave: |  RS Failover Process Control - Jobs  |  |  | | --- | --- | | **Important Note:** | For the  **Reporting Services – Activate RS** & **Reporting Services – De-activate RS**   1. The key function of this job is start/stop the reporting service on the server. As such the executing user needs to have permission to carry out such a task, so consider using a service account, which you will need to define as a credential under security and then add as a proxy account to **Operating System (cmdExec)** under SQL Agent > Proxies.   Below is the T-SQL to set up a credential  USE [master]  GO  CREATE CREDENTIAL [SQLTest.Job] WITH IDENTITY = N'FORTHPORTS\SQLTest.Job', SECRET = N'sql999'  GO  Below is the T-SQL to add the credential as a proxy account to **Operating System (cmdExec)**  USE [msdb]  GO  EXEC msdb.dbo.sp\_add\_proxy @proxy\_name=N'SQLTest.Job',@credential\_name=N'SQLTest.Job',  @enabled=1  GO  EXEC msdb.dbo.sp\_grant\_proxy\_to\_subsystem @proxy\_name=N'SQLTest.Job', @subsystem\_id=3  GO   1. To restore the encryption keys in **Reporting Services – Activate RS** is a OS command line, that will be executed by the job. The command requires to encryption key password, which is in clear text, so be aware that anyone who can view/edit the job can also see the encryption key password. As such, you may want to consider who has access to the job if encryption key security is important. |  |  |  | | --- | --- | | **Job Name:** | **Reporting Services – Activate RS** | | **Purpose:** | When triggered by the alert to execute, this job carries out the necessary tasks to switch the reporting service onto the current machine. These will include:   1. Starting the reporting service on the current machine 2. Delay for 10 seconds to allow RS to carry out necessary start up tasks. (Note: I initially tried this without the delay, but it always failed). 3. Deletes Reporting Service instance keys from keys from the **client** table in **ReportServer** Database. (this is important as otherwise RS think that we are trying to set up a RS web farm.) 4. Restoring the encryption keys | | **When to use:** | Automatically called when an information alert is raised where the **ReportServer** database is set to the ***Principle*** role on the current machine. | | **Definition:** | USE [msdb]  GO  /\*\*\*\*\*\* Object: Job [Reporting Services - Activate RS] Script Date: 03/01/2010 14:13:42 \*\*\*\*\*\*/  BEGIN TRANSACTION  DECLARE @ReturnCode INT  SELECT @ReturnCode = 0  /\*\*\*\*\*\* Object: JobCategory [[Uncategorized (Local)]]] Script Date: 03/01/2010 14:13:42 \*\*\*\*\*\*/  IF NOT EXISTS (SELECT name FROM msdb.dbo.syscategories WHERE name=N'[Uncategorized (Local)]' AND category\_class=1)  BEGIN  EXEC @ReturnCode = msdb.dbo.sp\_add\_category @class=N'JOB', @type=N'LOCAL', @name=N'[Uncategorized (Local)]'  IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback  END  DECLARE @jobId BINARY(16)  EXEC @ReturnCode = msdb.dbo.sp\_add\_job @job\_name=N'Reporting Services - Activate RS',  @enabled=1,  @notify\_level\_eventlog=0,  @notify\_level\_email=0,  @notify\_level\_netsend=0,  @notify\_level\_page=0,  @delete\_level=0,  @description=N'Starts SSRS on the local machine. Initialises the instance and restore the data encryption keys to the instance.',  @category\_name=N'[Uncategorized (Local)]',  @owner\_login\_name=N'FORTHPORTS\SQLTest.Job', @job\_id = @jobId OUTPUT  IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback  /\*\*\*\*\*\* Object: Step [Start Reporting Services] Script Date: 03/01/2010 14:13:42 \*\*\*\*\*\*/  EXEC @ReturnCode = msdb.dbo.sp\_add\_jobstep @job\_id=@jobId, @step\_name=N'Start Reporting Services',  @step\_id=1,  @cmdexec\_success\_code=0,  @on\_success\_action=3,  @on\_success\_step\_id=0,  @on\_fail\_action=2,  @on\_fail\_step\_id=0,  @retry\_attempts=0,  @retry\_interval=0,  @os\_run\_priority=0, @subsystem=N'CmdExec',  @command=N'net start ReportServer',  @flags=0,  @proxy\_name=N'SQLTest.Job'  IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback  /\*\*\*\*\*\* Object: Step [Wait 10 seconds] Script Date: 03/01/2010 14:13:42 \*\*\*\*\*\*/  EXEC @ReturnCode = msdb.dbo.sp\_add\_jobstep @job\_id=@jobId, @step\_name=N'Wait 10 seconds',  @step\_id=2,  @cmdexec\_success\_code=0,  @on\_success\_action=3,  @on\_success\_step\_id=0,  @on\_fail\_action=2,  @on\_fail\_step\_id=0,  @retry\_attempts=0,  @retry\_interval=0,  @os\_run\_priority=0, @subsystem=N'TSQL',  @command=N'WAITFOR DELAY ''00:00:10''',  @database\_name=N'master',  @flags=0  IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback  /\*\*\*\*\*\* Object: Step [Delete Report Server Keys] Script Date: 03/01/2010 14:13:42 \*\*\*\*\*\*/  EXEC @ReturnCode = msdb.dbo.sp\_add\_jobstep @job\_id=@jobId, @step\_name=N'Delete Report Server Keys',  @step\_id=3,  @cmdexec\_success\_code=0,  @on\_success\_action=3,  @on\_success\_step\_id=0,  @on\_fail\_action=2,  @on\_fail\_step\_id=0,  @retry\_attempts=0,  @retry\_interval=0,  @os\_run\_priority=0, @subsystem=N'TSQL',  @command=N'DELETE FROM Keys WHERE client > -1',  @database\_name=N'ReportServer',  @flags=0  IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback  /\*\*\*\*\*\* Object: Step [Restore Report Server Encryption Key] Script Date: 03/01/2010 14:13:43 \*\*\*\*\*\*/  EXEC @ReturnCode = msdb.dbo.sp\_add\_jobstep @job\_id=@jobId, @step\_name=N'Restore Report Server Encryption Key',  @step\_id=4,  @cmdexec\_success\_code=0,  @on\_success\_action=1,  @on\_success\_step\_id=0,  @on\_fail\_action=2,  @on\_fail\_step\_id=0,  @retry\_attempts=1,  @retry\_interval=1,  @os\_run\_priority=0, @subsystem=N'CmdExec',  @command=N'echo Y | "C:\Program Files (x86)\Microsoft SQL Server\90\Tools\binn\RSKeymgmt.exe" -a -f "C:\SQL Server Files\SSRS\sqltest01-ssrs-encr-key.snk" -p "pa$$w0rd"',  @flags=0,  @proxy\_name=N'SQLTest.Job'  IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback  EXEC @ReturnCode = msdb.dbo.sp\_update\_job @job\_id = @jobId, @start\_step\_id = 1  IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback  EXEC @ReturnCode = msdb.dbo.sp\_add\_jobserver @job\_id = @jobId, @server\_name = N'(local)'  IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback  COMMIT TRANSACTION  GOTO EndSave  QuitWithRollback:  IF (@@TRANCOUNT > 0) ROLLBACK TRANSACTION  EndSave: |  |  |  | | --- | --- | | **Job Name:** | **Reporting Services – De-activate RS** | | **Purpose:** | When triggered by the alert to execute, this job carries out the necessary tasks to switch the reporting service onto the current machine. This will be to stop reporting services | | **When to use:** | Automatically called when an information alert is raised where the **ReportServer** database is set to the ***Mirror*** role on the current machine. | | **Definition:** | USE [msdb]  GO  /\*\*\*\*\*\* Object: Job [Switch (Stanby) - ReportServer] Script Date: 02/26/2010 16:50:50 \*\*\*\*\*\*/  BEGIN TRANSACTION  DECLARE @ReturnCode INT  SELECT @ReturnCode = 0  /\*\*\*\*\*\* Object: JobCategory [[Uncategorized (Local)]]] Script Date: 02/26/2010 16:50:51 \*\*\*\*\*\*/  IF NOT EXISTS (SELECT name FROM msdb.dbo.syscategories WHERE name=N'[Uncategorized (Local)]' AND category\_class=1)  BEGIN  EXEC @ReturnCode = msdb.dbo.sp\_add\_category @class=N'JOB', @type=N'LOCAL', @name=N'[Uncategorized (Local)]'  IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback  END  DECLARE @jobId BINARY(16)  EXEC @ReturnCode = msdb.dbo.sp\_add\_job @job\_name=N'Switch (Stanby) - ReportServer',  @enabled=1,  @notify\_level\_eventlog=0,  @notify\_level\_email=0,  @notify\_level\_netsend=0,  @notify\_level\_page=0,  @delete\_level=0,  @description=N'No description available.',  @category\_name=N'[Uncategorized (Local)]',  @owner\_login\_name=N'FORTHPORTS\sqlnordic.svc', @job\_id = @jobId OUTPUT  IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback  /\*\*\*\*\*\* Object: Step [Stop Reporting Services] Script Date: 02/26/2010 16:50:52 \*\*\*\*\*\*/  EXEC @ReturnCode = msdb.dbo.sp\_add\_jobstep @job\_id=@jobId, @step\_name=N'Stop Reporting Services',  @step\_id=1,  @cmdexec\_success\_code=0,  @on\_success\_action=1,  @on\_success\_step\_id=0,  @on\_fail\_action=2,  @on\_fail\_step\_id=0,  @retry\_attempts=0,  @retry\_interval=0,  @os\_run\_priority=0, @subsystem=N'CmdExec',  @command=N'net stop ReportServer$POT\_NORDIC',  @flags=0,  @proxy\_name=N'sqlnordic.job'  IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback  EXEC @ReturnCode = msdb.dbo.sp\_update\_job @job\_id = @jobId, @start\_step\_id = 1  IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback  EXEC @ReturnCode = msdb.dbo.sp\_add\_jobserver @job\_id = @jobId, @server\_name = N'(local)'  IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback  COMMIT TRANSACTION  GOTO EndSave  QuitWithRollback:  IF (@@TRANCOUNT > 0) ROLLBACK TRANSACTION  EndSave: | |  |  |  RS Failover Process Control - Alerts  |  |  | | --- | --- | | **Important Note:** | If you want to define your alerts by using T-SQL as I have done, then before define the alerts on each server, you will need to find out what the Job ID is for each each.  You can find this by looking at the sysjob table in the msdb database. (**Note: Job ID will be different on each machine!)**  SELECT [job\_id] ,[name] ,[description] FROM [msdb].[dbo].[sysjobs]  where [name] like 'Reporting Services%'  For defining the alerts, you may find it easier and quicker to use the GUI in SSMS, as you can select the job from a pull down list. |  |  |  | | --- | --- | | **Alert Name:** | **ReportServerTempDB – Switched to Principle** | | **Purpose:** | This alert is to provide a ‘trigger’ of sorts, to kick off the **Reporting Services – Activate RS** job, which will start the reporting service on the server and restore the encryption keys. | | **When to use:** | Automatically triggered when the event description keyword is found in the SQL Server event log. | | **Definition:** | USE [msdb]  GO  /\*\*\*\*\*\* Object: Alert [ReportServerTempDB – Switched to Principle] Script Date: 03/01/2010 13:07:52 \*\*\*\*\*\*/  EXEC msdb.dbo.sp\_add\_alert @name=N'ReportServerTempDB – Switched to Principle',  @message\_id=0,  @severity=10,  @enabled=1,  @delay\_between\_responses=0,  @include\_event\_description\_in=1,  @event\_description\_keyword=N'"ReportServerTempDB" is changing roles from "MIRROR" to "PRINCIPAL" due to Failover from partner',  @category\_name=N'[Uncategorized]',  @job\_id=N'5ebac8b7-ed3f-4569-920f-46f1c059b8fb |  |  |  | | --- | --- | | **Alert Name:** | **ReportServerTempDB – Forced to Principle** | | **Purpose:** | This alert is to provide a ‘trigger’ of sorts, to kick off the **Reporting Services – Activate RS** job, which will start the reporting service on the server and restore the encryption keys. | | **When to use:** | Automatically triggered when the event description keyword is found in the SQL Server event log. | | **Definition:** | USE [msdb]  GO  /\*\*\*\*\*\* Object: Alert [ReportServerTempDB – Forced to Principle] Script Date: 03/01/2010 13:06:50 \*\*\*\*\*\*/  EXEC msdb.dbo.sp\_add\_alert @name=N'ReportServerTempDB – Forced to Principle',  @message\_id=0,  @severity=10,  @enabled=1,  @delay\_between\_responses=0,  @include\_event\_description\_in=1,  @event\_description\_keyword=N'"ReportServerTempDB" is changing roles from "MIRROR" to "PRINCIPAL" due to Force\_Service\_Allow\_Data\_',  @category\_name=N'[Uncategorized]',  @job\_id=N'5ebac8b7-ed3f-4569-920f-46f1c059b8fb' |  |  |  | | --- | --- | | **Alert Name:** | **ReportServerTempDB – Switched to Mirror** | | **Purpose:** | This alert is to provide a ‘trigger’ of sorts, to kick off the **Reporting Services – De-activate RS** job, which will stop the reporting service on the server. | | **When to use:** | Automatically triggered when the event description keyword is found in the SQL Server event log. | | **Definition:** | USE [msdb]  GO  /\*\*\*\*\*\* Object: Alert [ReportServerTempDB – Switched to Mirror] Script Date: 03/01/2010 13:07:24 \*\*\*\*\*\*/  EXEC msdb.dbo.sp\_add\_alert @name=N'ReportServerTempDB – Switched to Mirror',  @message\_id=0,  @severity=10,  @enabled=1,  @delay\_between\_responses=0,  @include\_event\_description\_in=1,  @event\_description\_keyword=N'"ReportServerTempDB" is changing roles from "PRINCIPAL" to "MIRROR" due to Failover.',  @category\_name=N'[Uncategorized]',  @job\_id=N'd2a3fb16-63d7-44ff-9ae0-0d0396e85e5d' |   Now you can test a switch over by starting the **Reporting Service Switch Live – Maintenance** on SQLTEST01.  After a short delay the Reporting service should stop on SQLTEST01 and be fully restored on SQLTEST02.  If you want to build in automated failover, then you’ll need to   1. configure a witness server 2. Configure the mirroring endpoint on the witness server (this will be done for you if you add a witness, when you setup database mirroring) 3. Pause Database mirroring, Add the witness server to the database mirror session and the resume mirroring. 4. You’ll need to configure an additional alert with a response to execute the job :  **Reporting Services – Activate RS**   This alert should only be defined if you intend to use a witness server for automatic failover.   |  |  |  | | --- | --- | --- | | **Alert Name:** | **ReportServerTempDB –Switched to Principle (AUTO)** |  | | **Purpose:** | This alert is to provide a ‘trigger’ of sorts, to kick off the **Reporting Services – Activate RS** job, which will start the reporting service on the server and restore the encryption keys. |  | | **When to use:** | Automatically triggered when the event description keyword is found in the SQL Server event log. |  | | **Definition:** | USE [msdb]  GO  /\*\*\*\*\*\* Object: Alert [ReportServerTempDB – Forced to Principle] Script Date: 02/26/2010 16:35:03 \*\*\*\*\*\*/  EXEC msdb.dbo.sp\_add\_alert @name=N'ReportServerTempDB – Forced to Principle',  @message\_id=0,  @severity=10,  @enabled=1,  @delay\_between\_responses=0,  @include\_event\_description\_in=1,  @event\_description\_keyword=N'"ReportServerTempDB" is changing roles from "MIRROR" to "PRINCIPAL" due Auto Failover',  @category\_name=N'[Uncategorized]'  --,@job\_id=N'00000000-0000-0000-0000-000000000000' |  |  Conclusion Once the jobs and alerts have been setup, the automation process is in place, an can follow 2 tracks:   |  | | --- | | Maintenance Failover | | This method is used if you need to failover the reporting service due to required maintenance on the current live server, when both servers are up. |  |  |  |  | | --- | --- | --- | |  | Current Principle Server (e.g. SQLTEST01) | Current Mirror Server (e.g. SQLTEST02) | |  | SSRS Started  ReportServer Database: Principle  ReportServerTempDB Database: Principle | SSRS Stopped  ReportServer Database: Mirror  ReportServerTempDB Database: Mirror | | **User Steps** | | | |  | Job Reporting Service Switch Live – Maintenance executed by server admin |  | | **The steps below are automatic** | | | |  | ReportServer & ReportServerTempDB, become mirror databases | ReportServer & ReportServerTempDB, become Principle databases | |  | Alert condition ReportServerTempDB – Switched to Mirror met | Alert condition ReportServerTempDB – Switched to Principle met | |  | Job Reporting Services – De-activate RS invoked by alert and executed using proxy account.   1. SSRS Service Stopped on the machine. | Job Reporting Services – Activate RS invoked by alert and executed using proxy account.   1. SSRS Service started on the machine 2. Process delayed for 10 seconds 3. Instance keys from the client table in ReportServer Database deleted (this is important as otherwise RS think that we are trying to set up a RS web farm.) 4. SSRS Encryption key restore to SSRS | |  | SSRS Stopped  ReportServer Database: Mirror  ReportServerTempDB Database: Mirror | SSRS Started  ReportServer Database: Principle  ReportServerTempDB Database: Principle |  |  |  |  | | --- | --- | --- | | **Forced Failover** | | | | This method is for forcing the reporting service and databases onto the backup server, due to a failure on the primary where the database server is unavailable. | | | | Initiate by running the Job on the current mirror database server. | | | |  | | | |  | | | |  | Current Principle Server (e.g. SQLTEST01) | Current Mirror Server (e.g. SQLTEST02) | | |  | Server Unavailable | SSRS Stopped  ReportServer Database: Mirror  ReportServerTempDB Database: Mirror | | | **User Steps** | | | | |  |  | Job Reporting Services Force Live – DR executed by server admin | | | **The steps below are automatic** | | | | |  |  | ReportServer & ReportServerTempDB, become Principle databases using the FORCE SERVICE ALLOW DATA LOSS command. | | |  |  | Alert condition ReportServerTempDB – Forced to Principle met | | |  |  | Job Reporting Services – Activate RS invoked by alert and executed using proxy account.   1. SSRS Service started on the machine 2. Process delayed for 10 seconds 3. Instance keys from the client table in ReportServer Database deleted (this is important as otherwise RS think that we are trying to set up a RS web farm.) 4. SSRS Encryption key restore to SSRS | | |  |  | SSRS Started  ReportServer Database: Principle  ReportServerTempDB Database: Principle | | |  | Important!: When the server is restored, the ReportServer & ReportServerTempDB will automatically be reassigned to be the mirror databases |  | | | **User Steps** | | | | |  |  | On the database mirroring page, you must resume mirroring (if appropriate) for both the ReportServer and ReportServerTempDB databases.    You will be prompted to acknowledge the following potential loss of transactions. (this theoretically isn’t an issue if you are using DB Mirroring in high protection mode).    Alternatively – reconfigure the database mirroring. | |   And that is it!  On my test servers, the switch over took literally a second. So this is a fast and effective method! | |
|  | |