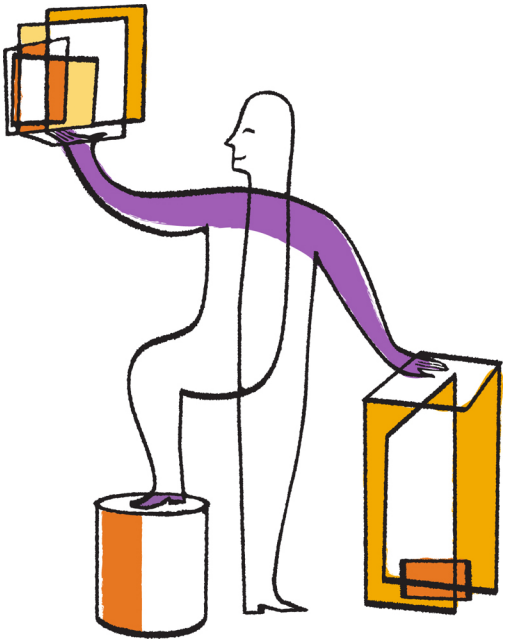
SnapManager® 7.2 for Microsoft® SQL Server®

Administration Guide



NetApp, Inc.
495 East Java Drive
Sunnyvale, CA 94089
U.S.

Telephone: +1 (408) 822-6000
Fax: +1 (408) 822-4501
Support telephone: +1 (888) 463-8277
Web: www.netapp.com
Feedback: doccomments@netapp.com

Part number: 215-09640_A0
January 2015

Contents

Product overview **6**
Backing up and verifying your databases **8**
SnapManager backup overview 8
Two ways that SnapManager performs full database backups 9
How SnapManager updates the SnapInfo directory 10
How SnapManager checks database integrity in backup sets 10
Prerequisites for VMDK verification or cloning on SnapMirror destination volumes 13
Defining a backup strategy 15
Backing up your databases for the first time 18
Verifying the initial backup set 20
Scheduling recurring backups 21
Scheduling recurring transaction log backups 22
Scheduling recurring backup set verifications 24
Managing backup retention 25
Maximum number of Snapshot copies per volume 25
Automatically deleting backups 26
Explicitly deleting backups 27
Managing Availability Group transaction log backups 29
Changing the backup management group of an existing backup set 29
What to do if a SnapManager backup operation fails 30
Restoring databases **34**
How SnapManager a restore operation works 34
Types of SnapManager restore operations 36
Sources and destinations for a SnapManager restore 38
Transaction log backups from SQL Server Management Studio 39
Post-restore database recovery states 40
Requirements for restoring a database 40
Finding backup sets 41
Restoring a database from a local backup set 41
Restoring a database from a backup set created on a different server 44
Restoring replicated, publisher, and subscriber databases 46

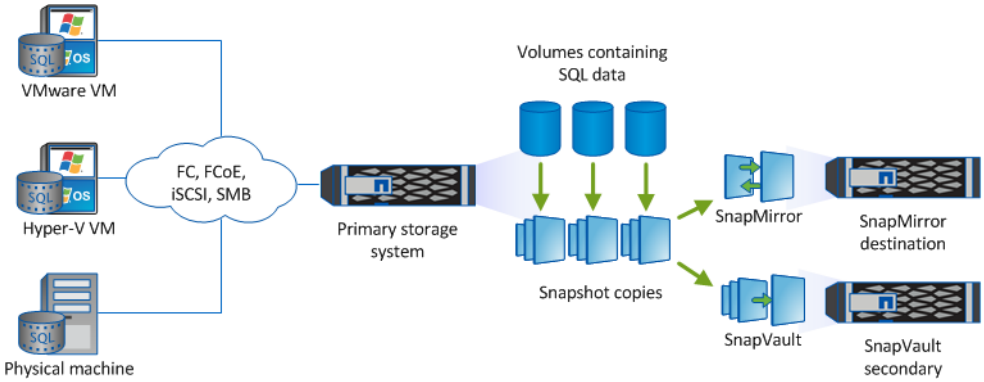
Reseeding a database on an AlwaysOn cluster 47
Recovering databases using archived backup sets 48
Recovering databases using SnapMirror 48
Recovering databases on VMDKs using SnapMirror 51
Preparing the primary site for recovery 51
Preparing the secondary site for recovery 51
Recovering databases from the secondary site 52
Cloning databases 54
Cloning limitations for VMDKs 54
Prerequisites for VMDK verification or cloning on SnapMirror destination volumes 55
Cloning a database from a local backup or an archived backup 57
Cloning a database that is in production 59
Creating a clone replica of an AlwaysOn cluster 60
Cloning an already cloned database 60
Splitting a cloned database 62
Deleting cloned databases 63
Using SnapManager reports 64
Viewing SnapManager reports 64
Configuring monitoring and reporting settings 65
Changing the location of the SnapManager report directory 66
Modifying your database configuration on NetApp storage 68
Moving multiple SnapInfo directories to a single SnapInfo directory 68
Migrating SQL Server databases back to local disks 69
Setting up a SnapManager share for centralized backups of transaction logs 70
Importing or exporting database configurations using a control file 71
Sample XML schema for the control file settings 72
Configuring SnapManager application settings 85
Modifying backup settings 85
Modifying verification settings 86
Modifying restore settings 87
Modifying event notification settings 89
Setting defaults for preoperation and postoperation commands 89
SnapManager arguments for preoperation and postoperation commands 90
Enabling SnapManager to allow databases on any LUN or VMDK configuration 95

Viewing fractional space reservation status 96
Configuring fractional space reservation policies 98
Advanced administration 100
Maximum configurations supported by SnapManager 100
Service account requirements for archiving backup sets with SnapVault (7-Mode environments only) 101
SnapManager disk requirements in a Windows cluster using LUNs 102
Upgrading SnapManager 104
Upgrading SnapManager interactively 104
Upgrading SnapManager from a command line 104
Repairing, reinstalling, and uninstalling SnapManager 106
Repairing SnapManager 106
Reinstalling SnapManager 106
Uninstalling SnapManager 107
SnapManager cmdlet guidelines 108
clone-backup 109
clone-database 116
clone-replica 126
delete-backup 135
delete-clone 137
export-config 139
get-backup 141
import-config 143
new-backup 146
reseed-backup 156
restore-backup 161
split-clone 169
verify-backup 171
Copyright information 178
Trademark information 179
How to send comments about documentation and receive update notification 180
Index 181

Product overview

SnapManager for Microsoft SQL Server is a host-side component of NetApp's integrated storage solution for SQL Server, offering application-aware primary Snapshot copies of SQL databases. You can use SnapManager with NetApp's SnapMirror technology to create mirror copies of backup sets on another volume, and with NetApp's SnapVault product to archive backups efficiently to disk.

Together these tools offer a complete Snapshot-based data protection scheme, as scalable, reliable, and highly available as the underlying storage system. The following illustration shows the components in a SnapManager deployment with clustered Data ONTAP:



SnapManager highlights

SnapManager features seamless integration with Microsoft products on the Windows host and with NetApp Snapshot technology on the back end. It offers an easy-to-use, wizard-based administrative interface.

- *Integration with the Microsoft Volume Shadow Copy Service (VSS)* ensures that write requests are frozen and write caches flushed before backups are taken. SnapManager provides full support for Windows Volume Manager, Windows Server Failover Clustering, Microsoft Multipath I/O (MPIO), and SQL Server AlwaysOn Availability Groups.
- *Fast, nondisruptive Snapshot technology* using NetApp's SnapDrive for Windows (SDW) software lets you back up databases in seconds and restore them in minutes without taking SQL Servers offline. Snapshot copies consume minimal storage space. You can store up to 255 copies per volume.
- *Automated central administration* offers hands-off, worry-free data management. You can schedule routine SQL Server database backups, configure policy-based backup retention, set up point-in-time and up-to-the-minute restores, and proactively monitor your SQL Server

environment with periodic email alerts. PowerShell cmdlets are available for easy scripting of backup and restore operations.

In addition to these major features, SnapManager offers the following:

- Integrated FlexClone software lets you create space-efficient point-in-time copies of production databases for testing or data extraction (FlexClone license required)
- Simplified migration of existing databases to NetApp storage with an easy-to-use Configuration wizard
- Nondisruptive, automated backup verification
- Fast reseeding of databases in an AlwaysOn cluster
- Federated database backup of multiple SQL Server instances and databases
- Support for backup of LUNs, SMB shares, and VMDKs
- Support for physical and virtualized infrastructures
- Support for iSCSI, Fibre Channel, FCoE, RDM, and VMDK over NFS and VMFS

Backing up and verifying your databases

You should back up your databases as soon as they are available in NetApp storage. You can then verify the initial backups and schedule recurring backups and recurring backup verifications.

Note: For information on how to install SnapManager on Windows hosts and how to set up NetApp storage for SnapManager usage, see the *SnapManager for Microsoft SQL Server Installation and Setup Guide*.

Related information

SnapManager 7.2 for Microsoft SQL Server Installation and Setup Guide For Clustered Data ONTAP

SnapManager 7.2 for Microsoft SQL Server Installation and Setup Guide For Data ONTAP Operating in 7-Mode

SnapManager backup overview

SnapManager uses NetApp Snapshot technology to create online, read-only copies of databases. It uses an SQL Server utility to verify the integrity of the backups.

SnapManager backs up a database by creating Snapshot copies of the volumes in which the following reside:

- Database data files
- Transaction logs
- SnapInfo directories

Together these Snapshot copies comprise a *backup set*. SnapManager uses a backup set to restore a database.

After SnapManager backs up your databases, it can perform an integrity verification of the backup sets. SnapManager uses the Database Consistency Checker (DBCC), a Microsoft SQL Server utility, to verify the page-level integrity of databases. Verification ensures that you can use backup sets to restore databases as needed.

Important: SnapManager cannot restore databases from Snapshot copies created by Data ONTAP or SnapDrive. You should perform backups using SnapManager only.

Two ways that SnapManager performs full database backups

SnapManager uses two methods to back up your databases: a stream-based backup method and an online Snapshot copy backup method. The differences in these back-up methods underscores why you should not place system databases on the same volume as user databases.

Stream-based backup method

SnapManager uses the stream-based method to back up system databases. With this method, SnapManager creates the full database backup by streaming the contents of the databases over the network to the SnapInfo directory. SnapManager copies each system database individually.

Full database stream-based backup files are `.fbk` files named using the convention *date_time_databasename*: for example, `050814_0330_xxx.fbk`. This file is equivalent to the `.bak` file directly created by SQL Server.

All other database backups use the *online Snapshot copy* backup method.

Online Snapshot copies backup method

SnapManager uses the online Snapshot copy backup method to backup user databases that reside on LUNs, VMDKs, and SMB shares. With this method, SnapManager creates the backup by creating Snapshot copies of the databases.

When you select a database for a full database backup, SnapManager automatically selects all other databases that reside on the same storage system volume. You can clear databases that reside on a different LUN, SMB share, or VMDK from the databases you selected, even if the LUN, SMB share, or VMDK is on the same storage volume.

If the other LUN, SMB share, or VMDK stores only a single database, you can clear or reselect that database individually. If the other LUN, SMB share, or VMDK houses multiple databases, you must clear or reselect those databases as a group.

In a volume-wide backup, all the databases that reside on a single volume are backed up *concurrently* using Snapshot copies. If the maximum number of concurrent backup databases is 35, then the total number of Snapshot copies created equals the number of databases divided by 35.

> **Note:** When a Snapshot copy is made for a SnapManager backup, the entire storage system volume is captured in that Snapshot copy; however, that backup is valid only for the SQL host server for which the backup was created.
>
> If data from other SQL host servers resides on the same volume, that data is not restorable from the Snapshot copy.

How SnapManager updates the SnapInfo directory

The SnapInfo directory stores information about the streaming-based backups of system databases, copies of transaction log files, and a backup set's metadata. SnapManager updates the directory every time that it creates a backup set.

Every time a SnapManager backup set is created, SnapManager creates a new backup set subdirectory under the SnapInfo directory. SnapManager populates this subdirectory with the transaction logs backed up as part of that backup set in addition to the recovery information for that specific Snapshot copy. A complete backup set consists of this SnapInfo subdirectory and the corresponding Snapshot copies of the LUNs, SMB shares, or VMDKs that store the SQL Server databases.

SnapManager subdirectory names identify the configuration of the backed-up databases:

Configuration	Format of the SnapInfo subdirectory name
Databases belonging to the SQL Server default instance	The SnapInfo directory name is `SQL__` followed by the SQL Server computer host name: `SQL__`*`SqlServerName`* For example, the subdirectory for databases belonging to the default instance of the SQL Server on the Windows host system `CLPUBS-WINSRVR3` would be named as follows: `SQL__ CLPUBS-WINSRVR3`
Databases belonging to an SQL Server named instance	The SnapInfo directory name is `SQL__` followed by the name of the SQL Server instance: `SQL__`*`InstanceName`* For example, the subdirectory for databases that belong to the SQL Server instance `INST2` on the Windows host system `ENGR-WINSRVR7` would be named as follows: `SQL__INST2`

How SnapManager checks database integrity in backup sets

SnapManager uses Database Consistency Checker (DBCC) to verify SQL Server databases. DBCC is a Microsoft SQL Server utility that verifies the page-level integrity of databases.

Ways that SnapManager uses SQL Server DBCC

SnapManager uses the `DBCC CHECKDB` command to verify the integrity of live databases in addition to databases in SnapManager backup sets.

Live databases can be verified as a part of database migration and also as a part of a full database backup:

- Using the Configuration wizard, you can verify live databases before and after database migration.
- Using SnapManager Backup, you can verify live databases before and after a full database backup.

To perform live verification of databases residing on SMB shares, you must configure the verification settings in the SnapManager console to use the TABLOCK option.

Databases in backup sets can be verified on creation, separately, or before a restore operation:

- Using SnapManager Backup, you can verify the databases in full database backup sets as they are created or you can verify the databases in the most recent unverified backup sets.
- Using SnapManager Restore, if you select a backup set on which a consistency check has not been run successfully, SnapManager prompts (but does not require) you to first verify the databases in that backup set.

Requirements for running SQL Server DBCC against the databases in a backup set

When you verify the databases in a backup set (as opposed to live databases), Microsoft DBCC requires that all the database files be mounted at the same time. At a more granular level, this means that SnapManager, using SnapDrive commands, mounts all the LUNs or VMDKs that contain the backup sets selected for database verification.

To run the `DBCC CHECKDB` command, the verification server (whether local or remote) must have a sufficient number of drive letters available or a mount point to mount all the LUNs or VMDKs storing the database backup sets that you are verifying.

- When you run database verification against backup sets that are stored on a single LUN or VMDK, the SQL Server host that is used as the verification server must have at least one drive letter available or a mount point so that the LUN or VMDK can be mounted during database verification.
- When you run database verification against backup sets that contain multiple database files stored on separate LUNs or VMDKs, SnapManager mounts all those LUNs or VMDKs at the same time.

  Consequently, the SQL Server that is used as the verification server must have enough drive letters available so that SnapManager can mount each of the LUNs or VMDKs simultaneously.

  For example, suppose you want to run database integrity verification against backup sets containing five file groups using three transaction logs stored on eight separate LUNs or VMDKs. In this case, the verification server would need to have a minimum of eight drive letters or a mount point available.

Note: Available drive letters are not required if the databases reside on SMB shares.

Ways to separate database verification from database backup

Running database verification on a production SQL Server is CPU-intensive for the host running the verification and also involves a substantial amount of activity on the storage system. For this reason, verification can degrade SQL Server response, particularly during peak usage.

By default, a SnapManager full database backup operation runs DBCC immediately after the backup is complete. However, SnapManager provides two options that enable you to separate the process of verification from the backup itself: deferred database verification and remote database verification.

Instead of allowing a full database backup to automatically verify the databases when the operation is complete, you can disable that feature. You can then run a separate database verification operation any time after the full database backup operation is complete.

To prevent database verification from affecting the performance of your production SQL Server computer, you can run verification on another SQL Server computer.

Options for when to verify the databases in a backup set

You can verify the databases in your SnapManager backup sets at various times:

- Automatically verify full database backup sets on creation
  By default, SnapManager verifies the databases in a backup set at the time the backup is created. This is simple and ensures that each database in the backup set is verified. However, this method significantly increases the time required to complete the backup.
- Explicitly start or schedule database verification only
  With this method, a single operation can be initiated to verify the databases contained in one or more backup sets that have already been created. You can start the verification immediately, or you can schedule the verification to occur later, when it does not affect performance or delay later backups.
- Defer verification until you restore from the backup set
  If you attempt to restore from a backup set on which a database consistency check has not been run successfully, SnapManager prompts (but does not require) you to first verify the databases in that backup set.

Options for where to run SQL Server DBCC

Regardless of when you verify the databases in a backup set, the verification can be done on the production SQL Server (the Windows host system running the SQL Server instance used to create the databases) or on a remote verification system (another SQL Server).

In the simplest SnapManager configuration, verification is run from the production SQL Server. However, because the Microsoft DBCC command used for the verification is CPU-intensive, performing the verification on the production SQL Server host system during peak usage could affect SQL Server performance.

Performing the verification on a remote system minimizes the impact of verification on SQL Server system resources and backup schedule. The requirements for a remote verification server are described in the *SnapManager for Microsoft SQL Server Installation and Setup Guide*.

Prerequisites for VMDK verification or cloning on SnapMirror destination volumes

You can verify backup sets on SnapMirror destination volumes and you can clone databases from SnapMirror destination volumes. If the databases that you want to verify or clone are hosted on VMDKs, you must meet several prerequisites before you can perform either of those operations.

You can verify and clone from destination volumes when the database hosted on the VMDKs is replicated to a site by SnapMirror and the configuration meets the following requirements:

- The virtual machine is installed on the ESX server on the secondary site.
- SQL Server, SnapDrive, and SnapManager are installed on the virtual machine.
- The ESX server is managed by another vCenter Server and the VSC server on the secondary site.
- SnapDrive is installed on the secondary virtual machine that is pointing to the VSC server on the secondary site.
- On the primary site, you have selected the SQL Server on the secondary site as the remote verification server.
- On both the primary and secondary VSC servers, you have created a Windows share on the VSC repository folder where the backup metadata file resides.
- The SnapManager service account has read permission on the share at the primary site and write and modify permissions at the share on the secondary site.
- The primary VSC server has discovered the destination storage system.
- For NFS datastores residing on clustered Data ONTAP, a datastore must exist on the SnapMirror destination volume and the name of the destination datastore must be specified in the change list file.
- On the primary virtual machine where the backup is initiated, the following registry settings and values must be defined in `HKEY_LOCAL_MACHINE\SOFTWARE\Network Appliance \SnapManager for SQL Server\Server`:

  SMVITransformEnable

  dword:00000001

  SMVITransformScript

  SMVI_Metadata_update.exe

  SMVIDestinationServer

destination SMVI server name

SMVISourceBackupXmlUNC

*source SMVI server name**SMVI repository share name*\backups.xml

SMVIDestinationBackupXmlUNC

*destination SMVI server name**SMVI repository share name*\backups.xml

SMVIChangeListFile

change list file name

Requirements for the change list file

The change list file is a text file that contains information about source and destination volumes. The contents of the file must be in the following format, with fields separated by a space and each datastore beginning on a new line:

```
DatastoreType SourceDatastoreName DestinationDatastoreName
SourceVirtualMachineName DestinationVirtualMachineName
SourceVirtualMachineUUID DestinationVirtualMachineUUID
SourceVirtualMachineDirectoryName DestinationVirtualMachineDirectoryName
SourceStorageName DestinationStorageName SourceVolumeName
DestinationVolumeName [SourceDatastoreUUID DestinationDatastoreUUID]
```

In this format, *Datastore Type* is either NFS or VMFS and *Datastore UUID* is not required for an NFS volume.

Note: For NFS datastores residing on clustered Data ONTAP, *SourceStorageName* and *DestinationStorageName* must be the IP addresses of the source and destination NFS data LIFs.

The following example shows the contents of a change list file:

```
NFS ds-nfs1 ds-nfs1-dest snapmgr-05-vm2 snapmgr-54-vm1 4211945a-124a-b7c9-
ae63-cacc07f3f4f8 420f010b-7e5a-e66e-7ed1-7bef6a357cca snapmgr-05-vm1
snapmgr-54-vm1 172.17.233.24 172.17.232.74 snapmgr05_vmw1
snapmgr05_vmw1_mir
```

Finding the UUID for VMFS datastore

To find the UUID for VMFS datastore, follow the steps below:

1. Break the SnapMirror relationship from the storage system.
2. Make the SnapMirror destination volumes online on which the datastore reside.
3. On the secondary SMVI server, mount the LUN in the destination volume as the destination datastore on the secondary ESX server.

When you add the storage from the destination, select the option **Resignature the LUN**.

4. After the destination datastore is added to the secondary ESX server, note the datastore name and UUID values.

5. Replace the destination datastore name and the destination datastore UUID with the new values that you noted in the changelist file.

Defining a backup strategy

Defining a backup strategy before you create your backup jobs helps ensure that you have the backups that you need to successfully restore your databases. Your Service Level Agreement (SLA) and Recovery Point Objective (RPO) largely determine your backup strategy.

For SnapManager best practices, see *NetApp Technical Report 4353: Best Practices Guide for Microsoft SQL Server and SnapManager 7.1 for SQL Server with Clustered Data ONTAP* or *NetApp Technical Report 4232: Best Practice Guide for Microsoft SQL Server and SnapManager 7.0 for SQL Server with Data ONTAP Operating in 7-Mode*.

Different types of SnapManager backup

SnapManager supports two types of backups:

Backup type	Description
Full database backup	Backs up database files and truncated transaction logs. SQL Server truncates transaction logs by removing entries already committed to the database. This is the most common backup type.
Transaction log backup	Backs up truncated transaction logs, copying only transactions committed since the most recent backup. If you schedule transaction log backups alongside full database backups, SnapManager can restore databases to a specific recovery point more quickly. For example, you might schedule full database backups at the start and end of the day and transaction log backups every hour.

For both types of backups, you can choose the *copy-only* option to specify that SQL Server not truncate transaction logs. Use this option when you are backing up your databases with another backup application. Keeping transaction logs intact ensures that any backup application can restore the databases. You typically should not use copy-only in any other circumstance.

When should you back up your databases?

The most critical factor for determining a database backup schedule is the rate of change for the database. You might back up a heavily used database every hour, while you might back up a rarely

used database once a day. Other factors include the importance of the database to your organization, your Service Level Agreement (SLA), and your Recover Point Objective (RPO).

Even for a heavily used database, there is no requirement to run a full backup more than once or twice a day. Regular transaction log backups are usually sufficient to ensure that you have the backups you need.

Tip: The more often you back up your databases, the fewer transaction logs SnapManager has to play forward at restore time, which can result in faster restore operations.

Important: SnapManager can perform one operation at a time. Do not schedule overlapping SnapManager operations.

When should you verify backup copies?

Although SnapManager can verify backup sets immediately after it creates them, doing so can significantly increase the time required to complete the backup job. It is almost always best to schedule verification in a separate job at a later time. For example, if you back up a database at 5:00 p.m. every day, you might schedule verification to occur an hour later, at 6:00 p.m.

For the same reason, it is usually not necessary to run backup set verification every time you perform a backup. Performing verification at regular but less frequent intervals is usually sufficient to ensure the integrity of the backup set. A single verification job can verify multiple backup sets at the same time.

Important: SnapManager can perform one operation at a time. Do not schedule overlapping SnapManager operations.

How many backup jobs do you need?

You can back up your databases using one backup job or several. The number of backup jobs that you choose typically mirrors the number of volumes on which you placed your databases. For example, if you placed a group of small databases on one volume and a large database on another volume, you might create one backup job for the small databases and one backup job for the large database.

Other factors that determine the number of backup jobs that you need include the size of the database, its rate of change, and your Service Level Agreement (SLA).

Which backup naming convention do you want to use?

A backup naming convention adds a string to Snapshot copy names. The string helps you identify when the copies were created. There are two naming conventions:

Naming convention	Description
Unique	Adds a time stamp to all Snapshot copy names. This is the default option. Example: sqlsnap__QATX86W2K8_07-02-2014_10.45.08
Generic	Adds the string “recent” to the name of the most recent Snapshot copy. All other Snapshot copies include a time stamp. Example: sqlsnap__QATX86W2K8__recent

The selected naming convention applies to all backups. You should use the unique naming convention unless you have a script that requires the constant string “recent”.

You can change the naming convention in the **Backup Settings** dialog box.

Which backup management group do you want to assign to the backup job?

You select a backup management group to apply a labeling convention to Snapshot copies. When you back up a database, you can choose from three management groups:

Management group	Description
Standard	Does not include the name of the management group in Snapshot copy names. Example: sqlsnap__QATX86W2K8_07-02-2014_10.45.08
Daily	Adds “Daily” to Snapshot copy names. Example: sqlsnap__QATX86W2K8_07-02-2014_10.47.00__Daily
Weekly	Adds “Weekly” to Snapshot copy names. Example: sqlsnap__QATX86W2K8_07-02-2014_10.49.03__Weekly

For example, if you schedule daily and weekly backups, you should assign the backups to the Daily and Weekly management groups, respectively.

Note: Management groups do not enforce a backup schedule.

How long do you want to retain backup copies on the source storage system and the SnapMirror destination?

You can choose either the number of days you want to retain backup copies, or specify the number of backup copies you want to retain, up to 255. For example, your organization might require that you retain 10 days worth of backup copies.

If you set up SnapMirror replication, the retention policy is mirrored on the destination volume.

Note: For long-term retention of backup copies, you should use SnapVault.

How long do you want to retain transaction log backups on the source storage system?

SnapManager needs transaction log backups to perform *up-to-the-minute restore operations*, which restore your database to a time between two full backups. For example, if SnapManager took a full backup at 8:00 a.m. and another full backup at 5:00 p.m, it could use the latest transaction log backup to restore the database to any time between 8:00 a.m. and 5:00 p.m. If transaction logs are not available, SnapManager can perform *point-in-time restore operations* only, which restore a database to the time that SnapManager completed a full backup.

Typically, you require up-to-the-minute restores for only a day or two, which means you would retain transaction log backups for one or two days.

Do you want to verify backup copies using the source volume or a destination volume?

If you use SnapMirror or SnapVault, you can verify backup copies using the Snapshot copy on the SnapMirror or SnapVault destination volume, rather than the Snapshot copy on the primary storage system. Verification using a destination volume reduces load on the primary storage system.

If you need to create backups using another tool, what backup type should you use?

If you need to create backups using another backup tool, create copy or differential backups only with that tool. Normal (full) and incremental backups truncate transaction logs, effectively disabling SnapManager up-to-the-minute restores.

Backing up your databases for the first time

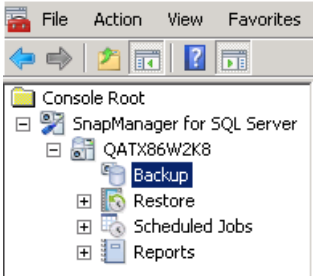
After you migrate your databases to NetApp storage, you should back them up immediately. You can schedule recurring backups after the initial backup and verification.

About this task

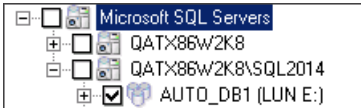
These steps show you how to quickly back up your databases using the Backup and Verify option. You can use the Backup wizard if you prefer.

Steps

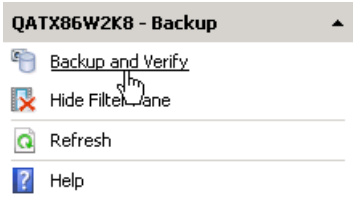
1. In the **Console Root** tree, expand the server on which the databases reside and click **Backup**.



2. In the **Backup** pane, select the databases that you want to backup.



3. In the **Actions** pane, click **Backup and Verify**.



4. In the **Backup and Verification** dialog box, keep **Full database backup** selected and define the properties for the backup job:

For this field...	Do this...
Copy-only backup	If you back up your databases using another backup application, select this field.
Run transaction log backup after full database backup	Keep this field selected.
Backup management group	Select a management group.
Delete full backups	Specify a retention policy for backup copies on the source storage system by defining the number of backup copies to retain or the number of days to retain backup copies.
Up-to-minute Restore Options	Click this field and then specify a retention policy for transaction logs.
Verify databases after backup	Clear this field because it is best to verify databases in a separate operation.

For this field...	Do this...
SnapMirror options	If you set up a SnapMirror destination volume, select the option to replicate the backup copy to the destination volume.
Backup archiving options	If you set up a SnapVault destination volume, select the option to archive the backup copy to the destination volume.
Run Command	If you want to run a command before or after the backup operation, click the **...** button and specify details for the command: where to run the command, the path to the program or script, the SnapManager variables to execute, and the command arguments.
Federated Backup	If you want to back up databases from different instances or different servers, click this field and then add databases to a federated group.
Availability Group Backup	If you have an Availability Group and you want to take backups on all replicas, the primary replica, the secondary replica, or preferred replicas, click this field and specify the replicas.

5. Click **Backup Now**.
6. In the Backup Status dialog box, click **Start Now**.

   You can view details of the operation in the Backup Task List and Backup Report tabs.

Verifying the initial backup set

You should verify an initial backup set to confirm the integrity of the databases.

Steps

1. In the **Backup** pane, select the databases that you want to include in the backup verification schedule.
2. In the **Actions** pane, click **Backup and Verify**.
3. In the **Backup and Verification** dialog box, select **Verify most recent unverified snapshot backups only** and then define the properties for the backup verification:

For this field...	Do this...
Number of snapshot backups to verify	Keep the default. You should have only one backup set at this point.
Backup management group	Select a management group.
SnapMirror options	If you replicated the backup set to a SnapMirror destination volume and you want to verify the backup set on the destination storage system to reduce load on the primary storage system, click **Verify on available SnapMirror destination volumes**.

For this field...	Do this...
Backup archiving options	If you archived the backup set to a SnapVault destination volume and you want to verify the backup set on the destination storage system to reduce load on the primary storage system, click **Verify archive backup on secondary storage**.
Run Command	If you want to run a command before or after the operation, click the **...** button and specify details for the command: where to run the command, the path to the program or script, the SnapManager variables to execute, and the command arguments.
Availability Group Backup	If you are scheduling the verification from the primary server, you can use this option to perform verification on databases whose backup was taken on a secondary replica copy.

4. Click **Verify Now**.
5. In the Backup Status dialog box, click **Start Now**.

   You can view details of the operation in the Backup Task List and Backup Report tabs.

Scheduling recurring backups

You can schedule recurring backup jobs using the SQL Server Agent or Windows Scheduled Tasks.

About this task

In a Windows Failover cluster, it is a best practice to schedule jobs using the SQL Server Agent.

Steps

1. In the **Backup** pane, select the databases that you want to include in the backup schedule.
2. In the **Actions** pane, click **Backup and Verify**.
3. In the **Backup and Verification** dialog box, keep **Full database backup** selected and define the properties for the backup schedule, as described in *Backing up your databases for the first time* on page 18.
4. Click **Schedule**.
5. In the **Schedule Job** dialog box, enter a job name, choose a schedule service (SQL Server Agent or Windows Scheduled Tasks), and click **OK**.
6. Create the schedule using the service that you chose:

If you chose...	Do this...
The SQL Server Agent	**a.** Click **Yes**. SQL Server Management Studio opens. **b.** Connect to the SQL instance. **c.** In the Object Explorer pane, expand the instance. **d.** Expand **SQL Server Agent**, expand **Jobs**, right-click the job and then click **Properties**. **e.** Click **Schedules** and then click **New**. **f.** Fill out the New Job Schedule dialog box and click **OK**.
Windows Scheduled Tasks	**a.** Click **Schedule**. **b.** Specify the schedule. **c.** Click **OK**. **d.** Click **Yes** to save the job.

After you finish

You can view details about the backup job in the SnapManager Scheduled Jobs pane.

Scheduling recurring transaction log backups

You should schedule transaction log backups alongside full database backups at a frequency that allows you to meet your Recovery Point Objective (RPO).

About this task

In a Windows Failover cluster, it is a best practice to schedule jobs using the SQL Server Agent.

Steps

1. In the **Backup** pane, select the databases that you want to include in the backup schedule.
2. In the **Actions** pane, click **Backup and Verify**.
3. In the **Backup and Verification** dialog box, select **Transaction log backup** and then define the properties for the transaction log backup schedule:

For this field...	Do this...
Copy-only log backup	If you back up your databases using another backup application, select this field.
Verify log backup upon completion	Clear this field because it is best to verify backups in a separate operation.
Delete log backups	Specify a retention policy for backup copies on the source storage system by defining the number of backup copies to retain or the number of days to retain backup copies.
Update SnapMirror after operation	If you set up a SnapMirror destination volume, select this field to replicate the backup copy to the destination volume.
Federated Backup	If you want to back up transaction logs from different instances or different servers, click this field and then add databases to a federated group.
Marking Transaction Options	If you specified a federated backup group, choose the default mark name and description or modify them. You use these marks to restore databases to the same marked transaction across multiple databases for a synchronous restoration.
Run Command	If you want to run a command before or after the backup operation, click the **...** button and specify details for the command: where to run the command, the path to the program or script, the SnapManager variables to execute, and the command arguments.
Availability Group Backup	If you have an Availability Group and you want to take backups on all replicas, the primary replica, the secondary replica, or preferred replicas, click this field and specify the replicas.

4. Click **Schedule**.
5. In the **Schedule Job** dialog box, enter a job name, choose a schedule service (SQL Server Agent or Windows Scheduled Tasks), and click **OK**.
6. Create the schedule using the service that you chose:

If you chose...	Do this...
The SQL Server Agent	**a.** Click **Yes**. SQL Server Management Studio opens. **b.** Connect to the SQL instance. **c.** In the Object Explorer pane, expand the instance. **d.** Expand **SQL Server Agent**, expand **Jobs**, right-click the job and then click **Properties**. **e.** Click **Schedules** and then click **New**. **f.** Fill out the New Job Schedule dialog box and click **OK**.
Windows Scheduled Tasks	**a.** Click **Schedule**. **b.** Specify the schedule. **c.** Click **OK**. **d.** Click **Yes** to save the job.

Scheduling recurring backup set verifications

You can schedule recurring backup set-verification jobs using the SQL Server Agent or Windows Scheduled Tasks.

About this task

In a Windows Failover cluster, it is a best practice to schedule jobs using the SQL Server Agent.

Steps

1. In the **Backup** pane, select the databases that you want to include in the backup verification schedule.
2. In the **Actions** pane, click **Backup and Verify**.
3. In the **Backup and Verification** dialog box, select **Verify most recent unverified snapshot backups only** and define the properties for the backup verification schedule as described in *Verifying the initial backup set* on page 20.

   You might need to modify the **Number of snapshot backups to verify** field, depending on the number of backups SnapManager will take between scheduled verifications.
4. Click **Schedule**.

5. In the **Schedule Job** dialog box, enter a job name, choose a schedule service (SQL Server Agent or Windows Scheduled Tasks), and click **OK**.
6. Create the schedule using the service that you chose:

If you chose...	Do this...
The SQL Server Agent	**a.** Click **Yes**. SQL Server Management Studio opens. **b.** Connect to the SQL instance. **c.** In the Object Explorer pane, expand the instance. **d.** Expand **SQL Server Agent**, expand **Jobs**, right-click the job and then click **Properties**. **e.** Click **Schedules** and then click **New**. **f.** Fill out the New Job Schedule dialog box and click **OK**.
Windows Scheduled Tasks	**a.** Click **Schedule**. **b.** Specify the schedule. **c.** Click **OK**. **d.** Click **Yes** to save the job.

After you finish

You can view details about the verification job in the SnapManager Scheduled Jobs pane.

Managing backup retention

Your backup retention strategy needs to balance storage efficiency with restore needs. You can specify that SnapManager automatically delete older backups or transaction logs, or you can delete these items explicitly.

Note: You should not use SnapDrive or storage system administration tools to delete Snapshot copies created by SnapManager. Doing so leaves behind unwanted data that cannot be removed.

Maximum number of Snapshot copies per volume

The Data ONTAP software used with SnapManager supports a maximum of 255 Snapshot copies per volume, including copies not created by SnapManager. Because each SnapManager backup operation

creates Snapshot copies, a SnapManager backup operation fails if the volume that contains the database LUN exceeds the 255 Snapshot copy capacity.

Note: The total number of Snapshot copies on a volume might exceed the number of retained backups. For example, if a single volume contains both the SnapInfo directory and the databases, each backup operation generates two Snapshot copies on the volume.

Automatically deleting backups

When you start or schedule a full database backup, you can also specify the number of backup sets of that database to be retained for that backup management group. After the backup is complete, SnapManager will automatically delete the oldest backup sets for that database in the specified backup management group, retaining only the number of backups you want to preserve.

SnapManager retention policy does not apply expiration days for individual backups, but instead manages how many backups are retained at any given time.

Note: If a database is deleted, SnapManager for SQL Server stops actively managing the backups. The backups remain until manually deleted.

Cases in which more backups are preserved

SnapManager does not count backups that failed verification when counting the number of stored backups. More backups might be preserved than you specify in the Delete Oldest Backups In Excess Of dialog box.

For example, suppose you are backing up databases A and B, which contain the following backup sets:

SnapManager backup set	Description
Database A	
`sqlsnap__orbit3_10-23-2013_16.21.07`	Old backup- good
`sqlsnap__orbit3__recent`	Recent backup- good
Database B	
`sqlsnap__orbit3_10-23-2013_16.21.07`	Old backup- good
`sqlsnap__orbit3__recent`	Recent backup- inconsistent

Also suppose you have set the Delete Oldest Backups in Excess Of dialog box to `1` to preserve only one of each backup set, the most recent one.

In order to preserve one good backup for Database B, SnapManager does not delete the Snapshot copy `sqlsnap__orbit3_10-23-2013_16.21.07`. Therefore, two backups for Database B remain instead of one.

Option to retain up-to-the-minute restore ability

If you delete backups that are not the oldest backups in your backup list (as can happen when you are deleting backups of a particular backup management group), the corresponding transaction logs are also deleted. That means the older remaining backups are no longer available for an up-to-the-minute restore because the transaction logs are no longer contiguous from the time when the older backup was taken to the present time.

SnapManager for Microsoft SQL Server enables you to preserve the logs in this case, thereby retaining the ability to use the older backups in an up-to-the-minute restore.

Explicitly deleting backups

You can automatically delete older backup sets by specifying the Delete full backups in excess of option and the Delete full backups older than option in the SnapManager backup tools. You should use this method for managing the number of backup sets stored. You can also explicitly select the backup sets that you want to delete.

Steps

1. In the SnapManager **Console Root**, select a server.
2. In the Actions pane, select **Delete Backup**.
3. Choose a delete operation:

If you want to delete...	Then...
Backup sets created from a backup operation	Keep **Delete backups** selected.
Snapshot copies created during a previous restore operation	Select **Delete Snapshot of LUNs created during restore**.

4. Specify the backups to delete based on the delete operation that you chose:

If you chose...	Then...
Delete backups	**a.** Select one or more databases. **b.** Specify the backup component that you want to delete: • Backup data sets • Log Snapshot copies only • SnapInfo Snapshot copies only **c.** Specify the management group of the backup copies that you want to delete. **d.** If you want to delete backups containing any one or more of the selected databases, click **Advanced** and enable the option. **e.** Specify the backups to delete: • The number of backup copies that you want to delete • Backup copies older than a specific number of days • All backups in the specified management group
Delete Snapshot of LUNs created during restore	Specify the backups to delete: • The number of backup copies that you want to delete • Backup copies older than a specific number of days • All backups in the specified management group

5. You can delete the backup sets immediately or you can preview the operation:

If you want to...	Then...
View the list of backup sets that SnapManager will delete	Click **Delete Preview**. The Delete backups dialog box appears. After a moment, the dialog box displays a count and list of the backups identified for deletion. To view a report, click **Show Report**. Based on the list displayed in the Delete backups dialog box, you can either cancel the delete operation or proceed with it.
Delete the backup sets	Click **Delete**.

Managing Availability Group transaction log backups

Before SnapManager can manage transaction log backups, you must set a number of options.

Before you begin

The SnapManager connection to the node in the Availability Group is already configured. The node can be primary or secondary.

Note: SnapManager must be configured to connect to a node in the SQL Server cluster, not to the Availability Group endpoint name itself.

Steps

1. Select **Backup** under the server.
2. Select **Backup Settings** in the **Actions** window.
3. Click the **Transaction Log Backup** tab.
4. Under **Repository Log backup Options**, specify options for copying transaction log backups to shares:
   a. Click either **Apply to all Databases** or **Apply to only Availability Group Databases**.
   b. Optional: Click **Delete Share Log backups** and choose how the backups are selected for deletion.

   Note: Copy transaction log backup to share is enabled by default for Availability Group databases.

Changing the backup management group of an existing backup set

You can use the Change Backup Management Group dialog box to change the backup management group to which the selected backup set belongs.

About this task

You cannot change the backup management group of the most recent backup sets that were created using the generic naming convention.

Steps

1. In the SnapManager console root, click **Restore**.

2. In the **Restore** panel, locate the backup set whose management group you want to change:
   - Database Snapshots (Standard group)
     - sqlsnap__sqlserverhostname__*date_time*
     - sqlsnap__sqlserverhostname__ recent
   - Database Snapshots (Daily or Weekly group)
     - sqlsnap__sqlserverhostname__*date_time__backupmgmtgroup*
     - sqlsnap__sqlserverhostname__*backupmgmtgroup__*recent
3. Right-click the name of the backup set to open a context menu and then select **Change Management Group**.
4. Review the backups listed in the **Backups sharing this Snapshot list**.

   The backup management group for all these backups is changed if you complete this operation because they share a common backup set.
5. In the **New Management Group** list, select the backup management group you want to change to.

   When you change a backup's backup management group, you also change that backup's name because the name includes the backup management group.
6. Click **OK**.

   The backup management group for this backup and all backups listed in the All Backups Sharing This Snapshots list is changed.

   The report for the backup management group change is in the `Miscellaneous report` directory.

What to do if a SnapManager backup operation fails

If a SnapManager backup fails, check the backup report for details about what SnapManager was trying to do when the failure occurred. SnapManager reports are described in "Managing SnapManager operational report." You can also review common backup failures to correct the issue.

Problem deleting backups due to busy Snapshot copy error

If you delete a backup copy of a LUN that was already backed up by another Snapshot copy, you get an error message saying that the Snapshot copy is busy and cannot be deleted. In this case, you must delete the most recent backup copy before the older backup can be deleted.

To avoid this situation, make sure that you do not make backup copies of LUNs that are already backed up by Snapshot copies (for example, during a verification or while archiving from a LUN backed by a Snapshot copy).

SnapInfo directory being accessed

Because a SnapManager backup might include renaming a SnapInfo subdirectory and Windows does not allow a directory name to be changed while it is being accessed, accessing the SnapInfo directory with a tool such as Windows Explorer could cause the backup to fail. Make sure that you do not hold any exclusive access to the SnapInfo directory on the SQL Server host system while a backup is progress.

SnapInfo directory out of space

Expand the LUN that contains the SnapInfo directory or, in the case of SMB shares, expand the volume.

> **Note:** When you expand a LUN, ensure that enough space remains in the volume for backup set creation, so that SnapManager can continue to function correctly.

Data does not match

This error occurs if you made changes to your SQL Server database configuration after SnapManager was started. Any of the following actions refresh your SnapManager view:

- Press F5 on your keyboard.
- From the SnapManager console root, click **Action** > **Refresh**.
- Restart SnapManager.

Backup set already exists

Either of the following circumstances might cause this error to occur:

- The system clock on the host running SnapManager might not be synchronized with the clock on the storage system. These two clocks must be synchronized for SnapDrive to function correctly. For more information, see the *Data ONTAP Software Setup Guide* for your version of Data ONTAP.
- If a SnapMirror replication job is running when you attempt to begin a SnapManager backup, the backup can fail. You can avoid this problem by making sure that SnapMirror replications have enough time to finish before you begin another SnapManager backup.

SnapManager server initialization failed

To correct this issue, close the SnapManager for Microsoft SQL Server GUI, restart the SnapManager service from within the services.msc console, and then reopen the SnapManager GUI.

Error processing backup job

SnapManager reports an error when scheduling a backup job using the SQL Server Agent:

```
An Error occurred while processing SQL Agent Job creation. The specified
@server_name (name)does not exist.
```

The SQL Server instance may be hidden or the SQL Browser service may be down.

To unhide the instance or start the SQL Browser service, close the SnapManager for Microsoft SQL Server GUI, restart the SnapManager service from within the services.msc console, and then reopen the SnapManager GUI.

VMDK backup fails when you specify a physical server as the verification server

The backup copy created on the VMDK cannot be verified on a physical server. To resolve this error, select a verification server running on a virtual machine.

Database is not in valid configuration

A SnapManager backup operation might fail with the following message in the backup report:

```
WARNING: Database DatabaseName of ServerName is not in valid configuration,
and will not be included in this backup.
```

This message can appear if any of the following is true:

- The database is not in the online state.
  For example, if autoshrink is enabled on the database, the database is not online during the shrink operation. For information about autoshrink, see *Microsoft article 315512*.
- The database configuration is not supported by SnapManager.
  To verify that your configuration is supported, see the *SnapManager for Microsoft SQL Server Installation and Setup Guide*.

[DBMSLPCN] ConnectionRead (WrapperRead())

If the SnapManager host system is running SQL Server 2005, a SnapManager backup operation might fail with the following message in the backup report:

```
[Microsoft][ODBC SQL Server Driver][DBMSLPCN]ConnectionRead
(WrapperRead()).
```

To avoid this problem, install MDAC 2.8 SP1 on the Windows host.

If Readable Secondary set is to “No”

If any of a replica's secondaries have Readable Secondary set to “No” and that replica is still part of the Availability Group backup, SnapManager displays an error similar to this:

```
The target database, 'test1', is participating in an availability group and
is currently not accessible for queries. Either data movement is suspended
or the availability replica is not enabled for read access. To allow read-
only access to this and other databases in the availability group, enable
```

`read access to one or more secondary availability replicas in the group. For more information, see the ALTER AVAILABILITY GROUP statement in SQL Server Books Online.`

To correct this, change Readable Secondary to “Yes” for all of the replicas participating in backups.

If the Availability Group was “Cloned as Replica” and the user takes an AG backup across all replicas

SnapManager can clone the AG to the selected replica with the **Readable Secondary** backup option set to “No”, but afterwards, if the user tries to select the same AG and tries to take a backup across all replicas, the backup of the cloned AG is skipped because the databases are mounted as read-only.

The problem occurs because the requested database has not been configured for backup. Instead, the database is marked as “Database is on a LUN backed by snapshot.” The message is similar to `[02:14:38.145] [SQL2012HA3] Database requested has not been configured for backup. This database is marked as: Database is on a LUN backed by snapshot. Server: SQL2012HA2 Database:1ag-db1. This Database will be skipped.`

Availability Group database status

SnapManager skips the backup of an Availability Group if any of the replica databases are not in Synchronizing or Synchronized states.

Lack of available drive letters causes DBCC CHECKDB to fail

When you run database verification against backup sets that contain multiple database files stored on separate LUNs or VMDKs, SnapManager mounts all those LUNs or VMDKs at the same time. Consequently, the SQL Server that is used as the verification server must have enough drive letters available so that SnapManager can mount each of the LUNs or VMDKs simultaneously.

If the verification server runs out of available drive letters while attempting to run `DBCC CHECKDB` for a SnapManager operation, the SnapManager operation fails with the following message in the report log:

```
[SnapDrive Error]: There are no remaining drive letters available on the
system.  Please delete or disconnect a drive and retry.
```

Note: Available drive letters are not required if the databases reside on SMB shares.

Restoring databases

You can use SnapManager to restore an SQL Server database without taking the server offline. You can restore the database to a number of different destinations in addition to its original location.

How SnapManager a restore operation works

It is important to understand how a SnapManager restore operation works before performing one.

SnapManager restores the databases that you select to the active file system. The restore method used by SnapManager depends on (1) the method that was used to create the backup set and (2) the specific subset of databases you choose to restore from the backup set.

SnapManager uses the *stream-based* restore method if you are restoring from a stream-based backup set. With this method, each of the databases is restored individually. Depending on the composition of the backup set, a stream-based restore can require additional time and free space on the storage system as compared to an online Snapshot copy restore.

SnapManager uses LUN, SMB share, and VMDK cloning if you are restoring from a backup set that contains multiple databases that reside on the same LUN, SMB share, or VMDK.

SnapManager uses the *copy-based* restore method if any of the following conditions are true:

- The backup set contains only a subset of the databases that reside on the same LUN, SMB share, or VMDK (not recommended).
- You select only a subset of the databases contained in the backup set.
- A new database was added to the same LUN, SMB share, or VMDK after the backup was created.

In a volume-wide backup, all the databases that reside on a single volume are backed up concurrently using Snapshot copies. Because the maximum number of databases supported per storage system volume is 35, the total number of Snapshot copies created equals the number of databases divided by 35.

If the database has transaction log backups, SnapManager Restore can apply the transaction log backups (if necessary).

Depending on the database restore option selected, SnapManager Restore performs a *point-in-time* restore or an *up-to-the-minute* restore.

Restore Snapshot copies

Every time you perform a restore operation using SnapManager, SnapManager first creates a Snapshot copy on each storage system volume that contains files for the databases you will be

restoring. That way, in the unlikely event that a catastrophic failure occurs during a restore operation, you have recent Snapshot copies of the LUNs, SMB shares, or VMDKs that can be used to re-create those databases as they existed prior to the start of the failed restore operation.

Each restore Snapshot copy is named using the following naming convention:

`rstsnap__`*SqlServerName_date_time*

The Snapshot copy name contains the name of the SQL Server instance to which the backup was restored (indicated by the variable *SqlServerName*) and the Snapshot copy creation date and time (indicated by the variable *date_time*).

After you verify that a restore was completed successfully and you are satisfied with the results, you can delete the restored Snapshot copy.

SQL Server cluster group state during a restore

SnapManager can restore databases in a Windows cluster without taking the SQL Server cluster group offline.

Cluster failure during a restore operation

If a cluster failure (a cluster group move operation) occurs during the restore operation (for example, if the node that owns the resources goes down), you must reconnect to the SQL Server instance and then restart the restore operation.

Transaction log restore operations

A SnapManager transaction log restore uses the SQL recovery process to play forward transactions from the log backup into the restored database.

Importance of verifying databases to be restored

The database verification process protects you from restoring a backup that contains any physical-level corruption. Physical-level database corruption can occur silently in SQL Server databases. The only way to know whether a particular database backup incurred physical-level corruption is to run database verification on that backup.

Before allowing a restore operation to proceed, SnapManager enables you to check that the selected backup set was verified through the use of DBCC `CHECKDB.

Backup verification status

SnapManager Restore shows you a list of the backups that have been taken. For each backup, the date and time of the backup is displayed, as well as an icon that indicates the backup verification status.

Icon description	Backup verification status
Circled check mark	The databases in this backup have been verified.
Circled question mark	The databases in this backup have not been verified.

If you select a database on which a consistency check has not been run successfully, SnapManager prompts (but does not require) you to run DBCC before performing a restore. Running database consistency checking as part of recovery increases the time the recovery takes.

Types of SnapManager restore operations

You can use SnapManager to perform a number of different types of restore operations.

- Up-to-the-minute restore operations
- Point-in-time restore operations
- Marked transaction restore operations

Up-to-the-minute restore operations

In an up-to-the-minute restore, databases are recovered up to the point of failure. SnapManager accomplishes this by performing the following sequence:

- The last active transaction log is automatically backed up.
- The databases are restored from the full database sets you select.

All the transaction logs that were not committed to the databases, including transaction logs from the *backup sets*, from the time the backup set was created up to the most current time, are played forward and applied to the databases (if selected).

An up-to-the-minute restore requires a *contiguous set of transaction logs*. The up-to-the-minute restore type is selected by default.

Because SnapManager cannot restore transaction logs from *log-shipping* backup files, you might not be able to restore the database using an up-to-the-minute restore. For this reason, you should use SnapManager only to back up your SQL Server database transaction log files.

If you do not need to retain up-to-the-minute restore capability for all backups, you can configure your system's transaction log backup retention through Up-to-minute Restore Options, located in the Backup and Verify window.

Example

You run SnapManager Backup every day at noon, and on Wednesday at 4:00 p.m. you need to restore from a backup. For some reason, the backup set from Wednesday lunch time failed verification, so you decide to restore from the Tuesday lunch time backup. If the After that backup is restored, all the transaction logs are played forward and applied to the restored databases, starting

with those that were not committed when you created Tuesday's backup set and continuing through the latest transaction log written on Wednesday at 4:00 p.m. (if the transaction logs were backed up).

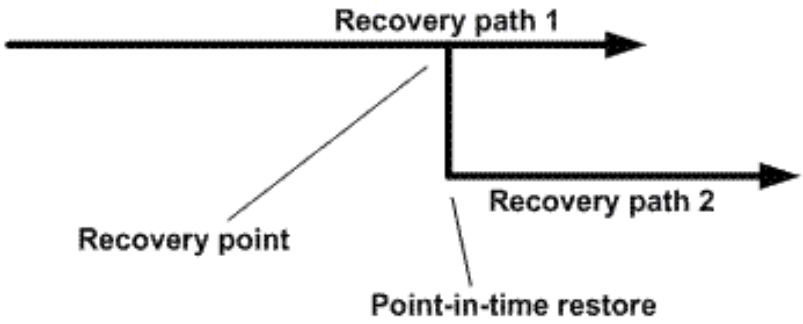
Point-in-time restore operations

In a point-in-time restore, databases are restored only to a point-in-time from the past. A point-in-time restore occurs in two restore scenarios:

- The database is restored to a given time in a backed up transaction log.
- The database is restored and only a subset of backed up transaction logs are applied to it.

Note: When you restore a database to a point in time, it results in a new recovery path.

The following image illustrates the potential problems when a point-in-time restore is performed:



In the image, Recovery path 1 consists of a full backup followed by the number of transaction log backups. The database administrator restores the database to a point in time. New transaction log backups are created after the point-in-time restores which results in Recovery path 2. The new transaction log backups are created without creating a new full backup. Due to data corruption or other problems, if you need to restore the current database, you will not be able to restore it because a new full backup was not created. Also, it is not possible to apply the transaction logs created in Recovery path 2 to the full backup belonging to Recovery path 1.

Note: You must ensure that you always create a full backup after restoring a database to a point in time.

If you apply transaction log backup sets, you can also specify a particular date and time at which you want to stop the application of backed up transactions. To do this, you specify a date and time within the available range and SnapManager will roll back any transactions that were not committed prior to

that point in time. You can use this method to restore databases back to a point in time before a corruption occurred, or to recover from an accidental database, or table deletion.

Example

Suppose you take full database backups once at midnight and a transaction log backup every hour. The database crashes at 9:45 a.m., but you still back up the transaction logs of the failed database. You can choose from among three point-in-time restore scenarios:

- Restore the full database backup taken at midnight and accept the loss of the database changes made afterward.
- Restore the full database backup and apply all the transaction log backups until 9:45 a.m.
- Restore the full database backup and apply transaction log backup sets. Specifying the time you want the transactions to restore from the last set of transaction log backups.

In this case, you would calculate the date and time where a certain error was reported. Any transactions that were not committed prior to the date and time specified in the `Restore` command are rolled back.

Marked transaction restore operations

Restore-to-marked transaction operations enable you to restore a database to a marked transaction. Using the marks created on a federated full backup, you can restore a backup to a marked transaction across multiple databases for a synchronous restoration.

Note: You can use *either* restore to mark *or* restore to point-in-time. They do not work simultaneously.

These transaction marks are recorded in the transaction log and included in the logs of the affected database.

Sources and destinations for a SnapManager restore

You can restore an SQL Server database from a backup set, from an offline archive of a backup set, or from a database residing on multiple LUNs, SMB shares, or VMDKs. You can restore the database to a number of different destinations in addition to its original location.

Sources for a restore operation

You can restore databases from various types of sources:

Source	Description
SnapManager backup set	You can restore databases from SnapManager backup sets created for the same SQL Server instance or created for a different server instance. The LUNs, SMB shares, or VMDKs containing the selected SQL Server's databases are restored from the backup.

Source	Description
Unmanaged media	You can also use SnapManager Restore to restore databases from offline archives (unmanaged media) of backup sets.
Database residing on multiple LUNs, SMB shares, or VMDKs	You can restore databases that reside on multiple LUNs, SMB shares, or VMDKs. The restore operation takes some time to complete because SnapManager takes one database at a time, serially, for the complete restore operation.

Destinations for a restore operation

You can restore databases to various types of destinations:

Destination	Description
The original location	By default, SnapManager restores to a database to the same location on the same SQL Server instance.
A different location	You can restore a database to a different location on the same SQL Server instance.
A different location as a database clone	You can use SnapManager Restore to restore an online database as a new database on the same SQL Server instance. However, you *cannot* restore an online database as a new database on a different SQL Server instance. You must clone the database.
Original or different location using different database names	You can restore to a different server instance on the same or different server using different database names.
A different location, but only temporarily mounted	The database is mounted at a temporary alternate location, but the transaction logs are not applied. Because the data is not current, you should use this function to view only the layout of the data.

Transaction log backups from SQL Server Management Studio

SnapManager cannot automatically restore the transaction log backups taken using SQL Server Management Studio. You need to modify the file extension before SnapManager can use the backups.

Transaction log backups created by SQL Server Management Studio have a default extension of `.TRN`. If you want to restore the transaction log backups from SnapManager, you must rename these `.TRN` files to `.TRB` and place them in the respective SnapInfo directory. After you place the transaction log backups in the correct SnapInfo directory, you can use SnapManager to restore from the logs.

Post-restore database recovery states

Post-restore database recovery states describe the condition of the database after a restore operation and any further restore actions that you can take.

The following table describes the post-restore database recovery states:

Database state	Description
Operational	All of the following conditions apply: • No more transaction logs can be restored. • The database is ready to use. This database state is selected by default.
Non-Operational	More transaction logs can be restored.
Read-Only	All the following conditions apply: • More transaction logs can be restored. • The undo file is enabled. If more transaction logs are restored, any changes can be rolled back if the restore operation for the transaction log is unsuccessful. **Note:** If you restore a database to a temporary, alternate location using a writable Snapshot copy with this option enabled, the Detach Database and Dismount Snapshot LUN(s) function is unavailable for this database.

Requirements for restoring a database

Before you restore a database from a SnapManager backup set, you must ensure that several requirements are met.

- The SQL Server must be online and running before you can restore a database. This applies to both user database restore operations and system database restore operations.
- The target databases must be detached or in a suspect state.
- If you want to restore online databases, the online restore option must be enabled in the Restore Settings dialog box.
- If you restore multiple databases to the same SQL Server instance, you must not assign the same target database name for multiple databases.

- All Windows Explorer instances must be closed on the SQL Server computer running SnapManager.
- SnapManager operations that are scheduled to run against the SQL Server data you are restoring must be disabled, including any jobs scheduled on remote management or remote verification servers.
- If the restore is from a database on a different SQL server, the source storage must have been made available to the current SQL server.
- If system databases are not functional, you must first rebuild the system databases using an SQL Server utility.

Finding backup sets

You can more easily find backup sets by using the Find Backups wizard. The wizard finds backups created on a specific SQL Server or backups residing on unmanaged media.

Steps

1. In the **Console Root** tree, expand the server on which the databases reside and click **Restore**.
2. In the **Actions** pane, click **Find Backups**.
3. On the **Start** page, click **Next**.
4. On the **SQL Server** page, choose whether you want to find a backup set created on this SQL Server, created on a different SQL Server, or a backup set residing on unmanaged media, and then click **Next**.
5. If you chose to restore from a different SQL Server or unmanaged media, enter the SnapInfo directory path for the backup set and click **Next**.
6. In the **Find Finish** page, click **Finish**.

   The Restore pane refreshes and displays the backup copies that match your search criteria.

Restoring a database from a local backup set

You can restore an SQL Server database from a local backup set by using the SnapManager Restore dialog box.

Steps

1. In the **Console Root** tree, expand the server on which the databases reside and click **Restore**.
2. In the **Restore** pane, double-click the backup set from which you want to restore.

3. In the **Actions** pane, click **Restore**.

   The SnapManager for SQL Server-Restore window appears.

   Note: When SnapManager takes transaction logs on multiple nodes of a Windows Server Failover Cluster, the Restore window displays multiple transaction logs that have the same timestamp, but different indexes. For example: "04-25-2014_05.57.08 (1)" and "04-25-2014_05.57.08 (2)"

4. To restore as a database with a different name than the original database, follow these steps:
   a. Click the tab marked ... next to **Restore as Database**.
   b. The Individual Database Restore As... dialog box appears.
   c. In the Restore as Database dialog box, enter the database name to which you want the backup restored. This database name must not already exist on the SQL Server instance to which you will be restoring the database.
   d. Click **OK** to apply your change and close the dialog box.
5. Click **OK** to apply your change and close the dialog box.
6. Select or enter the server name that you want the database to be restored to.
7. Choose the connection by selecting the **Use Windows Authentication** or **Use SQL Server Authentication** radio button.
8. Click **OK** to apply your change and close the dialog box.
9. Specify the restore type:

If...	Then...
You want to restore to a point-in-time backup	**a.** Click the tab marked "..." next to Point-in-Time Restore. The Point-in-Time dialog box opens. **b.** In the Point-in-Time Restore dialog box, specify the date and time after which transaction logs are not applied to the restored database. **c.** Click **OK**. **Note:** A point-in-time restore operation halts the restoration of transaction log entries that were recorded after the specified date and time.

If...	Then...
You want to restore to a marked transaction	**a.** Click the tab marked "..." next to Marked Transaction. The Marked Transaction dialog box opens. **b.** In the Marked Transaction dialog box, select which marked transaction at which to stop the restore operation. **c.** Click **OK**.

10. To run a command or script prior to performing the restore operation or after the restore operation finishes, select the **Run Command Settings** option.

11. Do the following to restore the database to a different location:

a. Click the ... tab next to **Restore to Other Location**.

b. Edit the location by selecting and modifying the Restore To field for each row or select the ... tab and browse for the location.
Note the following requirements for the location:

- If you restore a database to a different path and that path is an SMB share, the SMB share must be accessible from SnapDrive.
- If you chose to restore from unmanaged media, enter the location of the mounted disk where the database files are available.
- If you chose to restore from unmanaged media, ensure that you place the database files in the restoring location.
- You cannot spread a database's files across SAN and NAS environments.

  Note: If the alternate location does not have enough space, the restore will fail. If this happens, delete the partially copied database files.

12. Click **Restore** to start the restore operation.

SnapManager begins to restore your databases from the backup you selected. SnapManager Restore completes each task and checks it off the list shown in the Restore Task List view.

You can switch back and forth between the task check-off list and the progress report using the **Switch** buttons on either window.

If the restore operation is successful, the Task window shows the checkoff list with the tasks completed, and a dialog box reports that the restore operation was successful.

13. After all the restore tasks are finished, click **OK**.

Your restore operation is complete and your SQL Server computer comes back online.

14. After the restore is complete, you should perform a full backup and verification to confirm that your restored database is free of physical-level corruption. This step is especially important if you restored a database to a different path that is shared by existing databases.

Restoring a database from a backup set created on a different server

You can restore an SQL Server database from a backup set created on a different server by using the SnapManager Restore wizard.

Steps

1. If you have LUNs and the source LUNs for the failed databases are still online and mapped on the primary storage, do the following:
   a. Note the LUN drive letter assignments.
   b. Unmap the LUNs using System Manager or the storage system console.
   c. In Microsoft Failover cluster configurations, remove any cluster resource dependencies you might have configured on these LUNs.
2. If you have LUNs, reconnect the restored LUN objects with the SnapDrive MMC interface using the original drive letters.

   Consult the SnapDrive documentation for details. Ensure that the LUNs are accessible on the hosting SQL Server.
3. Use SQL Server Management Studio to attach the database located on the LUNs and SMB shares.

   If you cannot attach the database, you can reduce the loss of data by ensuring that the last active transaction log of the database is automatically backed up by SnapManager Restore:

   - See Microsoft KB article 253817, “HOW TO: Back up the Last Transaction Log When the Master and the Database Files Are Damaged.”
     This article describes how you can back up the currently active transaction log even if the SQL Server database file is damaged, provided that the transaction log file is still accessible.
   - Use this same Microsoft KB article as a general guide for gaining access to the last active transaction log of the database.
     While referring to the steps in that article, observe the following key points:
     - When you create a similar database that contains the same number of data and transaction log files as the original database, you are creating the database you will be restoring using SnapManager.

- Instead of using the SQL Server `Backup Log` command to back up the transaction log (as described in the Microsoft article), proceed to the next step.

4. Start the SnapManager for Microsoft SQL Server application.
5. Make sure that all other Windows Explorer windows are closed on the SQL Server computer running SnapManager.
6. Disable any SnapManager operations that are scheduled to run against the SQL Server data you are restoring including any jobs scheduled on remote management or remote verification servers.
7. In the SnapManager **Console Root** tree, expand the server on which the databases reside and click **Restore**.
8. In the **Actions** pane, click **Restore Wizard**.
9. On the **Welcome** page, click **Next**.
10. On the **SQL Server** page, select **Restore backup created on a different server** and click **Next**.
11. On the **Alternate SQL Server** page, specify details about the alternate SQL Server:
    a. Select the SQL Server whose backup sets you want to use to restore databases to this SQL Server.
    b. In the **SnapInfo Directory Path** dialog box, enter or browse to the name of the SnapInfo directory for those backup sets.
    c. Leave the **Use this server's SnapInfo directory** check box cleared.
    d. Click **Next**.
12. In the **Backup Set** page, double-click to select the backup under the database you want to restore. Click **Next**.
13. Use the**Point-in-time** option in the **Transaction Logs** screen to perform an up-to-the-minute restore operation or a point-in-time restore operation:
    - For an up-to-the-minute restore, backup the most recent transactions and select them for restore by selecting **Most recent backup selected**.
    - For a point-in-time restore operation, select the backup set, a combination of transaction log backups to be restored, or both, and select **Committed transactions at the specified time**.
14. Follow the instructions in the **Restore** wizard as you proceed.
15. To restore the database to a different location, do the following:
    a. Click the ... tab next to Restore to Other Location.
    b. Edit the location by selecting and modifying the **Restore To** field for each row or select the ... tab and browse for the location.

Note the following requirements for the location:

- If you restore a database to a different path and that path is an SMB share, the SMB share must be accessible from SnapDrive.
- You cannot spread a database's files across SAN and NAS environments.

Note: If the alternate location does not have enough space, the restore fails. If this happens, delete the partially copied database files.

16. After you verify that all the settings in the page are correct, click **Finish**.

    The Restore wizard closes and the Restore Status dialog box appears and displays the Restore Task List, which is used to show the progress of the restore operation after you start it.

17. Click **Start Now** to start the restore operation.

    SnapManager begins to restore your databases from the backup you selected. SnapManager Restore completes each task and checks it off on the list shown in the Restore Task List view.

    You can switch back and forth between the task checklist and the progress report by using the **Switch** buttons on either window.

    If the restore is successful, the Task window shows the checklist with the tasks completed, and a dialog box reports that the restore was successful.

    Note: If Notification is enabled, email is sent to the specified address. All events are posted to the Windows event log, even if notification is not enabled.

18. After the restore is complete, click **OK** to close the dialog box.

    Your restore is now complete and your SQL Server computer comes back online.

19. After the restore operation is complete, you should perform a full backup and verification to verify that your restored database is free of physical-level corruption. This step is especially important if you restored a database to a different path that is shared by existing databases.

Restoring replicated, publisher, and subscriber databases

You can restore replicated, publisher, and subscriber databases.

Before you begin

- Before restoring replicated databases, you must have stopped the running SQL Agent.
- You must have taken publisher and subscriber database offline.

About this task

Restore the following databases strictly in the given order:

1. Distribution database
2. Publisher database
3. Subscriber database

Note: If you do not restore the distribution database first, the replication settings are not maintained and you will have to restart the replication.

Steps

1. In the **Action** pane, click **Restore Setting**.
2. Select the check box options **Retain SQL database replication settings** and **Restore database even if existing databases are online**.
3. If you have multiple replication sets, restore the most recent distribution database to maintain the replication settings for all the other replicated databases.
4. Reinitialize the restored publisher or subscriber databases because they are out of sync with the latest distribution database.

Reseeding a database on an AlwaysOn cluster

If a replica database gets out of sync with the primary database in an Availability Group, you can reseed the replica database from an existing SnapManager backup. Using the reseed operation, any unhealthy secondary database can be recovered and brought back in synch with its primary database. This is quicker than copying the primary database to the replica and requires less network bandwidth.

Before you begin

A common SnapManager share for all the replica nodes and the share retention settings configured using **Actions** > **Backup Settings** > **Transaction log backup** > **Repository Log backup Options**.

About this task

Note that you cannot use a stream-based backup to reseed a database, and so stream-based backups are not displayed by the Reseed Wizard.

Primary databases and non-Availability Group databases are skipped during the reseed operation.

Steps

1. From the management console, select the standalone server hosting the Availability group databases you want to use to reseed. For example, **Console Root** > **SnapManager for SQL Server** > **AlwaysOn Cluster 1**.
2. Select **Reseed Wizard** in the **Actions** window.

The Reseed Wizard opens.

3. Follow the steps in **Reseed Wizard** to select the database, logs, and optionally a custom command, before verifying the reseed settings and starting the reseed operation.

Recovering databases using archived backup sets

You can restore SQL Server databases from archived backup sets.

Before you begin

- The storage system must be up and running and ready for data to be restored.
- The backup media must be available and ready to be used for restore.
- The database must be detached, using SQL Server Enterprise Manager or SQL Server Management Studio.
- The LUNs must be disconnected from the Windows host machines.
- For LUNs and SMB shares, you must know the original drive letters used by LUNs and the original names of the shares.

Steps

1. Run the **Restore** wizard and select the option **Restore from unmanaged media**.
2. Recover the archived LUNs and SMB shares containing the full backup dataset to the active file system of the storage system.
3. Reconnect the LUNs to the original drive letters and give the hosts access to the shares.

Recovering databases using SnapMirror

If a storage system or volume on a storage system fails, you can recover SQL Server databases that are mirrored using SnapMirror.

Before you begin

- You must have configured SnapMirror to replicate SQL Server database backups to mirrored volumes.
- For each LUN on the SnapMirror source volume, you must have the drive letter mappings on the SQL Server computer.
- For an SMB share, you must have the name of the original share.

Steps

1. If any LUNs from the failed source volume still appear to be connected, disconnect them.
2. Disable the mirror relationship:

If...	Then...
You have LUNs	Use SnapDrive to connect to the corresponding LUNs in the SnapMirror destination volume. Use the same drive letters for connecting to the mirrored LUNs that were used on the source volume. For each mirrored volume, SnapDrive breaks the replica and restores the LUN using the most recent Snapshot copy generated by SnapDrive or SnapManager.
You have SMB shares	**a.** Manually disable the mirror relationship using the Data ONTAP CLI or a management tool. **b.** Create a new share that matches the name of the original share.

3. Restart SQL Server if it has been stopped.
4. Use SQL Server Management Studio to attach the database located on the associated LUNs or SMB shares in the SnapMirror destination volume.
5. If you cannot attach the database on the SnapMirror destination volume and none of the transaction log files were lost, you can reduce the loss of data by ensuring that the last active transaction log of the database is automatically backed up by SnapManager:
   - See Microsoft KB article 253817, "HOW TO: Back up the Last Transaction Log When the Master and the Database Files Are Damaged."
     This article describes how you can back up the currently active transaction log even if the SQL Server database file is damaged, provided that the transaction log file is still accessible.
   - Use this same Microsoft KB article as a general guide for gaining access to the last active transaction log of the database on the SnapMirror destination volume.
     While referring to the steps in that article, observe the following key points:
     - When you create a similar database that contains the same number of data and transaction log files as the original database on the SnapMirror destination volume, you are creating the database you will restore using SnapManager.
     - Instead of using the SQL Server `Backup Log` command to back up the transaction log (as described in the Microsoft article), go to the next step.

   If any of the transaction log files were lost, no workaround is possible and you cannot minimize data loss.
6. If you attached the database on the SnapMirror destination volume, the steps for restoring the database depend on whether the transaction log volume was lost:

If...	Then...
You lost only the data files of the database	Run SnapManager and use the newest full database backup copy to perform either an up-to-the-minute restore or a point-in-time restore: • For an up-to-the-minute restore, SnapManager automatically backs up the last active transaction log before performing the restore operation. • For a point-in-time restore, select the backup set, a combination of transaction log backups to be restored, or both.
If you lost the transaction logs	**a.** Disable the option to back up the transaction log before performing the restore operation: **i.** From the SnapManager menu bar, select **Options** > **Restore Settings**. **ii.** In the Restore Settings dialog box, clear **Create transaction log backup before restore**. **iii.** Click **OK**. The reason you must disable this restore option is that the active transactions were lost due to the failure of the volume containing the transaction log. If the transaction log files were lost, the active transactions are lost and unrecoverable. Because the active transactions are unavailable, you must use SnapManager to perform a point-in-time restore and not an up-to-the minute restore. If you fail to disable this transaction log backup, subsequent SnapManager backup sets reside on a recovery path that is inconsistent with that of the database. Such backup sets cannot be applied to the database; attempts to restore the database from such backup sets results in failure, with the following error message in the restore log: `Failed with error code 0x800410df` **b.** Run SnapManager and use the newest full database backup copy to perform a point-in-time restore. Select the backup set, a combination of transaction log backups to be restored, or both.

Recovering databases on VMDKs using SnapMirror

The disaster recovery of databases on VMDKs involves the disaster recovery of the virtual infrastructure by Virtual Storage Console for VMware vSphere.

Preparing the primary site for recovery

Before creating a backup by SnapManager for SQL Server on the primary site, you must modify the registry keys by completing a series of steps. This enables SnapManager for SQL Server to update the metadata from the primary VSC for VMware vSphere server to the secondary VSC server.

Steps

1. On the primary server, navigate to the location of the registry keys at: `HKEY_LOCAL_MACHINE \SOFTWARE\NetworkAppliance\SnapManager for SQL Server\Server`
2. Change the registry keys as follows:
   - SMVIChangeListFile: The change list file path on the primary virtual machine (for example, `C:\DR\dr_info.txt`).
   - SMVIDestinationBackupXmIUNC: The path of the secondary SMVI server's `backups.xml` path (for example, `\\DestinationSMVIServer\repository\backups.xml`).
   - SMVIDestinationServer: The name or IP of the destination VSC server.
   - SMVISourceBackupXmIUNC: The path of the primary SMVI server's `backups.xml path` (for example, `\\PrimarySMVIServer\repository\backups.xml`).
   - SMVITransformEnable: 1

Preparing the secondary site for recovery

You need to prepare the secondary site for recovery before performing a restore operation.

Steps

1. Install VSC for VMware vSphere.
2. Configure VSC to use the volumes on the destination side (secondary site) storage systems.
3. Enter the vCenter server and storage system IP addresses or names in the **VSC Setup** window.
4. Run the `smvi servercredential set` command from the CLI, if necessary.
5. Stop the VSC service in Windows.

6. Establish the SnapMirror relationship on the underlying volume from the primary site to secondary site.

   Volumes used for VSC on the destination side storage should be used as the SnapMirror destination volumes.

7. Create a Windows share on the repository of both the primary and secondary VSC servers.

   Ensure that the SnapManager service account has Read permission on the share at the primary site and Write and Modify permissions at the share on the disaster recovery site.

8. Create a text file and save the following list information in the file:

   ```
   datastore type datastore name of both sites virtual machine name
   of both sites
   virtual machine uuid of both sites virtual machine directory name
   of both sites storage system name or IP address of both sites
   volume name of both sites| datastore uuid of both sites in case
   of VMFS type of datastore
   ```

   Ensure that all of the above information is in one line per datastore. Each field is separated with a space.

   Note: The virtual machine name and its universal unique identifier (UUID) can be the same if there is no preinstalled standby virtual machine on the disaster recovery site.

9. Save this file to any folder on the primary virtual machine or any other server that the SnapManager service can access through Windows share.

Recovering databases from the secondary site

You need to use VSC to recover the virtual machine before performing the database restore operation in SnapManager.

Steps

1. Break the SnapMirror relationship from the storage system.
2. Bring online the SnapMirror destination volumes on which the datastores reside.
3. Create an NFS export for the NFS storage on the storage system for the destination volume.
4. Add the new NFS export to each of the destination VSC servers on the ESX host.
5. Right click on the datastore and select **Browse data store**.
6. In the left pane, click the virtual machine's name.
7. In the right pane, right-click the virtual machine's VMX file and select the option **Add to Inventory**.

8. Follow the steps in the wizard to add the virtual machine to the ESX server.
9. Power on the virtual machine.
10. Log in to the virtual machine.
11. From the command prompt, enter the following command:

    `sdcli smvi_config list`

    The command lists the primary VSC server.
12. Switch to the secondary VSC server by entering the following command:

    `sdcli smvi_config set -host` ***`IP_of_the_secondary_SMVI_Server`***
13. Restart the SnapDrive for Windows service by using the following commands:

    `net stop swsvc`

    `net start swsvc`
14. After the SnapDrive for Windows service starts successfully, check that all of the VMDKs are available by entering the `sdcli disk list` command.
15. On the recovered virtual machine, run SnapManager for Microsoft SQL Server restore to recover SQL databases.

Cloning databases

Database cloning is the process of creating a point-in-time copy of a production database or its backup set. You might clone databases during application development, to populate data warehouses, or to recover data.

You can use database clones for several reasons:

- During application development cycles, for testing functionality that has to be implemented using the current database structure and content
- For data extraction and manipulation tools when populating data warehouses
- For recovering data that was mistakenly deleted or changed

The database cloning feature enables you to clone specific databases or all databases simultaneously. You can either rename a cloned database or accept the default name provided. You can select the SQL Server instance either from a host on which the database resides or from a remote host connected to the storage system.

Note: If a database resides on a virtual machine with a VMDK disk, SnapManager cannot clone the database to a physical server.

A database cloning operation generates two reports: a backup report and a restore report.

Cloning limitations for VMDKs

Cloning databases that reside on VMDKs using a source volume is similar to cloning databases from LUNs and SMB shares; however, you should be aware of several limitations.

- You cannot perform a database clone on a remote physical server when the database resides on a VMDK.
- You cannot clone a database on a SnapVault secondary because there is no remote backup available for clone operation.
- When the databases are hosted on the VMDKs that are replicated to a site by SnapMirror, you cannot clone databases from a SnapMirror destination volume to the local SQL Server. However, you can clone databases from destination volumes to an SQL Server running on a remote virtual machine.
- SnapManager does not support cloning on SnapMirror destination volumes if the source and destination volumes contain VMFS datastores.

  Note: This limitation applies to clustered Data ONTAP only.

Prerequisites for VMDK verification or cloning on SnapMirror destination volumes

You can verify backup sets on SnapMirror destination volumes and you can clone databases from SnapMirror destination volumes. If the databases that you want to verify or clone are hosted on VMDKs, you must meet several prerequisites before you can perform either of those operations.

You can verify and clone from destination volumes when the database hosted on the VMDKs is replicated to a site by SnapMirror and the configuration meets the following requirements:

- The virtual machine is installed on the ESX server on the secondary site.
- SQL Server, SnapDrive, and SnapManager are installed on the virtual machine.
- The ESX server is managed by another vCenter Server and the VSC server on the secondary site.
- SnapDrive is installed on the secondary virtual machine that is pointing to the VSC server on the secondary site.
- On the primary site, you have selected the SQL Server on the secondary site as the remote verification server.
- On both the primary and secondary VSC servers, you have created a Windows share on the VSC repository folder where the backup metadata file resides.
- The SnapManager service account has read permission on the share at the primary site and write and modify permissions at the share on the secondary site.
- The primary VSC server has discovered the destination storage system.
- For NFS datastores residing on clustered Data ONTAP, a datastore must exist on the SnapMirror destination volume and the name of the destination datastore must be specified in the change list file.
- On the primary virtual machine where the backup is initiated, the following registry settings and values must be defined in `HKEY_LOCAL_MACHINE\SOFTWARE\Network Appliance \SnapManager for SQL Server\Server`:

  SMVITransformEnable

  dword:00000001

  SMVITransformScript

  SMVI_Metadata_update.exe

  SMVIDestinationServer

  `destination SMVI server name`

  SMVISourceBackupXmlUNC

`\\source SMVI server name\SMVI repository share name\backups.xml`

SMVIDestinationBackupXmlUNC

`\\destination SMVI server name\SMVI repository share name\backups.xml`

SMVIChangeListFile

`change list file name`

Requirements for the change list file

The change list file is a text file that contains information about source and destination volumes. The contents of the file must be in the following format, with fields separated by a space and each datastore beginning on a new line:

```
DatastoreType SourceDatastoreName DestinationDatastoreName
SourceVirtualMachineName DestinationVirtualMachineName
SourceVirtualMachineUUID DestinationVirtualMachineUUID
SourceVirtualMachineDirectoryName DestinationVirtualMachineDirectoryName
SourceStorageName DestinationStorageName SourceVolumeName
DestinationVolumeName [SourceDatastoreUUID DestinationDatastoreUUID]
```

In this format, *`Datastore Type`* is either NFS or VMFS and *`Datastore UUID`* is not required for an NFS volume.

Note: For NFS datastores residing on clustered Data ONTAP, *`SourceStorageName`* and *`DestinationStorageName`* must be the IP addresses of the source and destination NFS data LIFs.

The following example shows the contents of a change list file:

```
NFS ds-nfs1 ds-nfs1-dest snapmgr-05-vm2 snapmgr-54-vm1 4211945a-124a-b7c9-
ae63-cacc07f3f4f8 420f010b-7e5a-e66e-7ed1-7bef6a357cca snapmgr-05-vm1
snapmgr-54-vm1 172.17.233.24 172.17.232.74 snapmgr05_vmw1
snapmgr05_vmw1_mir
```

Finding the UUID for VMFS datastore

To find the UUID for VMFS datastore, follow the steps below:

1. Break the SnapMirror relationship from the storage system.
2. Make the SnapMirror destination volumes online on which the datastore reside.
3. On the secondary SMVI server, mount the LUN in the destination volume as the destination datastore on the secondary ESX server.
   When you add the storage from the destination, select the option **Resignature the LUN**.

4. After the destination datastore is added to the secondary ESX server, note the datastore name and UUID values.
5. Replace the destination datastore name and the destination datastore UUID with the new values that you noted in the changelist file.

Cloning a database from a local backup or an archived backup

Cloning a database backup is probably the most commonly used cloning feature. The cloned database can serve as a baseline for developing new applications, or to isolate application errors that occur in the production environment. It can also be used for recovery from soft database errors.

Steps

1. In the SnapManager **Console Root** tree, select a server.
2. In the **Actions** pane, click **Clone Wizard**.

   The SnapManager for SQL Server Clone wizard opens.
3. Click **Next**.
4. On the **Clone Type** page, select **Clone Databases from existing Backup Set** and click **Next**.
5. On the **Backup Selection** page, double-click the backup from which you want to create the clone and then click **Next**.

   Note: The first time you select a database that resides on an LUN, SnapManager automatically selects all databases on the same storage. You can then deselect any databases that you do not want to be cloned.
6. On the **Restore Settings** page, do the following and click **Next**:
   - Select backups to restore.
   - Choose where to apply point-in-time settings.
   - Choose a point-in-time or marked transaction.
7. Click **Next**.

   SnapManager displays the list of databases to be cloned. By default, SnapManager provides the same name to the clone as the original database.
8. Rename the cloned database, and click **Next**.
9. On the **Restore Settings** page, specify the clone database name and click **Next**.

10. On the **Clone to Server** page, specify the clone server name, choose whether you will use a letter drive or a mount point, and click **Next**.

    If you specify a mount point, make sure the directory is empty. If there is a database in the directory, the database will be in an invalid state after the mount.

11. On the **Verification Settings** page, you can do the following:

    - Update SnapMirror after the clone operation completes.
    - Archive the clone to a SnapVault backup.
    - Clone from a SnapVault backup.

12. On the **Restore Settings** page, do any of the following and click **Next**:

    - Click **Clone Restore Settings** to configure advanced settings.
    - Choose whether you want to clone the database on an available SnapMirror destination volume.
    - Choose whether you want to change the clone database paths based on the new database name.

13. On the **Clone Life Cycle Management** page, choose to resynchronize the clone and to automatically delete the clone and define a schedule for each.

    Cloned database resynchronization syncs the cloned database with the live database and automates cloned database deletion. Automatically deleting clones improves resource and storage efficiency by deleting unnecessary clones.

14. On the **Restore Settings** page, select the state of the database you want after the restore operation and click **Next**.

    If you select **Leave the database in read-only mode and available for restoring additional transaction logs**, the **Undo file directory** option activates.

    Note: The default path for the SnapInfo directory in the **Undo file directory** option is that of the source host.

15. To run a command or script before performing the clone operation or after the clone operation finishes, select the **Run Command Settings** option and click **Next**.

16. Click **Finish**.

17. Click **Start Now** to start cloning.

    SnapManager checks off tasks in the Clone Task List as they complete. A message appears indicating the successful completion of the cloning operation.

Cloning a database that is in production

You typically use the clone of a database that is in production when you need to test a new application or function with the latest database content before taking the application into production.

About this task

Cloning a current database involves two steps. First you create the backup of the selected database, then you restore the database from the just-created backup set.

Steps

1. In the SnapManager **Console Root** tree, select a server.
2. In the **Actions** pane, click **Clone Wizard**.
3. On the **Start** page, click **Next**.
4. On the **Clone Type** page, select **Clone Active Production Databases** and click **Next**.

   Note: If you select Run Through Clone QuickStart Wizard, the wizard applies default options for most of the settings.

5. On the **Database Selection** page, double-click the backup from which you want to create the clone, and then click **Next**.

   Note: The first time you select a database that resides on an LUN, SnapManager automatically selects all other databases on the same storage. You can then deselect any databases that you do not want to clone.

6. Continue with the next steps, as instructed in the wizard.
7. If you want to rename the new database clone's paths based on the name of the new database, select the appropriate check box in the wizard.

   Note: You cannot specify database paths for a clone.

8. To perform a clone on a SnapMirror destination volume, select the **Clone on available SnapMirror destination volumes** check box.
9. To run a command or script before performing the clone operation or after the clone operation finishes, select the **Run Command Settings** option.
10. The wizard takes you to the final option that displays the SnapManager clone task list. Click **Start Now** to begin the specified tasks.

    SnapManager checks off tasks in the Clone Task List as they complete. A message appears indicating the successful completion of the cloning operation.

Creating a clone replica of an AlwaysOn cluster

If you need a failover copy of the databases in an AlwaysOn cluster, you can use the SnapManager for SQL Server Availability Group Replica Wizard to create them. After the clone replica is completed, you have a new replica created on the existing availability group as a normal secondary replica.

About this task

The SnapManager for SQL Server Availability Group Replica Wizard uses Snapshot copies to quickly clone the databases to a remote server and then join the clone databases to an Availability Group as a new Availability Group database replica.

Note: A clone replica should be used in the same manner as a clone database, only for temporary purposes.

Steps

1. In the SnapManager **Console Root** tree, select a server.
2. In the **Actions** pane, select **Replica Wizard**.
3. Follow the steps in the **SnapManager for SQL Server Availability Group Replica Wizard** to specify the source, the settings for the replica, and the destination for the replica.

   In the Quick Clone Replica page, you can click **Run through Quick Availability Group Clone Replica Wizard** so that the wizard automatically sets the mandatory settings.

Cloning an already cloned database

You can create a clone of an already cloned database or a database that is in production by using the Clone wizard. You can back up or restore the clone, or you can create a new clone from the already backed-up database.

Steps

1. In the SnapManager **Console Root** tree, select a cloned database.
2. In the **Actions** pane, click **Clone Wizard**.

   The Clone wizard launches and the Welcome window appears.
3. Click **Next**.
4. On the **Clone Type** page, select **Clone Databases from existing Backup set** or **Clone Active Production Databases** and click **Next**.

Note: While choosing Clone Active Production Databases, you can see the Run Through Clone QuickStart Wizard option, and the wizard applies default options for most of the settings.

5. On the **Backup Selection** page, double-click the backup copy (clone) from which you want to create a clone and then click **Next**.

   Note:

   - You cannot select an SQL server database beyond the second level of the clone.
   - When selecting an SQL server database that resides on VMDKs over VMFS, the clone of clone continues to work beyond two levels.
   - You cannot select an SQL server database beyond the second level of clone for backup and restore operations with verification.

6. On the **Restore Settings** page, do the following and click **Next**:
   - Select backups to restore.
   - Choose where to apply point-in-time settings.
   - Choose a point-in-time or marked transaction.
7. Click **Next**.

   SnapManager displays the list of databases to be cloned. By default, SnapManager provides the same name to the clone as the original database.
8. Rename the cloned database and click **Next**.
9. On the **Restore Settings** page, specify the clone database name and click **Next**.
10. On the **Clone to Server** page, specify the clone server name, choose whether to use a letter drive or a mount point, and click **Next**.

    If you specify a mount point, make sure that the directory is empty. If there is a database in the directory, that database will be in an invalid state after the mount.
11. On the **Verification Settings** page, you can do the following:
    - Update SnapMirror after the clone operation completes.
    - Archive the clone to a SnapVault backup.
    - Clone from a SnapVault backup.
12. On the **Restore Settings** page, do any of the following and click **Next**:
    - Click **Clone Restore Settings** to configure advanced settings.

- Choose whether you want to clone the database on an available SnapMirror destination volume.
- Choose whether you want to change the clone database paths based on the new database name.

13. On the **Clone Life Cycle Management** page, choose to resynchronize the clone and to automatically delete the clone and define a schedule for each.

    Cloned database resynchronization synchronizes the cloned database with the live database and automates cloned database deletion. Automatically deleting clones improves resource and storage efficiency by deleting unnecessary clones.

14. On the **Restore Settings** page, select the state of the database you want after the restore operation and click **Next**.

    If you select **Leave the database in read-only mode and available for restoring additional transaction logs**, the **Undo file directory** option activates.

    Note: The default path for the SnapInfo directory in the **Undo file directory** option is that of the source host.

15. To run a command or script before performing the clone operation or after the clone operation finishes, select the **Run Command Settings** option and click **Next**.

16. Click **Finish**.

17. Click **Start Now** to start cloning.

    SnapManager checks off tasks in the Clone Task List as they become complete. A message appears, indicating the successful completion of the cloning operation.

Splitting a cloned database

The split clone feature enables you to split the cloned database from the parent database. After the clone split is completed, the cloned database and parent database are independent of each other, and have their own individual storage space.

Steps

1. In the SnapManager **Console Root** tree, select a cloned database.

2. In the **Actions** pane, click **Clone Wizard**.

   The SnapManager for SQL Server Clone wizard opens.

3. Click **Next**.

4. On the **Clone Type** page, select **Split Cloned Database** and click **Next**.

5. Select the cloned database that you want to split.

   Note: The split clone interface displays all of the clones for VMDK over VMFS, but these clones cannot be split because they are cloned using FlexClone LUNs.

6. Click **Finish**.

   Note: After splitting a cloned database that is cloned from the source database, the split database cannot be migrated to the LUN where the source database is located.

   SnapManager checks off tasks in the Clone Task List as they become complete. A message appears, indicating the successful completion of the split operation.

Deleting cloned databases

You can delete a cloned database that has outlived its purpose. Deleting the cloned database implies that you are disconnecting the Snapshot copy.

About this task

You can also delete clones through **Delete Clone** in the Actions pane.

Steps

1. In the SnapManager **Console Root** tree, select a server.
2. In the **Actions** pane, click **Clone Wizard**.

   The Clone wizard launches and the Welcome window appears.
3. Click **Next**.

   SnapManager displays an option for selecting the operation that you want to perform.
4. Select the operation you want to perform, and click **Next**.

   SnapManager displays the Database to clone window listing the available cloned databases.
5. Select the cloned databases that you want to delete.
6. In the **Delete clone summary** screen, verify the settings selected in the previous steps and click **Finish**.
7. Click **Start now** to begin the specified tasks.

   SnapManager checks off tasks in the Clone Task List as they complete. A message appears indicating the successful completion of the delete cloning operation.

Using SnapManager reports

SnapManager reports list step-by-step details of every SnapManager operation that you perform, their final statuses, and any error messages that you encounter during the operation.

Report	Description
Backup	Contains a log file for every backup set (full database backup or transaction log backup) created by SnapManager
Config	Contains a log file for each time SnapManager migrates a database
Debug	Contains a debugging log in SnapManager when debug logging is enabled
Delete Backup Set	Contains a log file for every delete backup operation
Miscellaneous	Contains log files for all other operations
Monitor	Contains a log file for SnapManager monitoring features
Restore	Contains a log for every restore operation (whether it is a stream-based restore, a copy-based restore, or an online Snapshot restore) performed on an SQL Server that is configured using SnapManager

Viewing SnapManager reports

You can view SnapManager reports to troubleshoot errors with SnapManager operations.

Steps

1. In the SnapManager **Console Root** tree, expand a server and click **Reports**.
2. In the **Reports** pane, expand a folder and select the report you want to display.
3. Use the **Actions** pane to manage a report folder or a specific report.

   For example, you can open a report in Notepad or find specific words in a report.

Configuring monitoring and reporting settings

You can enable SnapManager to send automatic, scheduled email notifications on the status of all backup, verification, and clone operations.

About this task

Each email notification can include the following information:

- A summary of operations
- A summary of individual SQL Server instances, with the number of successful and failed operations
- A list of all operations performed on individual SQL Server instances
- A summary of successful, failed, and “not run” operations on individual SQL Server instances

  Note: For backup and verification operations, incomplete or “not run” operations are logged as an error in the Windows event log, but are logged with an informational message for clone operations.

- A list of all failed operations for individual SQL Server instance

Steps

1. Select a server in the SnapManager **Console Root** tree.
2. In the **Actions** pane, click **Monitor Settings**.

   The Monitoring and Reporting Settings window opens.
3. Select **Enable Monitoring and Reporting**.
4. Choose the operations that you want SnapManager to monitor:

   - **For Backup**
   - **For Verification**
   - **For Clone Resync**

   Note: If you select **For Backup** and **For Clone Resync**, you receive email notifications on all backup and clone operations, but not on verification operations.

5. In the **Select Reporting Interval** pane, specify how often you want to receive notifications.

Example

Select **1** in the days field to receive email notifications once per day.

6. In the **Report operations starting at** field, set the time at which the report operations should start.
7. Click **OK**.

Changing the location of the SnapManager report directory

If you find that there is limited space in the current SnapManager report directory, you can change the report directory to a different location that has more available disk space.

About this task

The reasons for changing the location of the SnapManager report directory include:

- You have limited space.
- You have a clustered environment.
  If you are running SQL Server and SnapManager in a Windows cluster, storing the SnapManager reports in the default location does not allow sharing of the report directory between the nodes in the cluster.

By default, the SnapManager reports are stored in a subdirectory named Report under the directory in which the SnapManager application is installed.

If you change the name or location of the SnapManager report directory, you cannot use the SnapManager Reports option to view or print any reports that were created in that report directory. However, if the previous report directory was not explicitly changed or removed, any reports created in that directory are still accessible. To view or print those older reports, you must change the report directory back to its previous location.

Steps

1. From the **Actions** pane, select **Report Directory Settings**.
2. Specify the new location for the report directory.

   Note the following about the location of the directory:

   - The directory cannot be located on a CIFS share.
   - Do not use a disk that contains SQL Server or SnapManager data for the report directory.
3. Click **OK** to apply your changes.
4. To refresh the information displayed in the **Reports** option, go the **Actions** pane and select **Refresh**.

After you finish

If you changed the report directory for a node in a clustered environment, change the directory on all other nodes in the cluster.

Modifying your database configuration on NetApp storage

After you migrate your databases to NetApp storage, you can modify your database configuration at any time using the SnapManager Configuration wizard.

Moving multiple SnapInfo directories to a single SnapInfo directory

If you previously configured multiple SnapInfo directories, you can rerun the Configuration wizard to move them to a single SnapInfo directory.

About this task

The Configuration wizard enables you to create multiple SnapInfo directories in the following ways:

- A default SnapInfo directory for each SQL Server instance
- A separate SnapInfo directory for multiple databases on one or two LUNs, SMB shares, or VMDKs
- A different (non-default) SnapInfo directory for a database in an SQL Server instance

If you currently have multiple SnapInfo directories, you can choose to combine them into a single directory.

Steps

1. In the **Actions pane**, click **Configuration Wizard**.
2. Complete the following pages without specifying any configuration changes:
   - Verification Settings
   - Database Selection
3. On the **SnapInfo Settings** page, select the **Single SnapInfo Directory** option, and then click **Next**.

   The Specify a Single SnapInfo Directory screen appears. Note that, in the Current SnapInfo Directory list, all the current SnapInfo directories are selected by default.
4. On the **Available Disks** page, select the LUN, SMB share, or VMDK to which you want to move all the SnapInfo directories.

5. Click the **move** (<=>) button.

   The Result SnapInfo Directory dialog box displays the path for the SnapInfo directory. Note that the default directory name is `SMSQL_SnapInfo`.

6. If you want to specify a different location or name, modify the information in the **Result SnapInfo Directory** dialog box.

   Note: The Configuration wizard will allow you to create the SnapInfo directory only in valid locations.

7. Complete the remaining pages of the **Configuration** wizard without specifying any further configuration changes.

8. In the **Finish** page, verify the changes you specified, and then click **Finish**.

9. In the **Configuration** page, click **Start Now**.

10. When a message box is displayed and notifies you that the configuration changes were completed successfully, click **OK** to close the message.

Migrating SQL Server databases back to local disks

If you choose not to use SnapManager as your data management tool, you can migrate your databases back to local disks.

Steps

1. From the **Actions** pane, click **Configuration Wizard Option Settings**.

   The Configuration Wizard Option Settings dialog box opens.

2. Select **Enable databases to be migrated back to local disks**, and click **OK**.

3. In the **Actions** pane, click **Configuration Wizard**.

4. On the **Start** page, click **Next**.

5. On the **Verification Settings** page, click **Next**.

6. On the **Database Selection** page, reassign the databases to a local disk:

   a. In the **Database Location Results** pane, select the databases that you want to move back to a local disk.

   b. Click **Reconfigure**.

   c. In the **Database Selection** pane, select the databases you just reconfigured.

Tip: In the database list, the Disk column displays Reconfig instead of the database location.

d. In the **Disk Selection** pane, select a local drive, and then click the <=> button.

e. Click **Next**.

7. In the **Select SnapInfo File** page, click **Next**.

   Note: Both SnapInfo directories must remain on the LUNs, SMB shares, or VMDKs on which you placed them during the original migration. They cannot be moved to local disks.

8. Complete the remaining pages of the **Configuration** wizard without specifying any further configuration changes.

9. On the **Finish** page, click **Finish**.

10. On the **Configuration** page, click **Start Now** to migrate your databases back to local disk.

Setting up a SnapManager share for centralized backups of transaction logs

A centralized network share makes sure that copies of transaction logs are available to all replicas within the Availability Group and non-Availability Group configurations, providing a centralized backup of transaction logs. If the transaction log backups are created on more than one Availability Group replica, those transaction logs are still accessible and can be used for tasks such as up-to-the-minute restores, database reseeding, and using a clone as a replica.

About this task

If you set up a SnapManager share, SnapManager copies any transaction log backups taken on Availability Group databases to the share, including the log backups taken during secondary Availability Group database migration. These log backups are required to perform an up-to-the-minute restore on another node's Availability Group databases.

You can use the Configuration Wizard to set a network location as a centralized location for copies of transaction logs. At the time logs are backed up, the backups are copied to this share. You can do this either as a step in the initial configuration, or as a separate task.

Steps

1. Start the **Configuration Wizard**.

2. Click **Next** until you reach the **Setup SnapManager Share** window.

3. Check **Enable SnapManager share** and enter or browse to an accessible network share.

Importing or exporting database configurations using a control file

You can use a control file to ease database configuration and modification of SnapManager settings. The control file is an XML file that contains settings for SnapManager configuration. The configuration data is represented in XML format that can be edited manually.

About this task

You can access the control file option from the SnapManager Configuration wizard. The control file is an alternative to manually defining database and SnapManager configurations using the Configuration wizard.

The control file is especially useful in the following scenarios:

- Disaster recovery
- Mass deployment

The configuration settings contained in the control file are grouped into the following sections:

- Storage Layout
- Notification settings
- Verification settings
- Report folder setting
- Backup settings
- Run Command Settings
- SnapMirror Volumes
- Scheduled Jobs
- Clone Scheduled Jobs
- Monitoring and Reporting Settings

Steps

1. On the **Actions** pane, click **Configuration Wizard**.
2. On the **Start** page, select **Use Control File** and click **Next**.
3. On the **Import or Export** page, define the settings for importing or exporting your configuration:

a. Select either the **Import** or the **Export** option.

b. If you are importing a configuration, select **Review settings in configuration wizards** if you want to see a summary of the imported configuration settings and do not want to proceed through the remaining pages of the wizard.

c. In the **Use control file** field, specify the location to import or export the control file.

d. Click **Advanced**.

e. In the **Configuration Import/Export Advanced options** window, specify the configuration settings that you want to import or export and click **OK**.

f. Click **Next** to proceed.

If you chose **export**, the wizard closes and confirms that the file was exported.

4. If you chose **import**, follow the remaining pages in the wizard to import your configuration.

Sample XML schema for the control file settings

The SnapManager schema file is distributed with the installation package. The sample configuration file provided in this topic shows the SnapManager control file settings.

Storage layout settings

The following schema depicts the storage layout settings section. You can edit the storage layout settings using an XML editor.

```
<?xml version="1.0" ?>
- <SMSQLCONFIG xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<HOST_NAME>SNAPMGR-19</HOST_NAME>
- <STORAGE_LAYOUT>
<MAX_DB_JOB>255</MAX_DB_JOB>
- <SQL_INSTANCES>
- <SQL_INSTANCE>
<SQL_INSTANCE_NAME>SNAPMGR-19</SQL_INSTANCE_NAME>
<SQL_INSTANCE_SNAPINFO_PATH>K:\SMSQL_SnapInfo</
SQL_INSTANCE_SNAPINFO_PATH>
<ADD_MSISIC_DEPENDENCY>false</ADD_MSISIC_DEPENDENCY>
- <DATABASES>
- <DATABASE>
<DATABASE_NAME>master</DATABASE_NAME>
- <FILE_GROUPS>
- <FILE_GROUP>
<GROUP_NAME>PRIMARY</GROUP_NAME>
- <DATABASE_FILES>
- <DATABASE_FILE>
<FILE_NAME>master</FILE_NAME>
```

```
<FILE_PATH>C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA
\master.mdf</FILE_PATH>
</DATABASE_FILE>
</DATABASE_FILES>
</FILE_GROUP>
</FILE_GROUPS>
- <LOG_FILES>
- <LOG_FILE>
<FILE_NAME>mastlog</FILE_NAME>
<FILE_PATH>C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA
\mastlog.ldf</FILE_PATH>
</LOG_FILE>
</LOG_FILES>
</DATABASE>
- <DATABASE>
<DATABASE_NAME>tempdb</DATABASE_NAME>
- <FILE_GROUPS>
- <FILE_GROUP>
<GROUP_NAME>PRIMARY</GROUP_NAME>
- <DATABASE_FILES>
- <DATABASE_FILE>
<FILE_NAME>tempdev</FILE_NAME>
<FILE_PATH>C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA
empdb.mdf</FILE_PATH>
</DATABASE_FILE>
</DATABASE_FILES>
</FILE_GROUP>
</FILE_GROUPS>
- <LOG_FILES>
- <LOG_FILE>
<FILE_NAME>templog</FILE_NAME>
<FILE_PATH>C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA
emplog.ldf</FILE_PATH>
</LOG_FILE>
</LOG_FILES>
</DATABASE>
- <DATABASE>
<DATABASE_NAME>model</DATABASE_NAME>
- <FILE_GROUPS>
- <FILE_GROUP>
<GROUP_NAME>PRIMARY</GROUP_NAME>
- <DATABASE_FILES>
- <DATABASE_FILE>
<FILE_NAME>modeldev</FILE_NAME>
<FILE_PATH>C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA
\model.mdf</FILE_PATH>
</DATABASE_FILE>
</DATABASE_FILES>
</FILE_GROUP>
</FILE_GROUPS>
- <LOG_FILES>
- <LOG_FILE>
<FILE_NAME>modellog</FILE_NAME>
<FILE_PATH>C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA
\modellog.ldf</FILE_PATH>
</LOG_FILE>
</LOG_FILES>
```

```
</DATABASE>
- <DATABASE>
<DATABASE_NAME>msdb</DATABASE_NAME>
- <FILE_GROUPS>
- <FILE_GROUP>
<GROUP_NAME>PRIMARY</GROUP_NAME>
- <DATABASE_FILES>
- <DATABASE_FILE>
<FILE_NAME>MSDBData</FILE_NAME>
<FILE_PATH>C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA
\MSDBData.mdf</FILE_PATH>
</DATABASE_FILE>
</DATABASE_FILES>
</FILE_GROUP>
</FILE_GROUPS>
- <LOG_FILES>
- <LOG_FILE>
<FILE_NAME>MSDBLog</FILE_NAME>
<FILE_PATH>C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA
\MSDBLog.ldf</FILE_PATH>
</LOG_FILE>
</LOG_FILES>
</DATABASE>
- <DATABASE>
<DATABASE_NAME>DB1</DATABASE_NAME>
<SNAPINFO_PATH>K:\SMSQL_SnapInfo</SNAPINFO_PATH>
- <FILE_GROUPS>
- <FILE_GROUP>
<GROUP_NAME>PRIMARY</GROUP_NAME>
- <DATABASE_FILES>
- <DATABASE_FILE>
<FILE_NAME>DB1</FILE_NAME>
<FILE_PATH>K:\MP\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA
\DB1.mdf</FILE_PATH>
</DATABASE_FILE>
</DATABASE_FILES>
</FILE_GROUP>
</FILE_GROUPS>
- <LOG_FILES>
- <LOG_FILE>
<FILE_NAME>DB1_log</FILE_NAME>
<FILE_PATH>K:\MP2\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA
\DB1_log.ldf</FILE_PATH>
</LOG_FILE>
</LOG_FILES>
- <DB_VOLUMES>
- <DB_VOLUME>
<FILER_NAME>rhine</FILER_NAME>
<VOLUME_NAME>grace2</VOLUME_NAME>
</DB_VOLUME>
</DB_VOLUMES>
</DATABASE>
- <DATABASE>
<DATABASE_NAME>DB3</DATABASE_NAME>
<SNAPINFO_PATH>K:\SMSQL_SnapInfo</SNAPINFO_PATH>
- <FILE_GROUPS>
- <FILE_GROUP>
```

```
<GROUP_NAME>PRIMARY</GROUP_NAME>
- <DATABASE_FILES>
- <DATABASE_FILE>
<FILE_NAME>DB3</FILE_NAME>
<FILE_PATH>I:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA
\DB3.mdf</FILE_PATH>
</DATABASE_FILE>
</DATABASE_FILES>
</FILE_GROUP>
</FILE_GROUPS>
- <LOG_FILES>
- <LOG_FILE>
<FILE_NAME>DB3_log</FILE_NAME>
<FILE_PATH>I:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA
\DB3_log.ldf</FILE_PATH>
</LOG_FILE>
</LOG_FILES>
- <DB_VOLUMES>
- <DB_VOLUME>
<FILER_NAME>rhine</FILER_NAME>
<VOLUME_NAME>grace1</VOLUME_NAME>
</DB_VOLUME>
</DB_VOLUMES>
</DATABASE>
- <DATABASE>
<DATABASE_NAME>DB2</DATABASE_NAME>
<SNAPINFO_PATH>K:\SMSQL_SnapInfo</SNAPINFO_PATH>
- <FILE_GROUPS>
- <FILE_GROUP>
<GROUP_NAME>PRIMARY</GROUP_NAME>
- <DATABASE_FILES>
- <DATABASE_FILE>
<FILE_NAME>DB2</FILE_NAME>
<FILE_PATH>K:\MP2\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA
\DB2.mdf</FILE_PATH>
</DATABASE_FILE>
</DATABASE_FILES>
</FILE_GROUP>
</FILE_GROUPS>
- <LOG_FILES>
- <LOG_FILE>
<FILE_NAME>DB2_log</FILE_NAME>
<FILE_PATH>K:\MP\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA
\DB2_log.ldf</FILE_PATH>
</LOG_FILE>
</LOG_FILES>
- <DB_VOLUMES>
- <DB_VOLUME>
<FILER_NAME>rhine</FILER_NAME>
<VOLUME_NAME>grace2</VOLUME_NAME>
</DB_VOLUME>
</DB_VOLUMES>
</DATABASE>
- <DATABASE>
<DATABASE_NAME>DB4</DATABASE_NAME>
<SNAPINFO_PATH>K:\SMSQL_SnapInfo</SNAPINFO_PATH>
- <FILE_GROUPS>
```

```
- <FILE_GROUP>
<GROUP_NAME>PRIMARY</GROUP_NAME>
- <DATABASE_FILES>
- <DATABASE_FILE>
<FILE_NAME>DB4</FILE_NAME>
<FILE_PATH>I:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA
\DB4.mdf</FILE_PATH>
</DATABASE_FILE>
</DATABASE_FILES>
</FILE_GROUP>
</FILE_GROUPS>
- <LOG_FILES>
- <LOG_FILE>
<FILE_NAME>DB4_log</FILE_NAME>
<FILE_PATH>I:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA
\DB4_log.ldf</FILE_PATH>
</LOG_FILE>
</LOG_FILES>
- <DB_VOLUMES>
- <DB_VOLUME>
<FILER_NAME>rhine</FILER_NAME>
<VOLUME_NAME>grace1</VOLUME_NAME>
</DB_VOLUME>
</DB_VOLUMES>
</DATABASE>
- <DATABASE>
<DATABASE_NAME>DB5</DATABASE_NAME>
<SNAPINFO_PATH>K:\SMSQL_SnapInfo</SNAPINFO_PATH>
- <FILE_GROUPS>
- <FILE_GROUP>
<GROUP_NAME>PRIMARY</GROUP_NAME>
- <DATABASE_FILES>
- <DATABASE_FILE>
<FILE_NAME>DB5</FILE_NAME>
<FILE_PATH>I:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA
\DB5.mdf</FILE_PATH>
</DATABASE_FILE>
</DATABASE_FILES>
</FILE_GROUP>
</FILE_GROUPS>
- <LOG_FILES>
- <LOG_FILE>
<FILE_NAME>DB5_log</FILE_NAME>
<FILE_PATH>I:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA
\DB5_log.ldf</FILE_PATH>
</LOG_FILE>
</LOG_FILES>
- <DB_VOLUMES>
- <DB_VOLUME>
<FILER_NAME>rhine</FILER_NAME>
<VOLUME_NAME>grace1</VOLUME_NAME>
</DB_VOLUME>
</DB_VOLUMES>
</DATABASE>
</DATABASES>
</SQL_INSTANCE>
- <SQL_INSTANCE>
```

```
<SQL_INSTANCE_NAME>SNAPMGR-19\MARS</SQL_INSTANCE_NAME>
<SQL_INSTANCE_SNAPINFO_PATH>K:\SMSQL_SnapInfo</
SQL_INSTANCE_SNAPINFO_PATH>
<ADD_MSISIC_DEPENDENCY>false</ADD_MSISIC_DEPENDENCY>
- <DATABASES>
- <DATABASE>
<DATABASE_NAME>master</DATABASE_NAME>
- <FILE_GROUPS>
- <FILE_GROUP>
<GROUP_NAME>PRIMARY</GROUP_NAME>
- <DATABASE_FILES>
- <DATABASE_FILE>
<FILE_NAME>master</FILE_NAME>
<FILE_PATH>C:\Program Files\Microsoft SQL Server\MSSQL.2\MSSQL\DATA
\master.mdf</FILE_PATH>
</DATABASE_FILE>
</DATABASE_FILES>
</FILE_GROUP>
</FILE_GROUPS>
- <LOG_FILES>
- <LOG_FILE>
<FILE_NAME>mastlog</FILE_NAME>
<FILE_PATH>C:\Program Files\Microsoft SQL Server\MSSQL.2\MSSQL\DATA
\mastlog.ldf</FILE_PATH>
</LOG_FILE>
</LOG_FILES>
</DATABASE>
- <DATABASE>
<DATABASE_NAME>tempdb</DATABASE_NAME>
- <FILE_GROUPS>
- <FILE_GROUP>
<GROUP_NAME>PRIMARY</GROUP_NAME>
- <DATABASE_FILES>
- <DATABASE_FILE>
<FILE_NAME>tempdev</FILE_NAME>
<FILE_PATH>C:\Program Files\Microsoft SQL Server\MSSQL.2\MSSQL\DATA
empdb.mdf</FILE_PATH>
</DATABASE_FILE>
</DATABASE_FILES>
</FILE_GROUP>
</FILE_GROUPS>
- <LOG_FILES>
- <LOG_FILE>
<FILE_NAME>templog</FILE_NAME>
<FILE_PATH>C:\Program Files\Microsoft SQL Server\MSSQL.2\MSSQL\DATA
emplog.ldf</FILE_PATH>
</LOG_FILE>
</LOG_FILES>
</DATABASE>
- <DATABASE>
<DATABASE_NAME>model</DATABASE_NAME>
- <FILE_GROUPS>
- <FILE_GROUP>
<GROUP_NAME>PRIMARY</GROUP_NAME>
- <DATABASE_FILES>
- <DATABASE_FILE>
<FILE_NAME>modeldev</FILE_NAME>
```

```
<FILE_PATH>C:\Program Files\Microsoft SQL Server\MSSQL.2\MSSQL\DATA
\model.mdf</FILE_PATH>
</DATABASE_FILE>
</DATABASE_FILES>
</FILE_GROUP>
</FILE_GROUPS>
- <LOG_FILES>
- <LOG_FILE>
<FILE_NAME>modellog</FILE_NAME>
<FILE_PATH>C:\Program Files\Microsoft SQL Server\MSSQL.2\MSSQL\DATA
\modellog.ldf</FILE_PATH>
</LOG_FILE>
</LOG_FILES>
</DATABASE>
- <DATABASE>
<DATABASE_NAME>msdb</DATABASE_NAME>
- <FILE_GROUPS>
- <FILE_GROUP>
<GROUP_NAME>PRIMARY</GROUP_NAME>
- <DATABASE_FILES>
- <DATABASE_FILE>
<FILE_NAME>MSDBData</FILE_NAME>
<FILE_PATH>C:\Program Files\Microsoft SQL Server\MSSQL.2\MSSQL\DATA
\MSDBData.mdf</FILE_PATH>
</DATABASE_FILE>
</DATABASE_FILES>
</FILE_GROUP>
</FILE_GROUPS>
- <LOG_FILES>
- <LOG_FILE>
<FILE_NAME>MSDBLog</FILE_NAME>
<FILE_PATH>C:\Program Files\Microsoft SQL Server\MSSQL.2\MSSQL\DATA
\MSDBLog.ldf</FILE_PATH>
</LOG_FILE>
</LOG_FILES>
</DATABASE>
</DATABASES>
</SQL_INSTANCE>
</SQL_INSTANCES>
</STORAGE_LAYOUT>
```

Notification settings

The following schema depicts the notification settings section:

```
-<COMMON_SETTINGS>
-<NOTIFICATION>
-<SEND_EMAIL_NOTIFICATION>
<SMTP_SERVER>SNAPMGR-19</SMTP_SERVER>
<FROM>SMSQLAutoSender</FROM>
<TO>autosupport@netapp.com</TO>
<SUBJECT>SnapManager for SQL Server</SUBJECT>
<NOTIFY_AUTO>true</NOTIFY_AUTO>
<LONG_MSG>false</LONG_MSG>
```

```
<AS_ATTACHMENT>false</AS_ATTACHMENT>
<SEND_ON_FAILURE>true</SEND_ON_FAILURE>
</SEND_EMAIL_NOTIFICATION>
<EMS_ENABLED>true</EMS_ENABLED>
<ASUP_ENABLED>true</ASUP_ENABLED>
<ASUP_ON_FAIL>true</ASUP_ON_FAIL>
</NOTIFICATION>
```

Verification settings

The following schema depicts the verification settings section:

```
-<VERIFICATION>
-<VERIFICATION_CLIENT_SETTING>
<VERIFICATION_SERVER>SNAPMGR-19</VERIFICATION_SERVER>
<VER_SERVER_NTAUTH>true</VER_SERVER_NTAUTH>
<VER_DBCC_NOINDEX>false</VER_DBCC_NOINDEX>
<VER_DBCC_ALL_ERROR_MSG>false</VER_DBCC_ALL_ERROR_MSG>
<VER_DBCC_NO_INFO_MSGS>false</VER_DBCC_NO_INFO_MSGS>
<VER_DBCC_TABLOCK>false</VER_DBCC_TABLOCK>
<VER_DBCC_PHYSICAL_ONLY>false</VER_DBCC_PHYSICAL_ONLY>
<VER_DBCC_ATTACH_DB>false</VER_DBCC_ATTACH_DB>
<VER_DBCC_BEFORE_MIGRATION>true</VER_DBCC_BEFORE_MIGRATION>
<VER_DBCC_AFTER_MIGRATION>false</VER_DBCC_AFTER_MIGRATION>
<VER_DELETE_DB_FILE_ORIG>true</VER_DELETE_DB_FILE_ORIG>
<VER_RUN_UPDATE_STATISTICS>true</VER_RUN_UPDATE_STATISTICS>
</VERIFICATION_CLIENT_SETTING>
-<VERIFICATION_SERVER_SETTING>
<AUTO_DRIVELETTER>true</AUTO_DRIVELETTER>
<MP_DIR>C:\Program Files\NetApp\SnapManager for SQL Server
\SnapMgrMountPoint</MP_DIR>
</VERIFICATION_SERVER_SETTING>
</VERIFICATION>
```

Monitoring directory settings

The following schema depicts the monitoring directory settings:

```
- <MONITORING.>
<REPORT_BACKUP> true</REPORT_BACKUP>
<REPORT_CLONE>false</REPORT_CLONE>
<INTERVAL_HOURS>1</INTERVAl_HOURS>
<REPORT_CLOCK>23:15:00</REPORT_CLOCK>
</MONITORING>
```

Report directory settings

The following schema depicts the report directory settings section:

```
<REPORT_DIRECTORY>C:\Program Files\NetApp\SnapManager for SQL Server
\Report</REPORT_DIRECTORY>
```

Backup settings

The following schema depicts the backup settings section:

```
-<BACKUP>
-<BACKUP_CLIENT_SETTING>
<NAMING_CONVENTION>0</NAMING_CONVENTION>
<BACKUP_SET_TO_KEEP>3</BACKUP_SET_TO_KEEP>
<BACKUP_SET_TO_KEEP_IN_DAYS>0</BACKUP_SET_TO_KEEP_IN_DAYS>
<LOG_BACKUP_SET_TO_KEEP>7</
LOG_BACKUP_SET_TO_KEEP><LOG_BACKUP_SET_TO_KEEP_IN_DAYS>0</
LOG_BACKUP_SET_TO_KEEP_IN_DAYS><DELETE_BACKUPS_OPTION>0</
DELETE_BACKUPS_OPTION>
<DELETE_LOG_BACKUPS_OPTION>0</
DELETE_LOG_BACKUPS_OPTION><BACKUP_SET_TO_VERIFY>0</BACKUP_SET_TO_VERIFY>
<BACKUP_SET_TO_KEEP_UTM>8</BACKUP_SET_TO_KEEP_UTM>
<BACKUP_SET_TO_KEEP_IN_DAYS_UTM/>
<DELETE_BACKUPS_OPTION_UTM>0</DELETE_BACKUPS_OPTION_UTM></
BACKUP_CLIENT_SETTING>
-<BACKUP_SERVER_SETTING>
<RUN_CMD_HOST>SNAPMGR-19</RUN_CMD_HOST> <RUN_CMD_PATH>notepad.exe</
RUN_CMD_PATH>
<RUN_CMD_ARGUMENT>$SqlSnapshot $InfoSnapshot</RUN_CMD_ARGUMENT>
</BACKUP_SERVER_SETTING>
</BACKUP>
```

SnapMirror volumes settings

The following schema depicts the SnapMirror relationship settings section:

```
-<VERIFICATION_ON_DESTINATION>
-<SELECTED_DESTINATIONS>
-<SELECTED_DESTINATION>
<SOURCE_FILER>rhine</SOURCE_FILER>
<SOURCE_VOLUME>grace2</SOURCE_VOLUME>
<DESTINATION_FILER>rhine</DESTINATION_FILER>
<DESTINATION_VOLUME>grace2_mir</DESTINATION_VOLUME>
</SELECTED_DESTINATION>
-<SELECTED_DESTINATION>
<SOURCE_FILER>rhine</SOURCE_FILER>
<SOURCE_VOLUME>grace2</SOURCE_VOLUME>
<DESTINATION_FILER>rhine</DESTINATION_FILER>
<DESTINATION_VOLUME>grace2_mir</DESTINATION_VOLUME>
</SELECTED_DESTINATION>
</SELECTED_DESTINATIONS>
</VERIFICATION_ON_DESTINATION>
```

Schedule job settings

The following schema depicts the schedule job settings section:


```
-<VERIFICATION_ON_DESTINATION>
-<SELECTED_DESTINATIONS>
-<SELECTED_DESTINATION>
<SOURCE_FILER>rhine</SOURCE_FILER>
<SOURCE_VOLUME>grace2</SOURCE_VOLUME>
<DESTINATION_FILER>rhine</DESTINATION_FILER>
<DESTINATION_VOLUME>grace2_mir</DESTINATION_VOLUME>
</SELECTED_DESTINATION>
-<SELECTED_DESTINATION>
<SOURCE_FILER>rhine</SOURCE_FILER>
<SOURCE_VOLUME>grace2</SOURCE_VOLUME>
<DESTINATION_FILER>rhine</DESTINATION_FILER>
<DESTINATION_VOLUME>grace2_mir</DESTINATION_VOLUME>
</SELECTED_DESTINATION>
</SELECTED_DESTINATIONS>
</VERIFICATION_ON_DESTINATION>
-<SCHEDULE_JOBS>
-<JOB>
<SCHEDULER>Windows Task Scheduler</SCHEDULER>
<JOB_NAME>bkup1</JOB_NAME>
<HOST_NAME>snapmgr-19</HOST_NAME>
<START_DIR>C:\Program Files\NetApp\SnapManager for SQL Server\</
START_DIR> <APPLICATION_NAME>C:\Program Files\NetApp\SnapManager for SQL
Server\SMSQLJobLauncher.exe</APPLICATION_NAME>

<COMMAND>new-backup ñsvr 'SNAPMGR-19' -db 'SNAPMGR-19', '8', 'DB1',
'DB2', 'DB3', 'DB4', 'DB5', 'master', 'model', 'msdb',
'SNAPMGR-19\MARS', '3', 'master', 'model', 'msdb' -ver ñversvr
'SNAPMGR-19' -del -rtbkups 2 -lgbkafbk -noutm -uniq ñmgmt standard</
COMMAND>
<START_TIME>11/6/2007 1:32:00 PM</START_TIME>
-<SCHEDULES>
-<WEEKLY_TRIGGERS>
-<WEEKLY_TRIGGER>
-<TASK_TRIGGER>
<TriggerSize>48</TriggerSize>
<Reserved1>0</Reserved1>
<BeginYear>2007</BeginYear>
<BeginMonth>10</BeginMonth>
<BeginDay>27</BeginDay>
<EndYear>0</EndYear>
<EndMonth>0</EndMonth>
<EndDay>0</EndDay>
<StartHour>13</StartHour>
<StartMinute>32</StartMinute>
<MinutesDuration>0</MinutesDuration>
<MinutesInterval>0</MinutesInterval>
<Flags>0</Flags>
<Type>TIME_TRIGGER_WEEKLY</Type>
-<Data>
-<daily>
```

```
<DaysInterval>1</DaysInterval>
</daily>
-<weekly>
<WeeksInterval>1</WeeksInterval>
<DaysOfTheWeek>4</DaysOfTheWeek>
</weekly>
-<monthlyDate>
<Days>262145</Days>
<Months>0</Months>
</monthlyDate>
-<monthlyDOW>
<WhichWeek>1</WhichWeek>
<DaysOfTheWeek>4</DaysOfTheWeek>
<Months>0</Months>
</monthlyDOW>
</Data>
<Reserved2>0</Reserved2>
<RandomMinutesInterval>0</RandomMinutesInterval>
</TASK_TRIGGER>
</WEEKLY_TRIGGER>
</WEEKLY_TRIGGERS>
</SCHEDULES>
</JOB>
-<JOB>
<SCHEDULER>SQL Server Agent</SCHEDULER>
<JOB_NAME>bkupSqlAgt</JOB_NAME>
<HOST_NAME>SNAPMGR-19</HOST_NAME>
<START_DIR>C:\Program Files\NetApp\SnapManager for SQL Server\</
START_DIR> <APPLICATION_NAME>C:\Program Files\NetApp\SnapManager for SQL
Server\SMSQLJobLauncher.exe</APPLICATION_NAME>
<COMMAND>ackup ñsvr 'SNAPMGR-19' -db 'SNAPMGR-19', '8', 'DB1', 'DB2',
'DB3', 'DB4', 'DB5', 'master', 'model', 'msdb', 'SNAPMGR-19\MARS', '3',
'master', 'model', 'msdb' -ver ñversvr 'SNAPMGR-19' -del -rtbkups 2 -
lgbkafbk -noutm -uniq ñmgmt standard</COMMAND>
<START_TIME>11/7/2007 1:00:00 AM</START_TIME>
-<SQLAGENTSCHEDULES>
<START_DATE_TIME>11/5/2007 12:00:00 AM</START_DATE_TIME>
<START_TIME_OF_DAY>01:00:00</START_TIME_OF_DAY>
<END_DATE_TIME>12/31/9999 12:00:00 AM</END_DATE_TIME>
<END_TIME_OF_DAY>23:59:59</END_TIME_OF_DAY>
<FREQUENCY_TYPE>Daily</FREQUENCY_TYPE>
<FREQUENCY_INTERVAL>1</FREQUENCY_INTERVAL>
<FREQUENCY_SUBDAY_TYPE>Once</FREQUENCY_SUBDAY_TYPE>
<FREQUENCY_SUBDAY_INTERVAL>0</FREQUENCY_SUBDAY_INTERVAL>
<FREQUENCY_RELATIVE_INTERVAL>First</FREQUENCY_RELATIVE_INTERVAL>
<FREQUENCY_RECURRENCE_FACTOR>0</FREQUENCY_RECURRENCE_FACTOR>
</SQLAGENTSCHEDULES>
</JOB>
-<JOB>
<SCHEDULER>SQL Server Agent</SCHEDULER>
<JOB_NAME>bkupSqlAgtMars</JOB_NAME>
<HOST_NAME>SNAPMGR-19\MARS</HOST_NAME>
<START_DIR>C:\Program Files\NetApp\SnapManager for SQL Server\</
START_DIR> <APPLICATION_NAME>C:\Program Files\NetApp\SnapManager for SQL
Server\SMSQLJobLauncher.exe</APPLICATION_NAME>
<COMMAND>backup ñsvr 'SNAPMGR-19' -db 'SNAPMGR-19', '8', 'DB1', 'DB2',
'DB3', 'DB4', 'DB5', 'master', 'model', 'msdb', 'SNAPMGR-19\MARS', '3',
```

```
'master', 'model', 'msdb' -ver ñversvr 'SNAPMGR-19' -del -rtbkups 2 -
lgbkafbk -noutm -uniq ñmgmt standard</COMMAND>
<START_TIME>11/11/2007 2:00:00 AM</START_TIME>
-<SQLAGENTSCHEDULES>
<START_DATE_TIME>11/5/2007 12:00:00 AM</START_DATE_TIME>
<START_TIME_OF_DAY>02:00:00</START_TIME_OF_DAY>
<END_DATE_TIME>12/31/9999 12:00:00 AM</END_DATE_TIME>
<END_TIME_OF_DAY>23:59:59</END_TIME_OF_DAY>
<FREQUENCY_TYPE>Weekly</FREQUENCY_TYPE>
<FREQUENCY_INTERVAL>1</FREQUENCY_INTERVAL>
<FREQUENCY_SUBDAY_TYPE>Once</FREQUENCY_SUBDAY_TYPE>
<FREQUENCY_SUBDAY_INTERVAL>0</FREQUENCY_SUBDAY_INTERVAL>
<FREQUENCY_RELATIVE_INTERVAL>First</FREQUENCY_RELATIVE_INTERVAL>
<FREQUENCY_RECURRENCE_FACTOR>1</FREQUENCY_RECURRENCE_FACTOR>
</SQLAGENTSCHEDULES>
</JOB>
</SCHEDULE_JOBS>
</COMMON_SETTINGS>
</SMSQLCONFIG>
```

Clone job settings

The following schema depicts the clone job settings section:

```
<?xml version="1.0"?>
<SMSQLCONFIG xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <HOST_NAME>W2K8R2SP1X64</HOST_NAME>
  <COMMON_SETTINGS>
    <SCHEDULE_JOBS>
      <JOB>
        <SCHEDULER>SQL Server Agent</SCHEDULER>
        <JOB_NAME>CloneAutoDel__abc__09-17-2012_21-38-44</JOB_NAME>
        <HOST_NAME>W2K8R2SP1X64</HOST_NAME>
        <START_DIR>"C:\Program Files\NetApp\SnapManager for SQL Server\
        </START_DIR>
        <APPLICATION_NAME>
          "C:\Program Files\NetApp\SnapManager for SQL Server
          \SmsqlJobLauncher.exe
        </APPLICATION_NAME>
        <COMMAND>delete-clone –svr 'W2K8R2SP1X64' -inst 'W2K8R2SP1X64'
          -d 'abc__Clone' -JobInstance 'W2K8R2SP1X64'
          -ResyncCloneJob
          'CloneResync__abc__09-17-2012_21-38-44'</COMMAND>
        <START_TIME>9/18/2012 9:38:37 PM</START_TIME>
        <SQLAGENTSCHEDULES>
          <START_DATE_TIME>20120918</START_DATE_TIME>
          <START_TIME_OF_DAY>213837</START_TIME_OF_DAY>
          <END_DATE_TIME>99991231</END_DATE_TIME>
          <END_TIME_OF_DAY>235959</END_TIME_OF_DAY>
          <FREQUENCY_TYPE>OneTime</FREQUENCY_TYPE>
          <FREQUENCY_INTERVAL>0</FREQUENCY_INTERVAL>
          <FREQUENCY_SUBDAY_TYPE>Unknown</FREQUENCY_SUBDAY_TYPE>
          <FREQUENCY_SUBDAY_INTERVAL>0</FREQUENCY_SUBDAY_INTERVAL>
          <FREQUENCY_RELATIVE_INTERVAL>First</
```

```
FREQUENCY_RELATIVE_INTERVAL>
          <FREQUENCY_RECURRENCE_FACTOR>0</FREQUENCY_RECURRENCE_FACTOR>
        </SQLAGENTSCHEDULES>
      </JOB>
      <JOB>
        <SCHEDULER>SQL Server Agent</SCHEDULER>
        <JOB_NAME>CloneResync__abc__09-17-2012_21-38-44</JOB_NAME>
        <HOST_NAME>W2K8R2SP1X64</HOST_NAME>
        <START_DIR>
          "C:\Program Files\NetApp\SnapManager for SQL Server\
        </START_DIR>
        <APPLICATION_NAME>"C:\Program Files\NetApp\SnapManager for
          SQL Server\SmsqlJobLauncher.exe</APPLICATION_NAME>
        <COMMAND>clone-database –svr
          'W2K8R2SP1X64' –inst 'W2K8R2SP1X64' -d 'abc'
          -tgInst 'W2K8R2SP1X64' -tgDb 'abc__Clone'
          -tgmpdir 'C:\Program Files\NetApp
          \SnapManager for SQL
            Server\SnapMgrMountPoint' -Resynchronize
          -ForceTerminateConnection -ver
          –verInst 'W2K8R2SP1X64' -mp
          –mpdir 'C:\Program Files\NetApp
          \SnapManager for SQL
          Server\SnapMgrMountPoint' -RetainShareBackups 7
          –mgmt standard </COMMAND>
        <START_TIME>9/18/2012 9:38:37 PM</START_TIME>
        <SQLAGENTSCHEDULES>
          <START_DATE_TIME>20120917</START_DATE_TIME>
          <START_TIME_OF_DAY>213837</START_TIME_OF_DAY>
          <END_DATE_TIME>99991231</END_DATE_TIME>
          <END_TIME_OF_DAY>235959</END_TIME_OF_DAY>
          <FREQUENCY_TYPE>Daily</FREQUENCY_TYPE>
          <FREQUENCY_INTERVAL>1</FREQUENCY_INTERVAL>
          <FREQUENCY_SUBDAY_TYPE>Hour</FREQUENCY_SUBDAY_TYPE>
          <FREQUENCY_SUBDAY_INTERVAL>12</FREQUENCY_SUBDAY_INTERVAL>
          <FREQUENCY_RELATIVE_INTERVAL>First</
FREQUENCY_RELATIVE_INTERVAL>
          <FREQUENCY_RECURRENCE_FACTOR>0</FREQUENCY_RECURRENCE_FACTOR>
        </SQLAGENTSCHEDULES>
      </JOB>
    </SCHEDULE_JOBS>
  </COMMON_SETTINGS>
</SMSQLCONFIG>
```

Configuring SnapManager application settings

You can configure or change SnapManager application settings at any time after you install SnapManager as long as you run SnapManager from the system console and not from a Terminal Services client.

Modifying backup settings

You can modify the settings that SnapManager uses to perform backup operations. The settings include how SnapManager names Snapshot copies, when it verifies database backups, and how SnapManager backs up transaction logs.

Steps

1. In the **Actions** pane, click **Backup Settings**.
2. In the **Full Database Backup** tab, specify settings for full database backups:
   a. Choose a backup naming convention:

Convention	Description
Generic	Adds the string “recent” to the name of the most recent Snapshot copy. All other Snapshot copies include a time stamp. If you enable the enhanced Snapshot copy naming convention by clicking **Advanced**, the most recent backup copy per management group (daily, weekly, standard) has the suffix “__recent”. For example, once enhanced Snapshot copy naming is enabled, backup copies will have the following names: • `sqlsnap__`*`SqlServerName`*`__Daily__recent` • `sqlsnap__`*`SqlServerName`*`__Weekly__recent` • `sqlsnap__`*`SqlServerName`*`__recent` **Note:** You cannot disable the Snapshot copy naming convention after you enable it.
Unique	Adds a time stamp to all Snapshot copy names. This is the default option and is almost always the best choice.

The naming convention you select applies to all backups. You should use the unique naming convention unless you have a script that requires the constant string “recent”.

b. Keep both of the **Run DBCC physical integrity verification...** options unchecked.

It is best to verify databases in an operation separate from the database backup operation.

3. In the **Transaction Log Backup** tab, specify settings for transaction log backups, including how long you want to retain SnapInfo Snapshot copies, whether you want to copy transaction log backup copies to a share, and how long you want to retain the backup copies on the share.

4. In the **Backup Concurrency** tab, keep the default setting or enter a smaller number, if needed for performance reasons.

Microsoft recommends setting a maximum of 35 databases per backup Snapshot copy if SQL Server thread resources are strained.

Note: During backup, SnapManager might use a different number of maximum databases per Snapshot copy than what is configured in the Backup Settings dialog box.

This happens because SnapManager tries to use the smallest number of Snapshot copies as possible. For example, if the maximum databases per Snapshot copy setting is 35 and there are 45 databases to back up, SnapManager might back up all 45 databases in the same Snapshot copy operation.

Modifying verification settings

You can use the Verification Settings dialog box to specify the verification server and configure database verification options.

About this task

When you change the database verification server, the change does not affect any database verification jobs that are already scheduled.

Steps

1. In the **Actions** pane, click **Backup Verification Settings**.

2. In the **Verification Settings** tab, define how SnapManager should verify backup copies:

For this field...	Do this...
Verification Server	For optimal performance, choose a remote verification server, which offloads work from the SQL production server.

For this field...	Do this...
SQL Server Connection	Choose an authentication method for SnapManager to connect to the SQL Server on the verification server during backup verification: **Use Windows authentication** SnapManager connects to the SQL Server using the Windows account under which SnapManager runs (the SnapManager service account). This is the most common method. **Use SQL Server authentication** SnapManager connects to the SQL Server using an account defined on the SQL Server. The account must have sysadmin server role privileges on the SQL Server instance.
Access a mounted LUN in snapshot	Keep the default option for mounting Snapshot copies to an empty NTFS directory. SnapManager mounts Snapshot copies to the verification server when it verifies backup copies. Using an empty NTFS directory is typically better than assigning drive letters because the verification server can run out of drive letters if there are more backup copies than available drive letters. For a Windows Failover Cluster, the mount point directory must be a shared disk.

3. In the **SnapMirror and SnapVault Options** tab, click **Verification on Destination Volumes** and choose the volumes.
4. In the **DBCC options** tab, specify the DBCC options that SnapManager uses to verify backup sets.

   For more information about DBCC options, see your Microsoft SQL Server documentation.

   Note: PHYSICAL_ONLY and **NO_INFOMGS** are selected by default.

Modifying restore settings

You can use the Restore Settings dialog box to configure default settings for SnapManager restore operations.

Steps

1. In the **Actions** pane, select **Restore Setting**.

   The Restore Settings dialog box appears. Only the **Create transaction log backup before restore** check box is selected by default.

2. Select any combination of the restore options you want to use:

Option	Description
Recover database without restoring at the end of restore operation if needed	If the database is not fully operational and you want to leave it operational after restore, SnapManager skips the restore operation and performs the recover operation.
Restore databases even if existing databases are online	If an existing database is online at the time of the restore operation, SnapManager proceeds with the restore operation and overwrites the existing database.
Retain SQL database replication settings	If you restore databases for an SQL Server instance that is acting as a Publisher or as a Subscriber in a replication topology, the replication relationship is retained after the SnapManager restore operation finishes.
Create transaction log backup before restore operation	If this option is not selected, SnapManager does not create a transaction log backup before the restore operation is performed, thereby decreasing overall restore time. Clear this option under the following circumstances: • You are recovering from a mirrored backup for which the transaction log files were lost. Disabling this option avoids subsequent creation of SnapManager backup sets on a recovery path that is inconsistent with that of the database. • You are restoring a log-shipped database.
Abort database restore operation if transaction log backup before restore fails	If the transaction log backup fails, SnapManager aborts the database restore operations.
Ignore log backups from SMSQL Repository Share	Log backups on the repository share are not used in the restore.

3. Click **OK**.

   The new settings will be applied to all subsequent database restore operations.

Modifying event notification settings

You can use the Notification Settings dialog box to enable and configure the SnapManager event notification services.

Steps

1. In the **Actions** pane, click **Notification Settings**.
2. In the **Notification Settings** dialog box, configure the settings for email notifications, event logging, and AutoSupport notifications.

   Most fields in this dialog box are self-explanatory. The following table describes fields for which you might need guidance:

Field	Description
Send e-mail notification	Enables email notifications to the specified address about the success or failure of SnapManager operations. If you enable this field, click **Advanced** to tune the notification settings—for example, to receive notifications only when operations fail.
Log SnapManager events to storage system syslog	Posts SnapManager events to the storage system's event log, if AutoSupport is enabled on the storage system. Technical support can use this information to troubleshoot issues.
Send AutoSupport notification	Enables email notifications to technical support about SnapManager events or storage system problems that might occur, if AutoSupport is enabled on the storage system.
On failure only	Limits the SnapManager events that are posted to the storage system event log and sent through AutoSupport to failure events only.

3. Click **OK**.

Setting defaults for preoperation and postoperation commands

When starting a SnapManager backup, database verification, restore, or clone operation, you can instruct the system to automatically run a command or script either before the operation starts or after it is complete. You can set defaults for the commands.

Steps

1. In the **Actions** pane, click **Run Command Settings**.

SnapManager displays the Configure Default Run Commands dialog box with the current default settings.

2. Select the operation (for example, backup, verify, restore, or clone) for which you want these default settings to apply.
3. Select **Pre-Operation Command** or **Post-Operation Command**, depending on whether you want to run the command before or after the operation.
4. If you want the SnapManager for SQL Server operation to stop when an error occurs in the custom user command, select **Treat pre command errors as fatal by stopping the remaining SnapManager operation**.
5. In the **Specify a computer where...** box, enter or browse to the name of the host on which your program or script resides.
6. In the **Specify the full path...** box, select your program or script.
7. Enter the command input string in the **Command Arguments** box.

   You can do this using any combination of the following methods:

   - To enter text directly into the Command Arguments box, click the box and type the desired text.
   - To enter a SnapManager variable into the Command Arguments box, do the following:

     a. If necessary, click in the Command Arguments box to position the cursor.

     b. In the SnapManager Variables list, select the variable you want to enter.

     c. Click **Select**.

     Note: Several parameters, like the `$SnapInfoPath` and `$LogBackupFile` variables, are enclosed within double quotes by default so that the actual path name can contain spaces without affecting the script invocation on the Windows command line. If you do not want the double quotes to appear in your command line, remove them from the Command Arguments field in the Run Commands dialog box.

8. Click **Save** to apply your changes and then click **Close**.

   Your changes are saved as the default.

SnapManager arguments for preoperation and postoperation commands

You can use several arguments for commands that you want to run before or after a SnapManager operation.

Precommand arguments

The following precommand arguments apply to backup, verify, restore, and clone operations:

Variable	Description

`$Database`	Specifies the logical name of the database processes.
	To prevent PowerShell from interpreting the value of this parameter, be sure to enclose the entire parameter value with single quotes: `-PreCmdArg '$Database $ServerInstance'`.
	Example:
	`DatabaseAccounting`
	If you want to have more than one database expanded, repeat the parameter as many times as you want.
	Example:
	`AccountingDB1 AcmeServer1/SqlInst1 FinanceDB2 AcmeServer1/SqlInst2`

`$ServerInstance`	Specifies the name of the SQL server instance that is actually processed. Example: `ACMESERVER1\SQLINSTANCE1`

Postcommand arguments

The following postcommand arguments apply to backup, verify, restore, and clone operations.

Note: To prevent PowerShell from interpreting the value of a parameter, be sure to enclose the entire parameter value within single quotes: `-PostCmdArg ‘$Database $ServerInstance $SqlSnapshot'`

Variable	Description

`$InfoSnapshot`	Expands to the name of a SnapInfo directory Snapshot copy.
	Examples:
	`sqlinfo__winsrvr2__01-31-2014_15.03.09`
	`sqlinfo__winsrvr2__recent`

Variable	Description
`$LogBackupFile`	Expands to the full path name of the transaction log backup file. Example: `I:\SMSQL_SnapInfo\SQL__WINSRVR2\DB__Northwind \LogBackup\ 11-01-2004_13.34.59__Northwind.TRB`
`$OperationStatus`	Provides the status of the SMSQL operation. Example: 5234
`$PreCommandStatus`	Provides the precommand status to the postcommand if the postcommand is executed based on the status of the earlier precommand. Example: 5234

`$SnapInfoName`	Expands to the name of the SnapInfo directory.
	Examples:
	`WINSRVR2__recent`
	`WINSRVR2_11-23-2013_16.21.07__Daily`
	If you use this variable, you must also provide the correct path to the directory.

`$SnapInfoPath`	Expands to the name of the SnapInfo subdirectory. This argument is used in backup and verification operations.
	Example:
	`I:\SMSQL_SnapInfo\SQL__WINSRVR2\DB__Northwind`
	For restore and clone operations, this argument specifies the path to the Snapshot copy information metadata that is being used for the database restore.
	Example:
	`U:\SMSQL_SnapInfo\VDISK__E\FG__ \05-14-2010_15.33.41\SnapInfo__05-14-2010_15.33 .41.sml`

Variable	Description

`$SqlSnapshot`	Expands to the name of an SQL Server database Snapshot copy. This argument is used for backup and verification operations.
	Examples:
	`sqlsnap__winsrvr2__01-31-2014_15.03.09`
	`sqlsnap__winsrvr2__recent`
	The number of database Snapshot copies in a SnapManager backup set depends on the number of volumes used to store the databases included in the backup.
	For restore and clone operations, this argument specifies the name of the Snapshot copy to be restored.
	Example:
	`sqlsnap__winsrvr2__01-31-2014_15.03.09`
	`sqlsnap__winsrvr2__recent`

Several parameters, like the `$SnapInfoPath` and `$LogBackupFile` variables, are enclosed within double quotes by default so that the actual path name can contain spaces without affecting the script invocation on the Windows command line. If you do not want the double quotation marks to appear in your command line, remove them from the Command Arguments field in the Run Commands dialog box.

The following postcommand arguments apply only to restore and clone operations:

Variable	Description
`$StandbyFile`	This is the full file system path of the SQL standby file used on a restore. This file path is calculated during the restore-clone operation as a temporary file when incomplete transactions are removed from the database and stored in the file for later use. The user requests to generate a standby (or *undo*) file in a certain directory, but the full file name path actually used is not known until the restore or clone operation is launched. This happens when several databases are restored at the same time to the same LUNs. By default, this is created in the `snap-info` directory. Example: `U:\SMSQL_SnapInfo\VDISK__E\UNDO_SECLOCSYS_db5.dat`
`$TargetDatabase`	Specifies the destination name of the database to restore. Example: `DatabaseAccountingRestoredCopy`
`$TargetServerInstance`	Specifies the destination SQL Server instance to be used. Example: `ACMESERVER2\SQLINSTANCECOPY`
`$TargetDatabaseFile`	Specifies the target file system database path to be used. Example: `Z:\MNT\NETAPP1\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data\DatabaseAccounting.mdf`

Command arguments that are operation-specific

Each SnapManager operation that supports the Run Commands feature parses only the variables that apply to the operation as you have specified it.

The following table shows which of the command variables are available to the Run Commands feature, depending on which SnapManager operation is used to invoke the feature, and in what context those variables are parsed:

Variable	SnapManager operation used to invoke the Run Commands feature				
	Full backup	**Transaction log backup**	**Verification of full backup**	**Restore**	**Clone**
`$Database`	Parsed	Parsed	Parsed	Parsed	Parsed
`$InfoSnapshot`	Parsed	Parsed	Not parsed	Not parsed	Not parsed
`$LogBackupFile`	Parsed	Parsed	Not parsed	Not parsed	Not parsed
`$ServerInstance`	Parsed	Parsed	Parsed	Parsed	Parsed
`$OperationStatus`	Parsed	Parsed	Parsed	Parsed	Parsed
`$PreCommandStatus`	Parsed	Parsed	Parsed	Parsed	Parsed
`$SnapInfoName`	Parsed	Parsed	Parsed	Not parsed	Not parsed
`$SnapInfoPath`	Parsed	Parsed	Parsed	Not parsed	Not parsed
`$SqlSnapshot`	Parsed	Not parsed	Parsed	Parsed	Parsed
`$StandbyFile`	Not parsed	Not parsed	Not parsed	Parsed	Parsed
`$TargetDatabase`	Not parsed	Not parsed	Not parsed	Parsed	Parsed
`$TargetDatabaseFile`	Not parsed	Not parsed	Not parsed	Parsed	Parsed
`$TargetServerInstance`	Not parsed	Not parsed	Not parsed	Parsed	Parsed

Enabling SnapManager to allow databases on any LUN or VMDK configuration

By default, you can store files belonging to an individual database across two or more LUNs or VMDKs only if those LUNs or VMDKs are not used for storing database files that belong to other databases. If necessary for your configuration, you can remove this restriction by enabling the unrestricted database layout option.

About this task

The unrestricted database layout option removes layout restrictions imposed during configuration, enabling you to place your databases in any LUN or VMDK configuration. For example, you can

spread a database's files across two or more LUNs or VMDKs, even if those LUNs or VMDKs are used for storing database files that belong to other databases.

You cannot disable the unrestricted database layout option after you enable it.

Note the following about enabling this option when you have existing LUN or VMDK configurations:

- If you enable the unrestricted database layout option and then move databases to a new configuration, you should take a full backup of all databases before taking any log backups. Taking a full backup ensures that you can perform up-to-the-minute restore operations.
- If you enable the unrestricted database layout option and you have an existing LUN that contains multiple databases, you can continue to restore and clone the entire LUN, as long as the LUN contains the *entire* contents of multiple databases.
  For example, you can no longer restore or clone at the LUN level if you add a database to the existing LUN by spreading its files across the existing LUN and other LUNs. The entire database must reside on the LUN.

Steps

1. From the **Actions** pane, click **Configuration Wizard Option Settings**.
2. Select **Enable unrestricted database layout**.
3. Click **OK** to close the dialog box.

Viewing fractional space reservation status

Viewing the fractional space reservation status shows you the current space consumption in storage system volumes containing LUNs that store SQL Server data or SnapInfo directories.

Steps

1. In the **Actions** pane, click **Fractional Space Reservation Settings**.
2. Note the space consumption for each LUN that stores SQL Server data or SnapInfo directories.

   The following columns display SnapManager configuration information:

   Drive Letter or MountPoint

   A SnapManager configuration setting for the LUN. The drive letter or NTFS mount point on which the LUN is mounted.

   Fractional Overwrite Reserve (%)

   The percent of space reserved for overwrites on the storage system volume that contains this LUN. Expressed as a percentage of the total size of all space-reserved LUNs in the volume. If Fractional Overwrite Reserve (%) is 100, the LUN is contained in a fully space-reserved volume rather than a fractionally space-reserved volume.

Backup AutoDelete Trigger (%)

A SnapManager fractional space reservation policy setting for the storage system volume that contains the LUN. The percentage of overwrite reserve utilization that triggers automatic deletion of SQL Server backup sets.

Disable Database Trigger (%)

A SnapManager fractional space reservation policy setting for the storage system volume that contains the LUN. The percentage of overwrite reserve utilization that triggers automatic disabling of SQL Server databases.

The following columns display the fractional overwrite reserve settings and status:

Storage System Snapshot AutoDelete

For the storage system volume that contains this LUN, the state of the Data ONTAP Snapshot copy automatic deletion feature: enabled or disabled. If this LUN stores SQL Server data files and is contained in a storage system volume for which the Data ONTAP Snapshot copy automatic deletion feature is enabled, disable this feature on that volume or ensure that it is configured so that it does not delete SnapManager backup set components.

Used Overwrite Reserve

For the storage system volume that contains this LUN, the amount of overwrite reserve *in use*. Expressed in two ways: as a percentage of the total size of all space-reserved LUNs in the volume and in megabytes.

Available Reserve (MB)

For the storage system volume that contains this LUN, the amount of overwrite reserve *available*.

3. If “Enabled” appears in the **Storage System Snapshot AutoDelete** column, investigate the cause and take preventive action, if necessary.

   Attention: If AutoDelete is enabled, the LUN is contained in a FlexVol volume that has overwrite reserve set to less than 100 percent and that also has the Data ONTAP automatic Snapshot copy deletion feature enabled and configured to trigger when the overwrite reserve is nearly full. If SQL Server data or SnapManager SnapInfo directories are stored on LUNs contained in a volume with these characteristics, the Data ONTAP Snapshot copy automatic deletion policy might delete SQL Server backup set components.

   Take one of the following actions on the volume:

   - Disable the Data ONTAP Snapshot copy automatic delete feature.
   - Ensure that the Data ONTAP Snapshot copy automatic delete feature is configured in such a way that it does not delete SQL Server backup set components.

   The SnapManager fractional space reservation policy includes a separate, SQL Server-aware automatic deletion feature. The SnapManager automatic deletion feature can be used in place of or in conjunction with the Data ONTAP automatic deletion feature; you can also select to disable the SnapManager automatic deletion feature.

Configuring fractional space reservation policies

In the Fractional Space Reservation Settings dialog box, you can use the Policy Settings tab to view or customize the default policy and to configure custom policies for individual fractional space-reserved LUNs.

About this task

SnapManager automatically applies the default policy to every storage system volume containing fractional space-reserved LUNs that store SQL Server database files or SnapInfo directories. This policy ensures that your storage is protected from an out-of-space condition, without requiring you to explicitly enable or configure any fractional space reservation policies.

Steps

1. Select **Fractional Space Reservation Settings** in the SnapManager **Actions** pane.

   The Fractional Space Reservation Settings window is displayed.

2. Click **Policy Settings**.

3. In the left navigation tree, select the scope of the policy you want to view or change in the main panel on the right side of the tab:

If you want to view or change...	Then do this...
The default policy	In the navigation tree, select **Default Policy**.
A volume-specific policy	In the navigation tree, select the storage system, and then select the volume.

4. Enable or disable fractional space reservation monitoring.

5. Disable or configure automatic deletion of SQL Server backup Snapshot copies.

   Use the Automatic deletion of backups panel to disable, enable, or configure automatic deletion of SQL Server backup Snapshot copies in fractional space-reserved LUNs on the volume.

   Although automatic deletion of SQL Server backup Snapshot copies does not necessarily prevent an out-of-space condition on the volume, it is a best practice to enable this feature for every volume that contains fractional space-reserved LUNs that store SQL Server data.

6. In the “Trigger point for overwrite reserve utilization” field, enter the level of overwrite reserve utilization (in percentage of total reserve) that you want to trigger deletion of SQL Server backup Snapshot copies.

   Note: The value must be a non-negative integer that is less than the “Trigger point for overwrite reserve utilization” value in the Automatic dismount of databases panel.

7. In the **Number of most recent backups to retain** field, enter the number of backups to be retained if automatic backup set deletion is triggered.

   Note: The value must be an integer from 1 through 255 and should be based on the backup creation and verification schedule.

8. Use the **Automatically dismount databases** panel to configure automatic dismounting of SQL Server databases in fractional space-reserved LUNs on the volume.

   Because automatic deletion of SQL Server backup Snapshot copies does not necessarily prevent an out-of-space condition on the volume, SnapManager does not allow you to disable the dismounting of databases for any fractional space reservation policy.

   In the Trigger point for overwrite reserve utilization field, enter the level of overwrite reserve utilization (in percentage of total reserve) that you want to trigger dismounting of SQL Server databases. The value must be an integer from 0 through 99.

   If Snapshot copy automatic deletion is enabled, SnapManager requires that this threshold be set to a higher level than the threshold that triggers automatic Snapshot copy deletion. This ensures that Snapshot copy automatic deletion is triggered first.

9. Click **OK** to apply the changes to the default or volume-specific policy.

Advanced administration

For very large implementations, you need to be aware of SnapManager's configuration limits. You also should be aware of requirements for advanced configurations.

Maximum configurations supported by SnapManager

SnapManager supports a maximum number of SQL Server instances, LUNs and VMDKs per host, databases, file groups, and storage system volumes. You should review these limits to ensure you have a supported configuration.

Configuration type	Maximum
SQL Server instances per Windows cluster or SQL Server host: • Windows cluster • Stand-alone Windows host	 25 50
LUNs per SQL Server host	165
VMDKs per SQL Server host	56
Databases per LUN or VMDK	500
Databases per storage system volume	500
Databases across all SQL Server instances in a stand-alone SQL Server	5000
Databases across all SQL Server Failover Cluster Instances in a Windows Server Failover Cluster	2500
File groups per database	5000
Storage system volumes that can be used to store the following: • A single database • LUNs connected to an individual SQL Server computer • VMDKs connected to an individual SQL Server computer	 30 165 56

Service account requirements for archiving backup sets with SnapVault (7-Mode environments only)

To archive backup sets with SnapVault on Data ONTAP systems operating in 7-Mode, the SnapManager service account should be the same account you used to configure SnapDrive access to the DataFabric Manager server. If you cannot use the same account, the SnapManager service account needs specific permissions on the server.

On the DataFabric Manager server, you can assign permissions to the SnapManager service account using one of the following methods:

If you want to...	**Then...**
Assign specific permissions	Assign the SnapManager service account a role on the DataFabric Manager server with the following capabilities: • DFM.DataBase.Read Global • DFM.DataSet.Write Global • DFM.Policy.Read Global • DFM.BackupManager.Backup Global • DFM.BackupManager.Read Global • DFM.BackupManager.Restore Global You can use the `dfm role create`, `dfm role add`, and `dfm user add` commands to create the role, add the capabilities, and create the user.

Assign full-control permissions	Assign the SnapManager service account full-control rights on the DataFabric Manager server as shown in the following examples:
	• On Windows:
	`dfm user add -r GlobalFullControl MyDomain\snapuser`
	• On UNIX:
	`dfm user add -r GlobalFullControl MyDomain\\snapuser`

SnapManager disk requirements in a Windows cluster using LUNs

In a Windows clustered environment, SnapManager disk requirements vary, depending on the cluster configuration.

SnapManager supports multiple-instance clusters if the following additional system requirements are met:

- Each instance must have its own LUNs that cannot be used by other instances.
- Each instance must be created in its own *cluster group*.
- All LUNs assigned to a specified instance must be in the cluster group for that instance and in the SQL Server list of dependencies.

Single-instance cluster example

In an active/passive two-node configuration, there are two clustered nodes and one SQL Server instance. If the active node (the node running SQL Server) fails, the cluster transfers the SQL Server instance to the other (previously passive) node, which then becomes the active node and takes over the LUNs.

For a single-instance SQL Server cluster, if your SQL Server data is on a shared resource, your disk requirements are the same as for a stand-alone SQL Server system. A LUN is added for the quorum disk. A minimum of three LUNs are required:

- One LUN for the databases
- One LUN for the SnapInfo directory
- One LUN if a shared quorum disk is used

Multiple-instance cluster example

In an active/active two-node configuration, there are two clustered nodes and an SQL Server instance running on each node. If one node fails, the other node takes over the SQL Server instance running on the failed node. Because both nodes must be able to run an active SQL Server instance, each node requires its own disks, as if it were a self-contained, stand-alone system.

In addition, one extra LUN is needed for the quorum disk, if a shared quorum disk is used. Whether you use a hard disk or a LUN as the quorum disk, each configuration requires a minimum of five disks used for the following purposes:

- For node 1
  - One LUN to store the SQL Server databases

  - One LUN to store the SnapInfo directory
- For node 2
  - One LUN to store the SQL Server databases
  - One LUN to store the SnapInfo directory
  - One LUN or hard disk to be used as the quorum disk

Each node must be able to own all clustered disk resources in a cluster at any time.

Upgrading SnapManager

When a new version of SnapManager becomes available, you can upgrade your existing installation using either an interactive wizard or a command.

Before you begin

- Your host and storage systems must meet the minimum requirements for the version of SnapManager to which you are upgrading.
- You should have backed up your SQL databases using SnapManager.

About this task

You do not need to stop SQL Server instances before or during the upgrade process.

Upgrading SnapManager interactively

You can use the SnapManager installation wizard to interactively upgrade SnapManager.

Steps

1. Download the software from the NetApp Support Site.

   NetApp Downloads: Software

2. Exit SnapManager, if you have not already done so.
3. Double-click the downloaded .exe file.
4. Complete the pages in the SnapManager installation wizard to upgrade SnapManager.

Upgrading SnapManager from a command line

You can quickly run the SnapManager upgrade program unattended, in silent mode, from the Windows command line.

Steps

1. Download the software from the NetApp Support Site.

   NetApp Downloads: Software

2. Exit SnapManager, if you have not already done so.

3. From a Windows command prompt, change to the location where you downloaded the product installer.

4. Enter the following command at the command prompt:

   ***installer*.exe /s /v"/qn REINSTALLMODE=vomus REINSTALL=ALL SILENT_MODE=1 SVCUSERNAME=*Domain\UserName* SVCUSERPASSWORD=*Password* SVCCONFIRMUSERPASSWORD=*Password* [/L*V *DirPath\LogFileName*]"**

   Enter the following for each variable:

Variable	Description
installer	The name of the .exe file.
Domain\UserName	The user account that Windows uses to run SnapManager. This SnapManager service account must have specific permissions on the Windows host and the SQL Server. For details, see the *Installation and Setup Guide*.
Password	The password for the specified user account. Leave the password blank if you entered a group Managed Service Account in the **Account** field.
DirPath\LogFileName	The location and name of a log file, which is useful for troubleshooting. The asterisk (*) specifies that all installation information (such as status messages, non-fatal warnings, and error messages) should be logged.

Example

```
"SMSQL7.2_x64.exe" /s /v"/qn REINSTALLMODE=vomus
REINSTALL=ALL SILENT_MODE=1 SVCUSERNAME=MKTG2\Administrator
SVCUSERPASSWORD=password SVCCONFIRMUSERPASSWORD=password /L*V C:
\SMSQL_upgrade.log"
```

Repairing, reinstalling, and uninstalling SnapManager

You can repair, reinstall, or uninstall SnapManager as needed.

Repairing SnapManager

In many cases, you can correct stability issues with SnapManager by repairing the software. Repairing the software fixes missing or corrupt files, shortcuts, and registry entries.

Steps

1. In **Control Panel**, navigate to your installed programs.
2. Select **SnapManager for Microsoft SQL Server**.
3. Click **Repair**.

Result

Windows configures SnapManager for Microsoft SQL Server.

Reinstalling SnapManager

You might reinstall SnapManager if you want to install an older version of it or if you ran a repair operation that did not resolve a stability issue within the software.

About this task

You do not need to stop SQL Server instances before or during the SnapManager software reinstallation process.

Steps

1. Record the locations of existing SnapInfo directories.
2. Uninstall SnapManager.
3. Reinstall SnapManager.
4. Configure SnapManager with the same SnapInfo directory locations that were used by SnapManager before the reinstallation.

If you configure SnapManager with *different* SnapInfo directory locations than were used previously, then SnapManager no longer has records of any backups taken before you reinstalled it.

Uninstalling SnapManager

You can uninstall SnapManager if you no longer need the software or if you need to troubleshoot issues.

About this task

In a cluster configuration, be sure to uninstall SnapManager from all nodes in the cluster.

Steps

1. If SnapManager is running, stop it.
2. Uninstall SnapManager by performing an interactive uninstall or silent uninstall:

If you want to use...	Do this...
An interactive uninstall	Use the Control Panel to uninstall SnapManager for Microsoft SQL Server.
A silent uninstall	Enter the following command, either directly at a command line or through a script: ***file_name* /v"REMOVE=ALL [REMOVEREPORTFOLDER=1] [/L*V *DirPath\LogFileName*] /qb"** • `REMOVEREPORTFOLDER=1` removes the SnapManager Report folder. • `/L*V DirPath\LogFileName` writes detailed information about the uninstallation to the specified directory and log file. Example: `C:\NetApp\downloads\SMSQL7.1_x64.exe /v"REMOVE=ALL REMOVEREPORTFOLDER=1 /qb"`

SnapManager cmdlet guidelines

SnapManager includes a set of Windows PowerShell cmdlets that you can use to run scripts instead of using the SnapManager console. You should review a few guidelines before you use the cmdlets.

Location of the SnapManager PowerShell

A shortcut titled **SnapManager for SQL Server PowerShell** is available from the Windows Start menu.

If the execution policies in your system are restricted, you might be unable to load the PowerShell snap-in. To check and reset the execution policies on your system, follow these steps:

1. Enter the `get-executionpolicy` command in PowerShell.
2. If the policy displayed is “Allsigned” or “Restricted,” enter any of the following commands: `set-executionpolicy unrestricted` or `set-executionpolicy remotesigned`

Tips for using the cmdlets

- All parameters and options are not case-sensitive.
  For example, if you use the option `-Daily`, it achieves the same results if you enter `-daily`.
- Some of the options must be invoked in a particular order.
  For best results, use the order specified in the syntax for all options.
- When a parameter value string contains spaces, enclose it in double quotes.
  For example, you should use **`First Backup Set`** rather than First Backup Set.
- Press Ctrl-D to cancel a running operation.
  Closing the PowerShell window does not cancel the running operation.

PowerShell syntax for backup operations performed in the GUI

When you use the SnapManager GUI to back up a database, SnapManager logs the PowerShell syntax to the Backup report. You might use this information to create scripts or troubleshoot issues.

Common parameters

The following parameters are common to each cmdlet:

Debug (-db)

Displays the debug information for the cmdlet used.

ErrorAction Action Preference (-ea)

Determines what to do in case of an error. Scripting blocks use this parameter. The following examples explain the use of this parameter:

- `SilentlyContinue`: Continues without printing.
- `Continue`: Prints and then continue.
  This is the default setting.
- `Stop`: Halts the command or script.
- `Inquire`: Asks the user what to do.

ErrorVariable (-ev)

Displays the error data in the specified variable.

OutVariable (-ov)

Displays the output data string.

OutBuffer (-ob)

Displays the output buffer.

Whatif

Provides a preview of an operation.

Confirm

Prompts you for confirmation before the actual deletion operation starts.

Verbose (-vb)

Displays the report content for backup, restore, configuration, and verification options.

clone-backup

Name

clone-backup

Synopsis

Use this cmdlet to clone databases from an existing backup or archive using the SnapManager SQL Server PowerShell command-line interface. You can also use this cmdlet to add (by cloning) a database to an Availability Group.

Syntax

```
clone-backup [-Server <String>] [-UserName <String>] [-Password <String>]
[-ServerInstance <String[]>] -Database <String[]> [-Backup <String>] [-
RestoreLastBackup <Int32>] [-TransLogsToApply <Int32[]>] [-ForceRestore
```

```
[<Boolean>]] [-ClusterAware] [-TargetDatabase <String[]>] [-
TargetServerInstance <String[]>] [-TargetServerMountPointDir <String>] [-
PointInTime <String[]>] [-SnapInfoDirectory <String>] [-MarkName
<String[]>] [-MarkTime <String[]>] [-RestoreBeforeMark [<Boolean>]] [-
RecoverDatabase <Boolean[]>] [-StandbyPath <String>] [-apicontext] [-
RestoreArchivedBackup] [-SnapVaultSecondary] [-CloneOnMirrorDestination] [-
ChangeClonePath] [-CloneMirrorDestVolumes <String[]>] [-PreCommand] [-
PreCommandPath <String>] [-PreCommandArguments <String>] [-PreCommandHost
<String>] [-PreCommandErrors <EnumHandleCmdError[]>] [-PostCommand] [-
PostCommandPath <String>] [-PostCommandArguments <String>] [-
PostCommandHost <String>] [-PostCommandErrors <EnumHandleCmdError[]>] [-
AvailabilityGroup] [-IgnoreRepLogs] [-WhatIf] [-Confirm]
[<CommonParameters>]
```

Description

You can use this cmdlet to clone a live database or a database that is already backed up in a backup set. This cmdlet restores the database from the existing backup set, to clone the database to an alternate temporary writable LUN location, or to an Availability Group for further use.

You can also implement these options with the SnapManager user interface.

Parameters

-Server <String> - Short Form: -svr

This parameter denotes the name of the host SQL server on which the SQL server instances reside. SnapManager takes the local computer name as the default server name.

Using this parameter, you can also specify a particular SQL server instance. The following examples illustrate the usage:

```
-svr win-225-161
```

```
-svr sql1
```

-UserName <String> - Short Form: -usr

This parameter denotes the SQL Server account name. If the login name is not specified, SnapManager uses Windows NT Authentication.

-Password <String> - Short Form: -pwd

This parameter is the SQL Server account password. SnapManager ignores this parameter if the parameter -UserName is not specified.

-ServerInstance <String[]> - Short Form: -inst

This parameter specifies the SQL Server instance where the database is backed up originally. SnapManager takes the local computer name as the default server instance.

If multiple databases reside on the same LUN but are owned by different SQL server instances when you backed them up originally, use the following format:

```
-Inst "SQLServerInstance1","SQLServerInstance2"
```

The first database specified in the -Database parameter refers the first server instance in the -ServerInstance parameter, the second database in the -Database parameter refers to the second server instance in the -ServerInstance parameter and so on.

-Database <String[]> - Short Form: -d

Use this option to specify the databases that need to be cloned. Use a comma-separated list of strings:

```
-d Database 1, Database 2, Database 3, Database 4,....
```

Multiple database names should be specified only if those databases share a single LUN or multiple LUNs together. For a multiple database restore, all the selected databases should be present in the selected Snapshot copy.

You cannot restore a database with a new name if you specify multiple databases. If you want to restore with a new name, restore those databases one by one. In case of restore to alternate location, specify only one database name.

-Backup <String> - Short Form: -bkup

Use this option to specify the name of the backup set. This is a mandatory parameter. The following example illustrates the usage:

-bkup sqlsnap__SYMNASQLDEV170_04-11-2007_15.22.27

-RestoreLastBackup <Int32> - Short Form: -lastBkup

Use this parameter to restore backups without specifying the name. If you try to use the `Backup` and `RestoreLastBackup` parameters together, SnapManager ignores the `RestoreLastBackup` parameter and uses the `Backup` parameter during restore operation. A typical usage example of the `restorelastbackup` parameter is as follows:

```
restore-backup -restorelastbackup 1 -backup <backup name>
```

> **Note:** If the value for RestoreLastBackup parameter is 0, SnapManager restores the latest backup. If the value is 1, SnapManager restores second-to-latest backup and so on.

-TransLogsToApply <Int32[]> - Short Form: -translogs

This parameter specifies the list of transactions logs that need to be applied. SnapManager applies all transaction logs of the databases specified in the -Database parameter by default. You can specify the number of transaction logs to be applied for every database mentioned in the -Database parameter. The list of number of transaction logs that have to applied has to be listed in the same sequence as the databases listed in the -Database parameter. For example,

```
restore-backup -svr MACHINE1\INST1 -database db1,db2 -TransLogsToApply 3,7
```

-ForceRestore [<Boolean>] - Short Form: -force

Use this parameter to force the restore of a database based on its state. SnapManager sets its value to "true" by default.

-ClusterAware - Short Form: -cl

Use this parameter to specify that the cmdlet runs solely on the active node in a cluster environment.

-TargetDatabase <String[]> - Short Form: -tgDb

Use this parameter to restore a database with a new name. The following example illustrates the usage:

```
-tgDb "NewDatabaseName1"," NewDatabaseName2"," NewDatabaseName3"
```

The parameter defines the new database name to which the original database is restored. The old database name is defined at the same position in the -Database parameter.

If no new database name is given, the database is restored to the original database name the database had during backup. If this original name already exists, the name is modified to: originalDbName__clone, or originalDbName__mount.

-TargetServerInstance <String[]> - Short Form: -tgInst

This parameter specifies the name of the new SQL server if you want to restore the database to a new SQL Server. SnapManager takes the source SQL server instance as the default.

-TargetServerMountPointDir <String> - Short Form: -tgmpdir

Use this parameter to specify the mount point path or directory of the target server instance in which the backups are cloned or mounted.

-PointInTime <String[]> - Short Form: -pit

Use this switch to restore databases until a specific point in time. The format for the point-in-time string is yyyy-mm-ddThh:mm:ss, with time specified in a 24-hour format.

In case of multiple databases you should specify the point-in-time values for every database separated by a comma. The number of values after the parameter name should equal the number of databases selected. The first value will be applied to the first database specified after the -Database parameter, the second value to the second database, and so on. The following example illustrates the usage:

```
-pit 2008-10-22T11:50:00, 2008-11-25T22:50:00
```

> **Note:** The parameter correspondence is one-to-one, that is, the first point-in-time parameter value specified after the parameter -pit is applied to the first database specified in the parameter -Database and the second point-in-time parameter value to second database and so on. The values should conform to the required PointInTime regular expression.

-SnapInfoDirectory <String> - Short Form: -snapinfo

Use this parameter to specify the SnapInfo directory path of the archived backup set.

-MarkName <String[]> - Short Form: -mark

This parameter indicates the marked transaction at which to stop the transaction log recovery.

-MarkTime <String[]> - Short Form: -mktm

This parameter specifies a unique timestamp to guarantee the uniqueness of the input restored mark.

-RestoreBeforeMark [<Boolean>] - Short Form: -beforemk

This true or false value indicates whether the specified marked transaction log should be included in the restore.

-RecoverDatabase <Boolean[]> - Short Form: -recoverdb

This parameter indicates whether the database fully recovered or left in a partially recovered state after the cmdlet finishes, to facilitate future SQL transaction log restores. This is an array of booleans, so it must match the same number of elements of the -database array. If the it does not match the number of elements of the -database array, an error is given. This defaults to $true for all databases unless the -standbyPath is given, in which case it defaults to $false for all databases.

-StandbyPath <String> - Short Form: -standby

This parameter indicates the path to the standby recovery file where incomplete transactions are stored after restoring a full database and its transaction logs. There is no default if you specify this parameter. The path must be to the standby directory if more than one database shares a LUN. If the database is on a dedicated LUN, then it must be a specific file. If the -standbypath parameter is given, the -RecoveryDatabase given must be -RecoverDatabase $False, otherwise it defaults to $false for all databases if no _RecoverDatabase parameter is specified.

-apicontext - Short form: none

Use this parameter when calling the cmdlet as an API call.

-RestoreArchivedBackup - Short Form: -rstarchbkup

Use this parameter to specify using remote backup to clone the database.

-SnapVaultSecondary - Short Form: -vaultsec

This optional parameter identifies the backup vault from which you want to clone a database. If you do not specify this parameter, SnapManager chooses one of the backup vaults. You use this parameter in conjunction with the `-RestoreArchivedBackup` parameter. If you specify this parameter with the `-AvailabilityGroup` parameter, then the Availability Group databases must be spread across the same volumes. Otherwise, do not specify this parameter and SnapManager will choose one of the backup vaults. This parameter applies to clustered Data ONTAP only.

The syntax for this parameter is as follows:

```
-SnapVaultSecondary n, Vserver:volume
```

Where n is the number of Storage Virtual Machine (SVM, formerly known as Vserver):volume pairs.

Example: `-SnapVaultSecondary 3, Vserver1:volume1, Vserver2:volume2, Vserver3:volume3`

-CloneOnMirrorDestination - Short Form: -cloneonmir

This parameter indicates to clone a database based on the Snapshot copy on the SnapMirror destination volume. Ensure that the SnapMirror relationship exists and SnapMirror was updated when using this option.

-ChangeClonePath (Boolean Parameter) - Short Form: -chgpath

Use this parameter to change clone database paths based on the new database clone name.

-CloneMirrorDestVolumes <String[]> - Short Form: -clonemir

Use this parameter to specify cloning using the Snapshot copy on the SnapMirror destination volume.

-PreCommand <String> - Short Form: -precmd

This parameter indicates to run a command before the current operation.

> **Note:** You cannot have more than one space between items that may be parsed in this parameter's value.

-PreCommandPath <String> - Short Form: -precmdpath

This parameter specifies the operating system path to the command to be run before the SnapManager operation starts.

-PreCommandArguments <String> - Short Form: -precmdargs

This parameter contains a list of strings of SnapManager operation-specific information or user-defined arguments to be passed to the program or script. The default is to pass no parameters to the script. If the parameter contains white spaces (tabs or spaces) you need to enclose it in double quotes. This parameter is processed only if the parameters -PreCommand and -PreCommandPath are specified.

-PreCommandHost <String> - Short Form: -precmdhost

This parameter specifies the host machine name on which the command is run before the operation starts. The default is to run on the current machine. This parameter is considered only if the parameters -PreCommand and -PreCommandPath are specified.

-PreCommandErrors <EnumHandleCmdError[]> - Short Form: -precmnderrors

This parameter specifies how to handle errors on the pre-command. The ContinueOnError value (the default) indicates that the SnapManager operation executes even if an error is detected during the pre-command launch. The StopOnPreCmdError value indicates that if a pre-command script gets an error, the remaining SMSQL operation is not attempted. This parameter is considered only if the parameters -PreCommand and -PreCommandPath are specified.

-PostCommand - Short Form: -postcmd

This parameter indicates to run a command after the current operation is complete.

> **Note:** You cannot have more than one space between items that may be parsed in this parameter's value.

-PostCommandPath <String> - Short Form: -postcmdpath

Use this parameter to specify the operating system path to the command to be run after the SnapManager operation starts.

-PostCommandArguments <String> - Short Form: -postcmdargs

This parameter contains a list of strings of SnapManager operation-specific information or user-defined arguments to be passed to the program or script. The default is to pass no parameters to the script. If the parameter contains white spaces (tabs or spaces) you enclose it in double quotes. This parameter is processed only if the parameters -PostCommand and -PostCommandPath are specified.

-PostCommandHost <String> - Short Form: -postcmdhost

This parameter specifies the host machine name on which the command is run after the operation is complete. The default is to run on the current machine. This parameter is considered only if the parameters -PostCommand and -PostCommandPath are specified.

-PostCommandErrors <EnumHandleCmdError[]> - Short Form: -postcmderrors

This parameter specifies how to handle SMSQL operation errors on the post-command run. The ContinueOnError value (the default) indicates that the SMSQL operation executes even if an error is detected during the post-command launch. The StopOnPostCmdError value indicates that if a post-command script gets an error, the remaining SMSQL operation is not attempted. This parameter is considered only if the parameters -PostCommand and -PostCommandPath are specified.

-AvailabilityGroup <String> - Short Form: -ag

Use this parameter to reseed databases belonging to the given Availability group.

-IgnoreRepLogs: - Short Form: -nosharelogs

Use this parameter to ignore the transaction logbackups from SnapManager Repository Share.

-WhatIf - Short form: -wi

This parameter gives you a preview of an operation.

-Confirm - Short form: -cf

This parameter prompts you for confirmation before the actual operation starts.

<CommonParameters>

This cmdlet supports the common parameters: Verbose, Debug, ErrorAction, ErrorVariable, WarningAction, WarningVariable, OutBuffer and OutVariable. For more information, see about_CommonParameters (http://go.microsoft.com/fwlink/?LinkID=113216).

Examples

Example 1: `clone-backup -Server win-225-165 -Database DB2 -Inst win-225-165 -Backup sqlsnap__win-225-165_09-06-2008_13.44.51`

This command creates a clone of the specified backup.

Example 2: `clone-backup -Server win-225-165 -Database DB2 -Inst win-225-165 -RestoreLastBackup 0`

This command restores the most recent clone that was created.

Example 3: `clone-backup -Server win-225-165 -Inst win-225-165 -AvailabilityGroup Ag1 -RestoreLastBackup 0`

This command restores the most recent clone of the Availability Group that was created.

Example 4: `clone-backup -svr win-225-165 -Database DB2 -Inst win-225-165 -Backup sqlsnap__win-225-165_09-06-2008_13.44.51 -RestoreArchivedBackup -vaultsec 2,vserver1:volume1,vserver2:volume2`

This command creates a clone of database DB2 from the specified SnapVault secondary volume.

clone-database

Name

clone-database

Synopsis

This cmdlet enables you to clone a live database or a database that is already backed up in a backup set using the SnapManager SQL Server PowerShell command-line interface.

Syntax

```
clone-database [-Server <String>] [-UserName <String>] [-Password <String>] [-LogBkup] [-Verify] [-VerifyServerInstance <String>] [-VerSvrLogin <String>] [-VerSvrPassword <String>] [-VerDestVolume] [-VerifyOnDestVolumes <String[]>] [-DBCCOption <EnumDbccOption[]>] [-CloneOnMirrorDestination] [-ChangeClonePath] [-Resynchronize] [-ForceTerminateConnection] [-ClusterAware] [-CloneMirrorDestVolumes <String[]>] [-VerifyDisable] [-UseMountPoint] [-MountPointDir <String>] [-UseDriveAvailable] [-RetainBackups <Int32>] [-RetainBackupDays <Single>] [-AttachDB] [-UpdateMirror] [-NoRetainUTM] [-ManagementGroup <String>] [-LogBkupOnly] [-BkupSIF] [-RetainSnapofSnapInfo <Int32>] [-RetainSnapofSnapInfoDays <Single>] [-TruncateSqlLog [<Boolean>]] [-TruncateLogs] [-PreCommand] [-PreCommandPath <String>] [-PreCommandArguments <String>] [-PreCommandHost <String>] [-PreCommandErrors <EnumHandleCmdError[]>] [-PostCommand] [-PostCommandPath <String>] [-PostCommandArguments <String>] [-PostCommandHost <String>] [-PostCommandErrors <EnumHandleCmdError[]>] [-RunDBCCAfter] [-RunDBCCBefore] [-GenericNaming] [-ArchiveBackup] [-VerifyArchiveBackup] [-ArchivedBackupRetention <String>] [-ServerInstance
```

```
<String[]>] -Database <String[]> [-TransLogsToApply <Int32[]>] [-
ForceRestore [<Boolean>]] [-TargetDatabase <String[]>] [-
TargetServerInstance <String[]>] [-TargetServerMountPointDir <String>] [-
MarkName <String[]>] [-MarkTime <String[]>] [-RestoreBeforeMark
[<Boolean>]] [-RecoverDatabase <Boolean[]>] [-StandbyPath <String>] [-
apicontext] [-RestoreArchivedBackup] [-RetainShareBackups] [-
RetainShareBackupDays] [-AvailabilityGroup] [-IgnoreRepLogs] [-WhatIf] [-
Confirm] [<CommonParameters>]
```

Description

This cmdlet enables you to clone a live database or a database that is already backed up in a backup set. It creates a backup set of the database and uses the backup set to clone the database. This cmdlet provides you various verification options, DBCC, recovery after restore, retaining backups, management groups and many other options.

You can also implement these options with the SnapManager user interface.

Parameters

-Server <String> - Short form: -svr

This parameter denotes the name of the host SQL Server on which the SQL Server instances reside. SnapManager takes the local computer name as the default server name.

Using this parameter, you can also specify a particular SQL server instance. The following examples illustrate the usage:

```
-svr win-225-161
```

```
-svr sql1
```

-Username <String> - Short form: -usr

This parameter denotes the SQL Server account name. If the login name is not specified, SnapManager uses Windows NT Authentication.

-Password <String> - Short form: -pwd

This parameter is the SQL Server account password. SnapManager ignores this parameter if the parameter -UserName is not specified.

-Logbkup - Short form: -lb

Use this option to specify that the transaction logs also need to be backed up after a full backup.

-Verify - Short form: -ver

Use this parameter if you wish to verify the backed up databases and logs.

-VerifyServerInstance <String> - Short form: -verInst

This parameter specifies the separate SQL server that is used to run the Database Consistency Check (DBCC) utility. If you have not specified the -verify parameter, SnapManager ignores this parameter.

The following example illustrates the usage:

```
-verInst win-225-161
```

Here the SQL server instance is the local or remote SQL server instance to verify on. SnapManager takes the configured SQL server instance that is used for verify in client configuration (registry) as the default SQL server instance.

-VerSvrLogin <String> - Short form: -verlogin

This parameter specifies that SQL Server authentication is used. If not specified, the default Windows NT Authentication mechanism is used.

-VerSvrPassword <String> - Short form: -verpwd

This parameter is used to input the verification server password. SnapManager ignores this parameter if the parameter -VerSvrLogin is not specified.

-VerDestVolume - Short form: -verdest

Use this parameter to verify the database on the SnapMirror destination volume. SnapManager sets it to "false" by default.

-VerifyOnDestVolumes <String[]> - Short form: -vermirror

Specify this parameter in order to override the default SnapMirror relationships. Enter the source and destination storage systems and volumes as a comma-separated list. SnapManager sets it to "false" by default.

-DBCCOption <EnumDbccOption[]> - Short form: -dbccopt

This parameter specifies options to the DBCC SQL command that are used to validate and verify the database that is being processed. When you use this parameter, you are explicitly requesting DBCC options, and the system does read the registry to determine the default DBCC options. The security access issues for the registry are bypassed when you use this cmdlet option. The parameter uses the following values:

NOOPTION

NOINDEX

ALL_ERRORMSGS

NO_INFOMSGS (default)

TABLOCK

PHYSICAL_ONLY (default)

For more information about these options, see your Microsoft SQL Server documentation.

-CloneOnMirrorDestination - Short form: -cloneonmir

Use this parameter to clone a database based on the Snapshot copy on the SnapMirror destination volume. Ensure that the SnapMirror relationship exists and SnapMirror was updated when using this option.

-ChangeClonePath - Short form: -chgpath

Use this parameter to change clone database paths based on the new database clone name.

-Resynchronize - Short form: -resync

Use this parameter to specify that the existing clone is refreshed with the live database.

-ForceTerminateConnection - Short form: -ftc

Use this parameter to specify that all the connections to the existing clone are terminated during clone resynchronize.

-ClusterAware - Short form: -cl

Use this parameter to specify that the cmdlet runs solely on the active node in a cluster environment.

-CloneMirrorDestVolumes <String[]> - Short form: -clonemir

Use this parameter to specify cloning using the Snapshot copy on the SnapMirror destination volume.

-VerifyDisable - Short form: -verDis

This parameter overrides verification and can disable verification even if the database was not verified after backup.

-UseMountPoint - Short form: -mp

This parameter specifies that the Snapshot copy must be mounted to an NTFS directory.

During a SnapManager verification operation, Snapshot copies are mounted to the default NTFS directory for database verification. The option is effective when there are no available drive letters to mount the Snapshot copies. It overrides pre-configured SnapManager verification settings.

-MountPointDir <String> - Short form: -mpdir

Use this parameter to specify the mount point directory on which a backup set will be mounted during database verification. Use this parameter with the parameter -UseMountPoint.

-UseDriveAvailable - Short form: -drvavail

Use this parameter to indicate that you should use available drive letter as mount point on which a backup set is mounted during database verification.

-RetainBackups <Int32> - Short form: -tgInst

Use this parameter to specify the number of backups to be retained after the delete operation.

-RetainBackupDays <Single> - Short form: -rtdays

Use this parameter to specify the number of days you want to retain the backups for. SnapManager deletes backups older than the specified number of days. The parameters RetainBackups and RetainBackupDays are mutually exclusive and cannot be specified together.

-AttachDB - Short form: -attdb

If the operation includes a database or transaction log verification, use this option when you want to specify that the databases are to be attached after the verification operation completes.

-UpdateMirror - Short form: -updmir

Use this option to update the SnapMirror destination after a backup or verification operation ends, if the operation uses backups that reside on volumes configured as SnapMirror sources.

-NoRetainUTM - Short form: -noutm

Use this option if you do not want to retain up-to-the-minute restore ability for older backups in other management groups.

-ManagementGroup <String> - Short form: -mgmt

This parameter denotes the backup or verify operation that SnapManager performs on daily, or weekly, or standard basis. The default management group is standard.

-LogBkupOnly - Short form: -lgbkonly

Use this option to back up your SQL Server transaction log files only. No full snapshot backup will be done.

-BkupSIF - Short form: -bksif

Use this option to create a Snapshot copy of the SnapInfo directory after the backup of the transaction log completes. The backup type should be a transaction log backup only.

-RetainSnapofSnapInfo <Int32> - Short form: -rtsifsnap

Use this option if you want to delete the oldest Snapshot copies in the SnapInfo directory, specified that the backup type is a transaction log backup only. It has an integer value. The following example illustrates the usage of this parameter: -rtsifsnap Number of SnapInfo Snapshots to keep

Note: This option is valid only if you specify the parameter - BkupSIF.

-RetainSnapofSnapInfoDays <Single> - Short form: -rtsifsnapdays

Use this parameter to delete SnapInfo Snapshot copies older than the specified number of days. This parameter is mutually exclusive with the parameter RetainSnapofSnapinfo and they cannot be specified together in the same cmdlet.

-TruncateSqlLog [<Boolean>] - Short form: -truncLog

This parameter specifies whether to truncate the SQL transaction logs. SQL transaction logs are truncated by default. Valid values are $true or $false. This parameter only works if -LogBkup or -LogBkupOnly are true.

-TruncateLogs - Short form: -trlog

This obsolete parameter (now replaced by TruncateSqlLog) specifies whether to truncate the SQL transaction logs. SQL transaction logs are not truncated by default. This parameter only works if -

LogBkup or -LogBkupOnly are true. In SMSQL 5.2 and later, if neither -TruncateLogs or -TruncateSqlLog is specified, the default behavior is to truncate the logs.

-PreCommand <String> - Short form: -precmd

This parameter indicates to run a command before the current operation.

> **Note:** You cannot have more than one space between items that may be parsed in this parameter's value.

-PreCommandPath <String> - Short form: -precmdpath

This parameter specifies the operating system path to the command to be run before the SnapManager operation starts.

-PreCommandArguments <String> - Short form: -precmdargs

This parameter contains a list of strings of SnapManager operation-specific information or user-defined arguments to be passed to the program or script. The default is to pass no parameters to the script. If the parameter contains white spaces (tabs or spaces) you need to enclose it in double quotes. This parameter is processed only if the parameters -PreCommand and -PreCommandPath are specified.

-PreCommandHost <String> - Short form: -precmdhost

This parameter specifies the host machine name on which the command is run before the operation starts. The default is to run on the current machine. This parameter is considered only if the parameters -PreCommand and -PreCommandPath are specified.

-PreCommandErrors <EnumHandleCmdError[]> - Short form: -precmnderrors

This parameter specifies how to handle errors on the pre-command. The ContinueOnError value (the default) indicates that the SMSQL operation executes even if an error is detected during the pre-command launch. The StopOnPreCmdError value indicates that if a pre-command script gets an error, the remaining SMSQL operation is not attempted. This parameter is considered only if the parameters -PreCommand and -PreCommandPath are specified.

-PostCommand - Short form: -postcmd

This parameter indicates to run a command after the current operation is complete.

> **Note:** You cannot have more than one space between items that may be parsed in this parameter's value.

-PostCommandPath <String> - Short form: -postcmdpath

This parameter specifies the operation system path for the command to be run after the SMSQL operation is complete.

-PostCommandArguments <String> - Short form: -postcmdargs

This parameter contains a list of strings of SnapManager operation-specific information or user-defined arguments to be passed to the program or script. The default is to pass no parameters to the

script. If the parameter contains white spaces (tabs or spaces) you enclose it in double quotes. This parameter is processed only if the parameters -PostCommand and -PostCommandPath are specified.

-PostCommandHost <String> - Short form: -postcmdhost

This parameter specifies the host machine name on which the command is run after the operation is complete. The default is to run on the current machine. This parameter is considered only if the parameters -PostCommand and -PostCommandPath are specified.

-PostCommandErrors <EnumHandleCmdError[]> - Short form: -postcmderrors

This parameter specifies how to handle SMSQL operation errors on the post-command run. The ContinueOnError value (the default) indicates that the SMSQL operation executes even if an error is detected during the post-command launch. The StopOnPostCmdError value indicates that if a post-command script gets an error, the remaining SMSQL operation is not attempted. This parameter is considered only if the parameters -PostCommand and -PostCommandPath are specified.

-RunDBCCAfter - Short form: -dbccaf

If the operation includes a database backup, use this parameter if you want to verify the live database after the backups are performed.

-RunDBCCBefore - Short form: -dbccbf

If the operation includes a database backup, use this parameter if you want to verify the live database before the backups are performed.

-GenericNaming - Short form: -gen

This parameter specifies that the backups must follow the Generic backup naming convention.

-ArchiveBackup - Short form: -arch

Use this parameter to archive database to a secondary storage system during the backup phase of the operation.

-VerifyArchiveBackup - Short form: -verarch

Use this parameter to verify database archived at the secondary storage system.

-ArchivedBackupRetention <String> - Short form: -archret

Use this parameter to specify whether you want to retain backups at the archived location on a daily, hourly, weekly, monthly or unlimited basis.

-ServerInstance <String[]> - Short form: -inst

This parameter specifies the SQL server instance where the database is backed up originally. SnapManager takes the local computer name as the default server instance.

You can specify multiple server instance names here as a comma-separated list. If multiple databases reside on the same LUN but are owned by different SQL server instances when you backed them up originally, use the following format:

```
-Inst "SQLServerInstance1","SQLServerInstance2"
```

The first database specified in the -Database parameter refers the first server instance in the -ServerInstance parameter, the second database in the -Database parameter refers to the second server instance in the -ServerInstance parameter and so on.

-Database <String[]> - Short form: -d

Use this option to specify the databases that need to be cloned. Use a comma-separated list of strings:

```
-d Database 1, Database 2, Database 3, Database 4,....
```

Multiple database names should be specified only if those databases share a single LUN or multiple LUNs together. For a multiple database restore, all the selected databases should be present in the selected Snapshot copy.

You cannot restore a database with a new name if you specify multiple databases. If you want to restore with a new name, restore those databases one by one. In case of restore to alternate location, specify only one database name.

-TransLogsToApply <Int32[]> - Short form: -translogs

This parameter specifies the count of transactions logs that need to be applied to each database restored. If the TransLogsToApply parameter is not given, then all transaction logs that apply to the full backup restored are applied by default (just as the GUI does). You can specify the number of transaction logs to be applied for every database mentioned in the -Database parameter. The list of number of transaction logs that are applied must be listed in the same sequence as the databases listed in the -Database parameter. For example:

-Database db1,db2

might correspond to:

-TransLogsToApply 1,8

which means 1 transaction log backup will be applied to db1, and 8 will be applied to db2.

-ForceRestore [<Boolean>] - Short form: -force

Use this parameter to force the restore of a database based on its state. SnapManager sets it's value to "true" by default.

-TargetDatabase <String[]> - Short form: -tgDb

Use this parameter to restore a database with a new name. The following example illustrates the usage:

```
-tgDb "NewDatabaseName1"," NewDatabaseName2"," NewDatabaseName3"
```

The parameter defines the new database name to which the original database is restored. The old database name is defined at the same position in the -Database parameter.

If no new database name is given, the database is restored to the original database name the database had during backup. If this original name already exists, the name is modified to: originalDbName__clone, or originalDbName__mount.

-TargetServerInstance <String[]> - Short form: -tgInst

This parameter specifies the name of the new SQL server if you want to restore the database to a new SQL server. SnapManager takes the source SQL server instance as the default.

-TargetServerMountPointDir <String> - Short form: -tgmpdir

Use this parameter to specify the mount point path or directory of the target server instance in which the databases are to be cloned or mounted.

-MarkName <String[]> - Short form: -mark

This parameter indicates the marked transaction at which to stop the transaction log recovery.

-MarkTime <String[]> - Short form: -mktm

This parameter specifies a unique timestamp to guarantee the uniqueness of the input restored mark.

-RestoreBeforeMark [<Boolean>] - Short form: -beforemk

This true or false value indicates whether the specified marked transaction log should be included in the restore.

-RecoverDatabase <Boolean[]> - Short form: -recoverdb

This parameter indicates whether the database fully recovered or left in a partially recovered state after the cmdlet finishes, to facilitate future SQL transaction log restores. This is an array of booleans, so it must match the same number of elements of the -database array. If the it does not match the number of elements of the -database array, an error is given. This defaults to $true for all databases unless the -standbyPath is given, in which case it defaults to $false for all databases.

-StandbyPath <String> - Short form: -standby

This parameter indicates the path to the standby recovery file where incomplete transactions are stored after restoring a full database and its transaction logs. There is no default if you specify this parameter. The path must be to the standby directory if more than one database shares a LUN. If the database is on a dedicated LUN, then it must be a specific file. If the -standbypath parameter is given, the -RecoveryDatabase given must be -RecoverDatabase $False, otherwise it defaults to $false for all databases if no -RecoverDatabase parameter is specified.

-apicontext - Short form: none

Use this parameter when calling the cmdlet as an API call.

-RestoreArchivedBackup - Short form: -rstarchbkup

Use this parameter to specify using remote backup to perform the clone operation.

-RetainShareBackups <Integer> - Short form: -rtsharebackups

Use this parameter to specify the number of log backups retained in the SnapManager for SQL repository share.

-RetainShareBackupDays <Integer> - Short form: -rtsharedays

Use this parameter to specify for how many days log backups are retained in the SnapManager Repository Share.

-AvailabilityGroup <String> - Short form: -ag

Use this parameter to reseed databases belonging to the given Availability group.

-IgnoreRepLogs: - Short form: -nosharelogs

Use this parameter to ignore the transaction logbackups from SnapManager Repository Share.

-WhatIf - Short form: -wi

This parameter gives you a preview of an operation.

-Confirm - Short form: -cf

This parameter prompts you for confirmation before the actual operation starts.

<CommonParameters>

This cmdlet supports the common parameters: Verbose, Debug, ErrorAction, ErrorVariable, WarningAction, WarningVariable, OutBuffer and OutVariable. For more information, see about_CommonParameters (http://go.microsoft.com/fwlink/?LinkID=113216).

Examples

Example 1: `clone-database -svr sql1 -Database "Db1"`

This command clones database Db1 located on SQL Server sql1.

Example 2: `clone-database -svr win-225-166 -Inst win-225-166 -Database dbtest1 -Verify -verinst win-225-166 -RecoverDatabase`

This example enables database cloning with a default name for a default instance.

Example 3: `clone-database -svr win-225-166 -Inst win-225-166 -Database dbtest1 -Verify -verinst win-225-166 -TargetDatabase dbtest1_Clone -RecoverDatabase`

This example enables database cloning with a new name for a default instance.

Example 4: `clone-database -svr win-225-166 -Inst win-225-166\Named -Database dbtest2 -Verify -verinst win-225-166 -RecoverDatabase`

This example enables database cloning with a default name for a named instance.

Example 5: `clone-database -svr win-225-166 -Inst win-225-166\Named -Database dbtest2 -Verify -verinst win-225-166 -TargetDatabase dbtest2_Clone -RecoverDatabase`

This example enables database cloning with a new name for a named instance.

Example 6: `clone-database -svr 'SNAPMGR-19' -inst 'SNAPMGR-19', 'SNAPMGR-19', 'SNAPMGR-19' -d 'DB3', 'DB4', 'DB5' -tgInst 'SNAPMGR-19' -tgDb 'DB3__Clone', 'DB4__Clone', 'DB5__Clone' -tgmpdir 'E:\Program Files`

```
\NetApp\SnapManager for SQL Server\SnapMgrMountPoint' -ClusterAware -
Resynchronize -ForceTerminateConnection -RetainBackups 3 -lb -mgmt standard
```

This example creates a new backup on database "DB3," "DB4," and "DB5" and refreshes the cloned databases on the active node.

Example 7: `clone-database -svr 'venudhar-2k8vm2' -inst 'venudhar-2k8vm2\heitz' -ag 'testag'`

This command clones all the databases belonging to the specified Availability group.

clone-replica

Name

clone-replica

Synopsis

Use this cmdlet to create an Availability Group replica by cloning existing Availability Group databases to a specified server, which then becomes a secondary.

Syntax

```
clone-replica [-Server <String>] [-UserName <String>] [-Password <String>]
[-LogBkup] [-Verify] [-VerifyServerInstance <String>] [-VerSvrLogin
<String>] [-VerSvrPassword <String>] [-VerDestVolume] [-VerifyOnDestVolumes
<String[]>] [-DBCCOption <EnumDbccOption[]>] [-CloneOnMirrorDestination] [-
ChangeClonePath] [-Resynchronize] [-ForceTerminateConnection] [-
ClusterAware] [-CloneMirrorDestVolumes <String[]>] [-VerifyDisable] [-
UseMountPoint] [-MountPointDir <String>] [-UseDriveAvailable] [-
RetainBackups <Int32>] [-RetainBackupDays <Single>] [-AttachDB] [-
UpdateMirror] [-NoRetainUTM] [-ManagementGroup <String>] [-LogBkupOnly] [-
BkupSIF] [-RetainSnapofSnapInfo <Int32>] [-RetainSnapofSnapInfoDays
<Single>] [-TruncateSqlLog [<Boolean>]] [-TruncateLogs] [-PreCommand] [-
PreCommandPath <String>] [-PreCommandArguments <String>] [-PreCommandHost
<String>] [-PreCommandErrors <EnumHandleCmdError[]>] [-PostCommand] [-
PostCommandPath <String>] [-PostCommandArguments <String>] [-
PostCommandHost <String>] [-PostCommandErrors <EnumHandleCmdError[]>] [-
RunDBCCAfter] [-RunDBCCBefore] [-GenericNaming] [-ArchiveBackup] [-
VerifyArchiveBackup] [-ArchivedBackupRetention <String>] [-ServerInstance
<String[]>] -Database <String[]> [-TransLogsToApply <Int32[]>] [-
ForceRestore [<Boolean>]] [-TargetDatabase <String[]>] [-
TargetServerInstance <String[]>] [-TargetServerMountPointDir <String>] [-
MarkName <String[]>] [-MarkTime <String[]>] [-RestoreBeforeMark
```

```
[<Boolean>]] [-RecoverDatabase <Boolean[]>] [-StandbyPath <String>] [-
apicontext] [-RestoreArchivedBackup] [-WhatIf] [-Confirm] -
AvailabilityGroup [-SynchronousCommit] [-FailoverMode] [-ReadableSecondary]
[<CommonParameters>]
```

Description

The cmdlet uses Snapshot technology to quickly replicate databases to a remote cluster SQL instance, and then groups them in an Availability Group. The replicated databases are associated with instances in the same cluster so that Availability Group failover can take place when required or requested.

An Availability Group supports up to three synchronous commit replicas and up to two automatic failover replicas.

You can also implement these options with the SnapManager user interface.

Parameters

-Server <String> - Short form: -svr

This parameter denotes the name of the host SQL Server on which the SQL Server instances reside. SnapManager takes the local computer name as the default server name.

Using this parameter, you can also specify a particular SQL server instance. The following examples illustrate the usage:

```
-svr win-225-161
```

```
-svr sql1
```

-Username <String> - Short form: -usr

This parameter denotes the SQL Server account name. If the login name is not specified, SnapManager uses Windows NT Authentication.

-Password <String> - Short form: -pwd

This parameter is the SQL Server account password. SnapManager ignores this parameter if the parameter -UserName is not specified.

-Logbkup - Short form: -lb

Use this option to specify that the transaction logs also need to be backed up after a full backup.

-Verify - Short form: -ver

Use this parameter if you wish to verify the backed up databases and logs.

-VerifyServerInstance <String> - Short form: -verInst

This parameter specifies the separate SQL server that is used to run the Database Consistency Check (DBCC) utility. If you have not specified the -verify parameter, SnapManager ignores this parameter.

The following example illustrates the usage:

```
-verInst win-225-161
```

Here the SQL server instance is the local or remote SQL server instance to verify on. SnapManager takes the configured SQL server instance that is used for verify in client configuration (registry) as the default SQL server instance.

-VerSvrLogin <String> - Short form: -verlogin

This parameter specifies that SQL Server authentication is used. If not specified, the default Windows NT Authentication mechanism is used.

-VerSvrPassword <String> - Short form: -verpwd

This parameter is used to input the verification server password. SnapManager ignores this parameter if the parameter -VerSvrLogin is not specified.

-VerDestVolume - Short form: -verdest

Use this parameter to verify the database on the SnapMirror destination volume. SnapManager sets it to "false" by default.

-VerifyOnDestVolumes <String[]> - Short form: -vermirror

Specify this parameter in order to override the default SnapMirror relationships. Enter the source and destination storage systems and volumes as a comma-separated list. SnapManager sets it to "false" by default.

-DBCCOption <EnumDbccOption[]> - Short form: -dbccopt

This parameter specifies options to the DBCC SQL command that are used to validate and verify the database that is being processed. When you use this parameter, you are explicitly requesting DBCC options, and the system does read the registry to determine the default DBCC options. The security access issues for the registry are bypassed when you use this cmdlet option. The parameter uses the following values:

NOOPTION

NOINDEX

ALL_ERRORMSGS

NO_INFOMSGS (default)

TABLOCK

PHYSICAL_ONLY (default)

For more information about these options, see your Microsoft SQL Server documentation.

-CloneOnMirrorDestination - Short form: -cloneonmir

Use this parameter to clone a database based on the Snapshot copy on the SnapMirror destination volume. Ensure that the SnapMirror relationship exists and SnapMirror was updated when using this option.

-ChangeClonePath - Short form: -chgpath

Use this parameter to change clone database paths based on the new database clone name.

-Resynchronize - Short form: -resync

Use this parameter to specify that the existing clone is refreshed with the live database.

-ForceTerminateConnection - Short form: -ftc

Use this parameter to specify that all the connections to the existing clone are terminated during clone resynchronize.

-ClusterAware - Short form: -cl

Use this parameter to specify that the cmdlet runs solely on the active node in a cluster environment.

-CloneMirrorDestVolumes <String[]> - Short form: -clonemir

Use this parameter to specify cloning using the Snapshot copy on the SnapMirror destination volume.

-VerifyDisable - Short form: -verDis

This parameter overrides verification and can disable verification even if the database was not verified after backup.

-UseMountPoint - Short form: -mp

This parameter specifies that the Snapshot copy must be mounted to an NTFS directory.

During a SnapManager verification operation, Snapshot copies are mounted to the default NTFS directory for database verification. The option is effective when there are no available drive letters to mount the Snapshot copies. It overrides pre-configured SnapManager verification settings.

-MountPointDir <String> - Short form: -mpdir

Use this parameter to specify the mount point directory on which a backup set will be mounted during database verification. Use this parameter with the parameter -UseMountPoint.

-UseDriveAvailable - Short form: -drvavail

Use this parameter to indicate that you should use available drive letter as mount point on which a backup set is mounted during database verification.

-RetainBackups <Int32> - Short form: -tgInst

Use this parameter to specify the number of backups to be retained after the delete operation.

-RetainBackupDays <Single> - Short form: -rtdays

Use this parameter to specify the number of days you want to retain the backups for. SnapManager deletes backups older than the specified number of days. The parameters RetainBackups and RetainBackupDays are mutually exclusive and cannot be specified together.

-AttachDB - Short form: -attdb

If the operation includes a database or transaction log verification, use this option when you want to specify that the databases are to be attached after the verification operation completes.

-UpdateMirror - Short form: -updmir

Use this option to update the SnapMirror destination after a backup or verification operation ends, if the operation uses backups that reside on volumes configured as SnapMirror sources.

-NoRetainUTM - Short form: -noutm

Use this option if you do not want to retain up-to-the-minute restore ability for older backups in other management groups.

-ManagementGroup <String> - Short form: -mgmt

This parameter denotes the backup or verify operation that SnapManager performs on daily, or weekly, or standard basis. The default management group is standard.

-LogBkupOnly - Short form: -lgbkonly

Use this option to back up your SQL Server transaction log files only. No full snapshot backup will be done.

-BkupSIF - Short form: -bksif

Use this option to create a Snapshot copy of the SnapInfo directory after the backup of the transaction log completes. The backup type should be a transaction log backup only.

-RetainSnapofSnapInfo <Int32> - Short form: -rtsifsnap

Use this option if you want to delete the oldest Snapshot copies in the SnapInfo directory, specified that the backup type is a transaction log backup only. It has an integer value. The following example illustrates the usage of this parameter: -rtsifsnap Number of SnapInfo Snapshots to keep

Note: This option is valid only if you specify the parameter - BkupSIF.

-RetainSnapofSnapInfoDays <Single> - Short form: -rtsifsnapdays

Use this parameter to delete SnapInfo Snapshot copies older than the specified number of days. This parameter is mutually exclusive with the parameter RetainSnapofSnapinfo and they cannot be specified together in the same cmdlet.

-TruncateSqlLog [<Boolean>] - Short form: -truncLog

This parameter specifies whether to truncate the SQL transaction logs. SQL transaction logs are truncated by default. Valid values are $true or $false. This parameter only works if -LogBkup or -LogBkupOnly are true.

-TruncateLogs - Short form: -trlog

This obsolete parameter (now replaced by TruncateSqlLog) specifies whether to truncate the SQL transaction logs. SQL transaction logs are not truncated by default. This parameter only works if -LogBkup or -LogBkupOnly are true. In SMSQL 5.2 and later, if neither -TruncateLogs or -TruncateSqlLog is specified, the default behavior is to truncate the logs.

-PreCommand <String> - Short form: -precmd

This parameter indicates to run a command before the current operation.

> **Note:** You cannot have more than one space between items that may be parsed in this parameter's value.

-PreCommandPath <String> - Short form: -precmdpath

This parameter specifies the operating system path to the command to be run before the SnapManager operation starts.

-PreCommandArguments <String> - Short form: -precmdargs

This parameter contains a list of strings of SnapManager operation-specific information or user-defined arguments to be passed to the program or script. The default is to pass no parameters to the script. If the parameter contains white spaces (tabs or spaces) you need to enclose it in double quotes. This parameter is processed only if the parameters -PreCommand and -PreCommandPath are specified.

-PreCommandHost <String> - Short form: -precmdhost

This parameter specifies the host machine name on which the command is run before the operation starts. The default is to run on the current machine. This parameter is considered only if the parameters -PreCommand and -PreCommandPath are specified.

-PreCommandErrors <EnumHandleCmdError[]> - Short form: -precmnderrors

This parameter specifies how to handle errors on the pre-command. The ContinueOnError value (the default) indicates that the SMSQL operation executes even if an error is detected during the pre-command launch. The StopOnPreCmdError value indicates that if a pre-command script gets an error, the remaining SMSQL operation is not attempted. This parameter is considered only if the parameters -PreCommand and -PreCommandPath are specified.

-PostCommand - Short form: -postcmd

This parameter indicates to run a command after the current operation is complete.

> **Note:** You cannot have more than one space between items that may be parsed in this parameter's value.

-PostCommandPath <String> - Short form: -postcmdpath

This parameter specifies the operation system path for the command to be run after the SMSQL operation is complete.

-PostCommandArguments <String> - Short form: -postcmdargs

This parameter contains a list of strings of SnapManager operation-specific information or user-defined arguments to be passed to the program or script. The default is to pass no parameters to the script. If the parameter contains white spaces (tabs or spaces) you enclose it in double quotes. This parameter is processed only if the parameters -PostCommand and -PostCommandPath are specified.

-PostCommandHost <String> - Short form: -postcmdhost

This parameter specifies the host machine name on which the command is run after the operation is complete. The default is to run on the current machine. This parameter is considered only if the parameters -PostCommand and -PostCommandPath are specified.

-PostCommandErrors <EnumHandleCmdError[]> - Short form: -postcmderrors

This parameter specifies how to handle SnapManager operation errors on the post-command run. The ContinueOnError value (the default) indicates that the SMSQL operation executes even if an error is detected during the post-command launch. The StopOnPostCmdError value indicates that if a post-command script gets an error, the remaining SMSQL operation is not attempted. This parameter is considered only if the parameters -PostCommand and -PostCommandPath are specified.

-RunDBCCAfter - Short form: -dbccaf

If the operation includes a database backup, use this parameter if you want to verify the live database after the backups are performed.

-RunDBCCBefore - Short form: -dbccbf

If the operation includes a database backup, use this parameter if you want to verify the live database before the backups are performed.

-GenericNaming - Short form: -gen

This parameter specifies that the backups must follow the Generic backup naming convention.

-ArchiveBackup - Short form: -arch

Use this parameter to archive database to a secondary storage system during the backup phase of the operation.

-VerifyArchiveBackup - Short form: -verarch

Use this parameter to verify database archived at the secondary storage system.

-ArchivedBackupRetention <String> - Short form: -archret

Use this parameter to specify whether you want to retain backups at the archived location on a daily, hourly, weekly, monthly or unlimited basis.

-ServerInstance <String[]> - Short form: -inst

This parameter specifies the SQL server instance where the database is backed up originally. SnapManager takes the local computer name as the default server instance.

You can specify multiple server instance names here as a comma-separated list. If multiple databases reside on the same LUN but are owned by different SQL server instances when you backed them up originally, use the following format:

```
-Inst "SQLServerInstance1","SQLServerInstance2"
```

The first database specified in the -Database parameter refers the first server instance in the -ServerInstance parameter, the second database in the -Database parameter refers to the second server instance in the -ServerInstance parameter and so on.

-Database <String[]> - Short form: -d

Use this option to specify the databases that need to be cloned. Use a comma-separated list of strings:

```
-d Database 1, Database 2, Database 3, Database 4,....
```

Multiple database names should be specified only if those databases share a single LUN or multiple LUNs together. For a multiple database restore, all the selected databases should be present in the selected Snapshot copy.

You cannot restore a database with a new name if you specify multiple databases. If you want to restore with a new name, restore those databases one by one. In case of restore to alternate location, specify only one database name.

-TransLogsToApply <Int32[]> - Short form: -translogs

This parameter specifies the count of transactions logs that need to be applied to each database restored. If the TransLogsToApply parameter is not given, then all transaction logs that apply to the full backup restored are applied by default (just as the GUI does). You can specify the number of transaction logs to be applied for every database mentioned in the -Database parameter. The list of number of transaction logs that are applied must be listed in the same sequence as the databases listed in the -Database parameter. For example:

-Database db1,db2

might correspond to:

-TransLogsToApply 1,8

which means 1 transaction log backup will be applied to db1, and 8 will be applied to db2.

-ForceRestore [<Boolean>] - Short form: -force

Use this parameter to force the restore of a database based on its state. SnapManager sets it's value to "true" by default.

-TargetDatabase <String[]> - Short form: -tgDb

Use this parameter to restore a database with a new name. The following example illustrates the usage:

```
-tgDb "NewDatabaseName1"," NewDatabaseName2"," NewDatabaseName3"
```

The parameter defines the new database name to which the original database is restored. The old database name is defined at the same position in the -Database parameter.

If no new database name is given, the database is restored to the original database name the database had during backup. If this original name already exists, the name is modified to: originalDbName__clone, or originalDbName__mount.

-TargetServerInstance <String[]> - Short form: -tgInst

This parameter specifies the name of the new SQL server if you want to restore the database to a new SQL server. SnapManager takes the source SQL server instance as the default.

-TargetServerMountPointDir <String> - Short form: -tgmpdir

Use this parameter to specify the mount point path or directory of the target server instance in which the databases are to be cloned or mounted.

-MarkName <String[]> - Short form: -mark

This parameter indicates the marked transaction at which to stop the transaction log recovery.

-MarkTime <String[]> - Short form: -mktm

This parameter specifies a unique timestamp to guarantee the uniqueness of the input restored mark.

-RestoreBeforeMark [<Boolean>] - Short form: -beforemk

This true or false value indicates whether the specified marked transaction log should be included in the restore.

-RecoverDatabase <Boolean[]> - Short form: -recoverdb

This parameter indicates whether the database will be fully recovered or left in a partially recovered state after the cmdlet finishes to facilitate future SQL transaction log restores. This is an array of booleans, so it must match the same number of elements of the -database array. If the it does not match the number of elements of the -database array, an error is given. This defaults to $true for all databases unless the -standbyPath is given, in which case it defaults to $false for all databases.

-StandbyPath <String> - Short form: -standby

This parameter indicates the path to the standby recover file where incomplete transactions are stored after restoring a full database and its transaction logs. There is no default if you specify this parameter. The path must be to the standby directory if more than one database shares a LUN. If the database is on a dedicated LUN, then it must be a specific file. If the -standbypath parameter is given, the -RecoveryDatabase given must be -RecoverDatabase $False, otherwise it defaults to $false for all databases if no -RecoverDatabase parameter is specified.

-apicontext - Short form: none

Use this parameter when calling the cmdlet as an API call.

-RestoreArchivedBackup - Short form: -rstarchbkup

Use this parameter to specify using remote backup to perform the clone operation.

-WhatIf - Short form: -wi

This parameter gives you a preview of an operation.

-Confirm - Short form: -cf

This parameter prompts you for confirmation before the actual operation starts.

-AvailabilityGroup <String> - Short form: -ag

This parameter specifies the name of the source Availability Group.

-SynchronousCommit - Short form: -syncCommit

This parameter specifies that replica databases are synchronized to their primary. If not specified, false is assumed.

-FailoverMode - Short form: -flMd

This parameter specifies that failover occur to the preferred replica, if the primary replica becomes unavailable. If not specified, false is assumed.

-ReadableSecondary - Short form: -readsec

This parameter specifies read-only access for all of the new secondary databases. If not specified, false is assumed.

<CommonParameters>

This cmdlet supports the common parameters: Verbose, Debug, ErrorAction, ErrorVariable, WarningAction, WarningVariable, OutBuffer and OutVariable. For more information, see about_CommonParameters (http://go.microsoft.com/fwlink/?LinkID=113216).

Examples

Example 1: `clone-replica -svr 'SQL2012HA2' -inst 'SQL2012HA2\INST2' -ag 'snapmgr2012 ' -tgInst 'SQL2012HA1\INST1'`

This command creates a secondary replica for the Availability Group "snapmgr2012" on the secondary "SQL2012HA1\INST1". Values for -SynchronousCommit, FailoverMode, and ReadableSecondary are not specified, so the default, false, is used.

Example 2: `clone-replica -svr 'SQL2012HA2' -inst 'SQL2012HA2\INST2' -ag 'snapmgr2012 ' -tgInst 'SQL2012HA1\INST1' -SychronousCommit -FailoverMode -ReadableSecondary`

This command creates a secondary replica for the Availability Group "snapmgr2012" on the secondary "SQL2012HA1\INST1". The replica is created with the properties synchronous commit, failover mode, and readable secondary set to true.

delete-backup

Name

delete-backup

Synopsis

This cmdlet enables you to delete the SnapManager backup sets using the SnapManager SQL Server PowerShell command-line interface.

Syntax

```
delete-backup [-Server <String>] [-UserName <String>] [-Password <String>]
[-ServerInstance <String>] -Database <String> -Backup <String> [-
apicontext] [-ArchiveBackup] [-SnapVaultSecondary] [-WhatIf] [-Confirm]
[<CommonParameters>]
```

Description

This cmdlet enables you to delete a database depending on the input criteria specified in the command-line interface. It deletes the specified backup set if it contains the specified database name.

You can also implement these options with the SnapManager user interface.

Parameters

-Server <String> - Short Form: -svr

This parameter denotes the name of the host SQL Server on which the SQL Server instances reside. SnapManager takes the local computer name as the default server name. In case of a clustered configuration, the virtual server name is the default server name.

Using this parameter, you can also specify a particular SQL Server instance. The following examples illustrate the usage:

```
-svr win-225-161
```

```
-svr sql1
```

-UserName <String> - Short Form: -usr

This parameter denotes the SQL Server account name. If the login name is not specified, SnapManager uses Windows NT Authentication.

-Password <String> - Short Form: -pwd

This parameter is the SQL Server account password. SnapManager ignores this parameter if the parameter -UserName is not specified.

-ServerInstance <String> - Short Form: -inst

This parameter specifies the SQL Server instance where the database is backed up originally. SnapManager takes the local computer name as the default server instance.

-Database <String> - Short Form: -d

This is a mandatory parameter that specifies a database.

-Backup <String> - Short Form: -bkup

Use this parameter to specify the backup set that needs to be deleted. It is a mandatory parameter.

-apicontext - Short form: none

Use this parameter when calling the cmdlet as an API call.

-ArchiveBackup - Short Form: -arcbk

Use this parameter to specify the archived backup set that needs to be deleted.

Note: This parameter is mandatory if you delete archived backup sets.

-SnapVaultSecondary - Short Form: -vaultsec

This optional parameter identifies the backup vault from which you want to delete the Snapshot copy. If you do not specify this parameter, all backups are deleted from the related backup vaults. You use this parameter in conjunction with the `-ArchiveBackup` parameter. This parameter applies to clustered Data ONTAP only. The syntax for this parameter is as follows:

```
-SnapVaultSecondary n, Vserver:volume
```

Where n is the number of Storage Virtual Machine (SVM, formerly known as Vserver):volume pairs.

Example: `-SnapVaultSecondary 3, Vserver1:volume1, Vserver2:volume2, Vserver3:volume3`

-WhatIf - Short form: -wi

This parameter gives you a preview of an operation.

-Confirm - Short form: -cf

This parameter prompts you for confirmation before the actual deletion operation starts.

<CommonParameters>

This cmdlet supports the common parameters: Verbose, Debug, ErrorAction, ErrorVariable, WarningAction, WarningVariable, OutBuffer and OutVariable. For more information, see about_CommonParameters (http://technet.microsoft.com/library/hh847884.aspx).

Examples

Example 1: `delete-backup -d "Db1" -bk "Db1bkup"`

This command deletes the backup set Db1bkup where DB1 is the cloned database.

Example 2: `delete-backup -d "Db1" -bk "Db1bkup" -ArchiveBackup -vaultsec 2,sn_vserver_dev:test_volume_sec,sn_vserver_dev:test_volume_sec2`

This command deletes the backup set Db1bkup from the specified SnapVault secondary volume.

delete-clone

Name

delete-clone

Synopsis

This cmdlet enables you to delete a cloned database.

Syntax

```
delete-clone [-Server <String>] [-UserName <String>] [-Password <String>]
[-ServerInstance <String>] -Database <String[]> [-JobInstance <String>] [-
ResyncCloneJob <String>] [-ClusterAware] [-TerminateConnection] [-
apicontext] [-WhatIf] [-Confirm] [ <CommonParameters>]
```

Description

This cmdlet helps you delete a cloned database using the SnapManager PowerShell command-line interface. Before deleting a clone, make sure all connections to the cloned database are disconnected.

You can also implement these options with the SnapManager user interface.

Parameters

-Server <String> - Short Form: -svr

This parameter denotes the name of the host SQL Server on which the SQL Server instances reside. SnapManager takes the local computer name as the default server name.

Using this parameter, you can also specify a particular SQL server instance. The following examples illustrate the usage:

```
-svr win-225-161
```

```
-svr sql1
```

-UserName <String> - Short Form: -usr

This parameter denotes the SQL Server account name. If the login name is not specified, SnapManager uses Windows NT Authentication.

-Password <String> - Short Form: -pwd

This parameter is the SQL server account password. SnapManager ignores this parameter if the parameter -UserName is not specified.

-ServerInstance <String> - Short Form: -inst

This parameter specifies the SQL server instance where the database is backed up originally. SnapManager takes the local computer name as the default server instance.

-Database <String[]> - Short Form: -d

This is a mandatory parameter that specifies the list of cloned databases to be deleted. Enter the cloned database names in a comma separated list.

-JobInstance <String> - Short Form: -jobinst

This parameter is followed by the name of the SQL Server instance on which the clone resync job is created.

-ResyncCloneJob <String> - Short Form: -rcjob

This parameter is followed by the name of the clone resync job for the specified cloned database.

-ClusterAware - Short Form: -cl

Use this parameter to specify that the cmdlet runs solely on the active node in a cluster environment.

-TerminateConnection - Short form: -terminate

Use this parameter to terminate open database connections.

-apicontext - Short form: none

Use this parameter when calling the cmdlet as an API call.

-WhatIf - Short form: -wi

This parameter gives you a preview of an operation.

-Confirm - Short form: -cf

This parameter prompts you for confirmation before the actual deletion operation starts.

<CommonParameters>

This cmdlet supports the common parameters: Verbose, Debug, ErrorAction, ErrorVariable, WarningAction, WarningVariable, OutBuffer and OutVariable. For more information, see about_CommonParameters (http://go.microsoft.com/fwlink/?LinkID=113216).

Examples

Example 1: `delete-clone -svr sql1 -d "Db1"`

This command deletes the clone Db1 on Server sql1.

Example 2: `delete-clone -svr 'SNAPMGR-25' -inst 'SNAPMGR-25' -d 'DB1__Clone', 'DB2__Clone' -ClusterAware -ResyncCloneJob "CloneResync_VDISK_W_07-08-2011_13-02-29" -JobInstance "SNAPMGR-19\MARS"`

This example deletes the clone of database "DB1" and "DB2" from SQL Server "SNAPMGR-25" and the corresponding clone refresh job "CloneResync_VDISK_W_07-08-2011_13-02-29" from SQL agent instance "SNAPMGR-19\MARS".

export-config

Name

export-config

Synopsis

This cmdlet enables you to export the existing configuration information of an SQL server to a control-file using SnapManager PowerShell command-line interface.

Syntax

```
export-config [-Server <String>] [-ControlFilePath <String>] [-Section
<String[]>] [-apicontext] [-exportobject] [-WhatIf] [-Confirm]
[<CommonParameters>]
```

Description

This cmdlet enables you to export the existing configuration information of an SQL server to a control-file using SnapManager PowerShell command-line interface.

You can also implement these options with the SnapManager user interface.

Parameters

-Server <String> - Short Form: -svr

This parameter denotes the name of the host SQL server on which the SQL server instances reside. SnapManager takes the local computer name as the default server name.

Using this parameter, you can also specify a particular SQL server instance. The following examples illustrate the usage:

```
-svr win-225-161
```

```
-svr sql1
```

-ControlFilePath <String> - Short Form: -config

This parameter specifies the name of the control-file and its path. SnapManager takes the current directory as the control-file path by default.

-Section <String[]> - Short Form: -sect

This parameter lists section names that are to be imported (separated by commas). If you do not specify any particular section, the default value of all sections is applied. The valid section names that can be applied are as follows: storage, notification, verification, report, backup, scheduledjob, runcommand, snapmirrorvolume, monitor, and clonejob.

-apicontext - Short form: none

Use this parameter when calling the cmdlet as an API call.

-exportobject - Short form: none

Use this parameter is to publish the configuration information as objects either shown on the output screen or to be piped to another cmdlet. This facilitates easy communication with SMSPS. Without this parameter, the default behavior is to export configuration information as an .xml file.

-WhatIf - Short form: -wi

This parameter gives you a preview of an operation.

-Confirm - Short form: -cf

This parameter prompts you for confirmation before the actual operation starts.

<CommonParameters>

This cmdlet supports the common parameters: Verbose, Debug, ErrorAction, ErrorVariable, WarningAction, WarningVariable, OutBuffer, and OutVariable. For more information, see about_CommonParameters (http://go.microsoft.com/fwlink/?LinkID=113216).

Examples

Example 1 `export-config -Server win-225-166 -ControlFilePath "C:\Program Files\NetApp\SnapManager for SQL Server\SMSQLConfig_16July_test4.xml" -Section storage,notification`

This cmdlet exports all sections of the existing configuration and settings to the specified control-file.

get-backup

Name

get-backup

Synopsis

This cmdlet allows you to list the backup sets made by SnapManager for Microsoft SQL Server.

Syntax

```
get-backup [-Server <String>] [-BackupServer <String>] [-UserName <String>] [-Password <String>] [-ServerInstance <String>] [-Database <String>] [-SnapInfoDirectory <String>] [-apicontext] [-WhatIf] [-Confirm] [<CommonParameters>]
```

Description

This cmdlet enables you to list the backup sets of a particular database by specifying an SQL Server, an SQL Server instance, or a database set. You can also implement these options with the SnapManager user interface.

Parameters

-Server <String> - Short Form: -svr

This parameter denotes the name of the host SQL Server on which the SQL Server instances reside. SnapManager takes the local computer name as the default server name.

Using this parameter, you can also specify a particular SQL Server instance. The following examples illustrate the usage:

```
-svr win-225-161
```

```
-svr sql1
```

For virtual server instances, specify the virtual server name. For example:

```
get-backup -server <virtual_server> -ServerInstance <virtual_instance> -d
aa1
```

-BackupServer <String> - Short Form: -bksvr

Use this parameter to specify where the backup was originally created. Use the host name or cluster name where the SQL Server instance resides. This parameter cannot be an SQL Server instance name. This parameter is optional, and is mainly used for a restore backup created from a different server. For example, this parameter can be used for DR using SnapMirror. By default, the backup server is the server currently connected, specified by -Server parameter. For example:

```
-Server win2k8-248-137 -backupserver 'SQL2K8VI1' -inst 'SQL2K8VI1\DE1' -
TargetServerInstance win2k8-248-137
```

The server is connected to a new server where the restore will be performed. But the backup was originally created on 'SQL2K8VI1', and the instance was 'DE1'.

-Username <String> - Short Form: -usr

This parameter denotes the SQL Server account name. If the login name is not specified, SnapManager uses Windows NT Authentication.

-Password <String> - Short Form: -pwd

This parameter is the SQL Server account password. SnapManager ignores this parameter if the parameter -UserName is not specified.

-ServerInstance <String> - Short Form: -inst

This parameter specifies the SQL Server instance where the database is backed up originally. SnapManager takes the local computer name as the default server instance. For named SQL Server instances, enter the instance in the following format: `HostName\InstanceName`

-Database <String> - Short Form: -d

This is a mandatory parameter that specifies the database. If you do not specify the database parameter, the cmdlet backs up all of the SQL Server instances that are peer instances of the SQL server in the `-Server` parameter.

-SnapInfoDirectory <String> - Short Form: -sif

This parameter enables you to list the system and user databases on a remote server.

-apicontext - Short form: none

Use this parameter when calling the cmdlet as an API call.

-WhatIf - Short form: -wi

This parameter gives you a preview of an operation.

-Confirm - Short form: -cf

This parameter prompts you for confirmation before the actual operation starts.

<CommonParameters>

This cmdlet supports the common parameters: Verbose, Debug, ErrorAction, ErrorVariable, WarningAction, WarningVariable, OutBuffer and OutVariable. For more information, see about_CommonParameters (http://go.microsoft.com/fwlink/?LinkID=113216).

Examples

Example 1 `get-backup -svr 'VM-VS-1' -inst vm-vs-1 -d 'ds_test7'`

This example retrieves the backed up database on a server instance of the specified server.

Example 2 `get-backup -svr snapmgr-62 -inst snapmgr-63\FEDERATED -snapinfo \ \172.17.233.163\ G$\SMSQL_SnapInf`

This example shows all the server and user databases on the remote server.

import-config

Name

import-config

Synopsis

This cmdlet enables you to import the configuration information from a SnapManager for SQL control-file using SnapManager PowerShell command-line interface.

Syntax

```
import-config [-Server <String>] [-ControlFilePath <String>] [-Section
<String[]>] [-ValidateAndApply] [-AllowLocal] [-UserName <String>] [-
Password <String>] [-ClusterAware] [-DBCCBefore [<Boolean>]] [-DBCCAfter
[<Boolean>]] [-DeleteOriginalDBFile [<Boolean>]] [-UpdateStatisticsTable
[<Boolean>]] [-apicontext] [-WhatIf] [-Confirm] [<CommonParameters>]
```

Description

This cmdlet enables you to import the configuration information from a SnapManager for SQL control-file using SnapManager PowerShell command-line interface. You can import sections like storage, notification, verification, report, backup, scheduled job, snapmirror volume and so on. You can also control DBCC integrity verification and update statistics table using this cmdlet.

You can also implement these options with the SnapManager user interface.

Parameters

-Server <String> - Short Form: -svr

This parameter denotes the name of the host SQL server on which the SQL server instances reside. SnapManager takes the local computer name as the default server name.

Using this parameter, you can also specify a particular SQL server instance. The following examples illustrate the usage:

```
-svr win-225-161
```

```
-svr sql1
```

-ControlFilePath <String> - Short Form: -config

This parameter specifies the name of the control-file and its path. SnapManager takes the current directory as the control-file path by default.

-Section <String[]> - Short Form: -sect

This parameter lists section names that are to be imported (separated by commas). If you do not specify any particular section, the default value of all sections is applied. The valid section names that can be applied are as follows: storage, notification, verification, report, backup, scheduledjob, runcommand, snapmirrorvolume, monitor, and clonejob.

-ValidateAndApply - Short Form: -apply

This parameter applies the imported storage and notification settings data to the current system after validation. If you specify this parameter and validation is successful the imported data will be applied. If you do not specify this parameter only validation occurs.

-AllowLocal - Short Form: -tolocal

This parameter specifies that the migration of databases to the local disk is permitted. Its value is set to "false" by default.

-UserName <String> - Short Form: -usr

This parameter denotes the SQL Server account name. If the login name is not specified, SnapManager uses Windows NT Authentication. This parameter is mandatory if you import a scheduled job.

-Password <String> - Short Form: -pwd

This parameter is the SQL server account password. SnapManager ignores this parameter if the parameter -UserName is not specified. This parameter is mandatory if you import a scheduled job.

-DBCCBefore [<Boolean>] - Short Form: -dbcc

This parameter runs the DBCC physical integrity verification before migration. Its value is set to "true" by default.

-DBCCAfter [<Boolean>] - Short Form: -dbcc2

This parameter runs the DBCC physical integrity verification after migration. Its value is set to "false" by default.

-DeleteOriginalDBFile [<Boolean>] - Short Form: -deletedbfile

This parameter deletes the copy of the migrated database at original location. Its value is set to "true" by default.

-UpdateStatisticsTable [<Boolean>] - Short Form: -updatestatistics

This parameter runs "Update statistics" on tables before detaching the databases. Its value is set to "true" by default.

-apicontext - Short form: none

Use this parameter when calling the cmdlet as an API call.

-WhatIf - Short form: -wi

This parameter gives you a preview of an operation.

-Confirm - Short form: -cf

This parameter prompts you for confirmation before the actual operation starts.

<CommonParameters>

This cmdlet supports the common parameters: Verbose, Debug, ErrorAction, ErrorVariable, WarningAction, WarningVariable, OutBuffer and OutVariable. For more information, see about_CommonParameters (http://go.microsoft.com/fwlink/?LinkID=113216).

Examples

Example 1: `import-config -server "sql1" -ControlFilePath "C:\Program Files \NetApp\SnapManager for SQL\SMSQLConfig_01_23_2007_23.10.20.xml" -Section backup`

This cmdlet validates the backup settings in the control-file. It does not apply the settings to the SQL server.

Example 2 `import-config -Server win-225-166 -Section storage,notification -ControlFilePath "C:\Program Files\NetApp\SnapManager for SQL Server \SMSQLConfig_16July_test4.xml" -ValidateAndApply -AllowLocal`

This cmdlet validates the imported storage and notification settings from control-file and applies it to the system.

new-backup

Name

new-backup

Synopsis

This cmdlet enables you to back up the SQL server databases in the SnapManager PowerShell command-line interface.

Syntax

```
new-backup [-Server <String>] [-UserName <String>] [-Password <String>] [-
Database <String[]>] [-FederatedGroups <String[]>] [-Mark <String>] [-
MarkDesc <String>] [-LogBkup] [-Verify] [-VerifyServerInstance <String>] [-
VerSvrLogin <String>] [-VerSvrPassword <String>] [-RetainBackups <Int32>]
[-RetainBackupDays <Single>] [-RetainUtmBackups <Int32>] [-RetainUtmDays
<Single>] [-UseMountPoint] [-MountPointDir <String>] [-UseDriveAvailable]
[-AttachDB] [-UpdateMirror] [-NoRetainUTM] [-VerDestVolume] [-
ManagementGroup <String>] [-LogBkupOnly] [-BkupSIF] [-RetainSnapofSnapInfo
<Int32>] [-RetainSnapofSnapInfoDays <Single>] [-TruncateSqlLog [<Boolean>]]
[-TruncateLogs] [-Command] [-RunCommand <String>] [-CommandArguments
<String>] [-CommandServer <String>] [-PreCommand] [-PreCommandPath
<String>] [-PreCommandArguments <String>] [-PreCommandHost <String>] [-
PreCommandErrors <EnumHandleCmdError[]>] [-PostCommand] [-PostCommandPath
<String>] [-PostCommandArguments <String>] [-PostCommandHost <String>] [-
PostCommandErrors <EnumHandleCmdError[]>] [-RunDBCCAfter] [-RunDBCCBefore]
[-DBCCOption <EnumDbccOption[]>] [-GenericNaming] [-GenericNamingAdvanced]
[-VerifyOnDestVolumes <String[]>] [-apicontext] [-ArchiveBackup] [-
VerifyArchiveBackup] [-ArchivedBackupRetention <String>] [-ClusterAware] [-
WhatIf] [-Confirm] [-AvailabilityGroup] [-BackupPriority] [-Primary] [-
Secondary] [-CopyOnly] [-PreferredBackupReplica] [-CopyOnlyLogBackup] [-
CopyLogBackupToShare] [-RetainShareBackups] [-RetainShareBackupDays]
[<CommonParameters>]
```

Description

This cmdlet enables you to begin the backup-only and backup-with-verify operations. SnapManager provides a separate cmdlet for verification. You can also implement these options with the SnapManager user interface.

Parameters

-Server <String> - Short Form: -svr

This parameter denotes the name of the host SQL Server on which the SQL server instances reside. SnapManager takes the local computer name as the default server name. If no default host exists, SnapManager attempts to use the following as the default:

- The `VerifyServerInstance` specified by the user
- The configured verification server for the current machine (in the registry) done in the configuration wizard, or backup verification settings
- The `VerificationServerInstance` from the SQL Server being backed up as the verification server
- The current machine

Using this parameter, you can also specify a particular SQL Server instance. The following examples illustrate the usage:

```
-svr win-225-161
```

```
-svr sql1
```

To back up all instances on a server that has a default instance, specify the following:

```
-server <server_name>
```

To back up all instances on a server that does not have a default instance, specify one of the named instances on the server in the following format:

```
-server <host\instance>
```

To back up all databases on specified instances, use the following format:

```
-server <SQL_server_name or host\instance> -d <host\instance>, 0
```

For example:

```
–server 'sql1' -d 'sql1\instance1', '0', 'sql1\instance2', '0'
```

-Username <String> - Short Form: -usr

This parameter denotes the SQL Server account name. If the login name is not specified, SnapManager uses Windows NT Authentication.

-Password <String> - Short Form: -pwd

This parameter is the SQL Server account password. SnapManager ignores this parameter if the parameter `-UserName` is not specified.

-Database <String[]> - Short Form: -d

Use this parameter to specify the original database that you want to back up. You can also specify multiple database names, but only if the databases share a single LUN or multiple LUNs. In this case, list the databases followed by `-Database` in following format:

```
-database sql-server-instance, count-of-databases, "database1"," database2"
```

If you do not specify the database parameter explicitly, the cmdlet backs up all the databases from all the SQL Server instances in the host. If storage other than NetApp storage exists on your system, the cmdlet omits databases located on that storage. Databases incompletely configured or databases in incompatible states are omitted when not explicitly provided with this parameter.

-FederatedGroups <String[]> - Short Form: -g

This parameter specifies the original federated groups to back up. If you specify multiple federated groups, the items in the list are separated by commas. If you do not specify the `FederatedGroups` parameter, the cmdlet backs up only the databases specified in the `Database` parameter. If neither parameter is specified, the cmdlet backs up all SQL server instances that are peer instances of the SQL server in the -Server parameter.

-Mark <String> - Short Form: -m

Use this parameter to specify a mark name when backing up transaction logs. If you do not specify a name, the default mark name “snapmgr_sqlbackup_[timestamp]” is used.

-MarkDesc <String> - Short Form: -md

Use this parameter to specify a mark description when backing up transaction logs. If you do not specify a name, the default mark description “snapmanager sql backup mark generated at [timestamp]” is used.

-Logbkup - Short form: -lb

Use this option to specify that the transaction logs also need to be backed up after a full backup.

-Verify - Short form: -ver

Use this parameter to verify the backed up databases and logs.

-VerifyServerInstance <String> - Short form: -verInst

This parameter specifies the separate SQL server that is used to run the Database Consistency Check (DBCC) utility. If you have not specified the `-verify` parameter, SnapManager ignores this parameter.

The following example illustrates the usage:

```
-verInst win-225-161
```

In this example, the SQL server instance is the local or remote SQL server instance to verify on. SnapManager takes the configured SQL server instance that is used for verify in client configuration (registry) as the default SQL server instance.

-VerSvrLogin <String> - Short form: -verlogin

This parameter specifies that SQL Server authentication is used. If not specified, the default Windows NT Authentication mechanism is used.

-VerSvrPassword <String> - Short form: -verpwd

This parameter is used to input the verification server password. SnapManager ignores this parameter if the parameter `-VerSvrLogin` is not specified.

-RetainBackups <Int32> - Short Form: -rtbackups

Use this parameter to specify the number of backups to be retained after the delete operation.

-RetainBackupDays <Single> - Short Form: -rtdays

Use this parameter to specify the number of days you want to retain the backups. SnapManager deletes backups older than the specified number of days. The parameters `RetainBackups` and `RetainBackupDays` are mutually exclusive.

-RetainUTMBackups <Single> - Short Form: -rubackups

Specifies the number of the most recent backup copies to retain up-to-the-minute restore ability after a SnapManager backup operation.

-RetainUTMDays <Single> - Short Form: -rudays

Specifies the number of days that the backups created within the time period retain up-to-the-minute restore ability. The backups older than the specified number of days lose up-to-the-minute restore ability and are for point-in-time restore only. Use this option in conjunction with `RetainUtmBackups`. Absence of this option denotes a up-to-the-minute restore ability retain policy based on backup count.

-UseMountPoint - Short Form: -mp

This parameter is a switch that specifies that the Snapshot copy must be mounted to an NTFS directory. During a SnapManager verification operation, Snapshot copies are mounted to the default NTFS directory for database verification. The option is effective when there are no available drive letters to mount the Snapshot copies. It overrides preconfigured SnapManager verification settings.

-MountPointDir <String> - Short Form: -mpdir

Use this parameter to specify the mount point directory on which a backup set is mounted during database verification. This parameter should be used along with the parameter `-UseMountPoint`.

Note: This option is valid only if you specify the parameter `-BkupSIF`.

-UseDriveAvailable - Short Form: -drvavail

Use this parameter to specify the mount point with available drive on which a backup set is to be mounted during database verification.

-AttachDB - Short Form: -attdb

If the operation includes a database or transaction log verification, use this option when you want to specify that the databases are to be attached after the verification operation completes.

-UpdateMirror - Short Form: -updmir

Use this option to update the SnapMirror destination after the backup or verification operations are complete, if you are using backups that reside on volumes configured as SnapMirror sources.

-NoRetainUTM - Short Form: -noutm

Use this option if you do not want to retain up-to-the-minute restore ability for older backups in other management groups.

-VerDestVolume - Short Form: -verdest

Use this parameter to verify the database on the SnapMirror destination volume. SnapManager sets it to “false” by default.

-ManagementGroup <String> - Short form: -mgmt

This parameter denotes the backup or verify operation that SnapManager performs on a daily, weekly, or standard basis. The default management group is standard.

-LogBkupOnly - Short form: -lgbkonly

Use this option to back up your SQL Server transaction log files only. No full Snapshot copy is made.

-BkupSIF - Short form: -bksif

Use this option to create a Snapshot copy of the SnapInfo directory after the backup of the transaction log finishes. The backup type should be a transaction log backup only.

-RetainSnapofSnapInfo <Int32> - Short form: -rtsifsnap

Use this option if you want to delete the oldest Snapshot copies in the SnapInfo directory, specified that the backup type is a transaction log backup only. It has an integer value. The following example illustrates the use of this parameter: -rtsifsnap Number of SnapInfo Snapshots to keep

Note: This option is valid only if you specify the parameter - `BkupSIF`.

-RetainSnapofSnapInfoDays <Single> - Short form: -rtsifsnapdays

Use this parameter to delete SnapInfo Snapshot copies older than the specified number of days. This parameter is mutually exclusive with the parameter `RetainSnapofSnapinfo`.

-TruncateSqlLog [<Boolean>] - Short form: -truncLog

This parameter specifies whether to truncate the SQL transaction logs. SQL transaction logs are truncated by default. Valid values are $true or $false. This parameter is valid only if `-LogBkup` or `-LogBkupOnly` are true.

-TruncateLogs - Short form: -trlog

This obsolete parameter (now replaced by `TruncateSqlLog`) specifies whether to truncate the SQL transaction logs. SQL transaction logs are not truncated by default. This parameter is valid only if `-LogBkup` or `-LogBkupOnly` are true. In SMSQL 5.2 and later, if neither `-TruncateLogs` or `-TruncateSqlLog` is specified, the default behavior is to truncate the logs.

-Command - Short form: -cmd

This switch parameter runs a command after the backup or verify operation.

-RunCommand - Short form: -runcmd

This parameter runs the specified command after the SnapManager backup or verification operation is complete. It defines the complete path for the command to be run after the backup or verify operation is complete. There is no default.

-CommandArguments <String> - Short form: -cmdargs

This option contains the string of SnapManager operation-specific information to be passed to your program or script. It is considered only if `Command` and `RunCommand` are specified. There is no default.

-CommandServer <String> - Short form: -cmdsvr

This obsolete parameter (now replaced by `PostCommandHost`) was used to indicate the machine where the desired command should run after the operation is complete. The default was to run on the current machine. This was only considered if `-command` and `-RunCommand` were specified.

-PreCommand <String> - Short form: -precmd

This parameter indicates to run a command before the current operation.

-PreCommandPath <String> - Short form: -precmdpath

This parameter specifies the operating system path to the command to be run before the SnapManager operation starts.

-PreCommandArguments <String> - Short form: -precmdargs

Use this parameter to specify a list of strings of SnapManager operation-specific information or user-defined arguments to be passed to the program or script.

-PreCommandHost <String> - Short form: -precmdhost

Use this parameter to specify the host machine name on which the command is to be run before the operation starts.

-PreCommandErrors <EnumHandleCmdError[]> - Short form: -precmnderrors

Use this parameter to specify how to handle errors on the precommand and the following SMSQL operation. The `ContinueOnError` value indicates that the following SMSQL operation is to be executed anyway. The `StopOnPreCmdError` value indicates that if a precommand script results in an error, the remaining SMSQL operation is not attempted.

-PostCommand - Short form: -postcmd

Use this parameter to run a command after the current operation.

-PostCommandPath <String> - Short form: -postcmdpath

Use this parameter to specify the operating system path to the command to be run after the SMSQL operation starts.

-PostCommandArguments <String> - Short form: -postcmdargs

Use this parameter to specify a list of strings of SnapManager operation-specific information or user defined arguments to be passed to the program or script.

-PostCommandHost <String> - Short form: -postcmdhost

Use this parameter to specify the host name on which the command is to be run after the operation is complete.

-PostCommandErrors <EnumHandleCmdError[]> - Short form: -postcmderrors

Use this parameter to specify how to handle errors on the next postcommand run. The `ContinueOnError` value indicates that the following SMSQL operation is to be executed anyway. The `StopOnPostCmdError` value indicates that if a postcommand script results in an error, the remaining SMSQL operation is not attempted.

-RunDBCCAfter - Short form: -dbccaf

If the operation includes a database backup, use this parameter if you want to verify the live database after the backups are performed.

-RunDBCCBefore - Short form: -dbccbf

If the operation includes a database backup, use this parameter if you want to verify the live database before the backups are performed.

-DBCCOption <EnumDbccOption[]> - Short form: -dbccopt

This parameter specifies options to the DBCC SQL command that are used to validate and verify the database that is being processed. When you use this parameter, you are explicitly requesting DBCC options, and the system reads the registry to determine the default DBCC options. The security access issues for the registry are bypassed when you use this cmdlet option. The parameter uses the following values:

NOOPTION

NOINDEX

ALL_ERRORMSGS

NO_INFOMSGS (default)

TABLOCK

PHYSICAL_ONLY (default)

For more information about these options, see your Microsoft SQL Server documentation.

-GenericNaming - Short Form: -gen

This parameter sets the naming convention for new backups as generic.

-GenericNamingAdvanced - Short Form: -genadv

This parameter sets the naming convention for new backups as enhanced.

After you use or enable the enhanced naming convention, it is permanent. You should continue to use `-genadv` and not revert to using only `-gen`.

-VerifyOnDestVolumes <String[]> - Short form: -vermirror

Specify this parameter to override the default SnapMirror relationships. Enter a comma-separated list of the source storage system, the source volume, the destination storage system, and the destination volume.

-apicontext - Short form: none

Use this parameter when calling the cmdlet as an API call.

-ArchiveBackup - Short Form: -arch

Use this parameter to archive the database to a secondary storage system.

-VerifyArchiveBackup - Short Form: -verarch

Use this parameter to verify the database archived at the secondary storage system.

-ArchivedBackupRetention <String> - Short Form: -archret

Use this parameter to specify whether you want to retain backups at the archived location on a daily, hourly, weekly, monthly, or unlimited basis.

-ClusterAware - Short form: cl

Use this parameter to specify that the cmdlet runs solely on the active node in a cluster environment.

-WhatIf - Short form: -wi

This parameter gives you a preview of an operation.

-Confirm - Short form: -cf

This parameter prompts you for confirmation before the actual operation starts.

-AvailabilityGroup <String> - Short Form: -ag

Use this parameter to specify one or more names of Availability Groups to which this backup applies.

-BackupPriority <Integer,Integer> - Short Form: -bp

Use this parameter to specify a set of secondary Availability Groups on a cluster by specifying a range of backup priorities. The operation applies to all replicas with backup priorities in that range. The maximum priority must be in the range of 1 to 100. The minimum backup priority must be less than or equal to the maximum priority.

-Primary - Short form: -prm

If this parameter is defined, then the backup is only made on the primary replica. If `BackupPriority` is also defined, then the primary replica must also satisfy the `BackupPriority` values.

-Secondary - Short form: -sec

If this parameter is defined, then the backup is made on all secondary replicas. If `BackupPriority` is also defined, then the secondary replicas must also satisfy the `BackupPriority` values.

-CopyOnly - Short form: -cpyonly

If this parameter is defined, a full backup is made as a copy-only full backup.

-PreferredBackupReplica - Short form: -preferbkreplica

Use this parameter to specify that only the preferred backup replica is backed up. The preferred backup replica is set from the Availability Group properties in the SQL Server 2012 Management Studio.

-CopyOnlyLogBackup - Short form: -cpyonlylgBk

Use this parameter to specify that transaction log backups are made as copy-only log backups.

-CopyLogBackupToShare <EnumBackupToShareType[]> - Short Form: -cpylgbkshare

Use this parameter to specify which transaction log backups are copied to the predefined repository share. The possible values are one of NOTHING_TOSHARE, COPYLOG_TOSHARE, or COPYLOG_TOSHARE_AGONLY. The repository share is set by the SnapManager for SQL repository share option.

-RetainShareBackups <Integer> - Short Form: -rtsharebackups

Use this parameter to specify the number of log backups retained in the SnapManager for SQL repository share.

-RetainShareBackupDays <Integer> - Short Form: -rtsharedays

Use this parameter to specify for how many days log backups are retained in the SnapManager Repository Share.

If you specify `-PreferredBackupReplica` along with `-Primary`, `-Secondary`, or `-BackupPriority`, the `-PreferredBackupReplica` value is used, and the others are ignored.

<CommonParameters>

This cmdlet supports the common parameters `Verbose`, `Debug`, `ErrorAction`, `ErrorVariable`, `WarningAction`, `WarningVariable`, `OutBuffer`, and `OutVariable`. For more information, see about_CommonParameters (http://go.microsoft.com/fwlink/?LinkID=113216).

Examples

Example 1: `new-backup -Server 'DBServer1' -Verify - VerifyServerInstance 'Snapmgr-50'`

This command creates a backup of all databases on the host DBServer1 and verifies the backups using the remote server Snapmgr-50.

Example 2: `new-backup -svr 'VM-VS-1' -d 'VM-VS-1', '4', 'ds_test1', 'ds_test2', 'ds_test6', 'ds_test7' -ver -verInst 'ZEUS-VM1\VERSERVER' -rtbackups 7 -lb -bksif -rtsifsnap 8 -trlog -noutm -mgmt standard -ArchiveBackup -VerifyArchiveBackup -ArchivedBackupRetention daily`

This example illustrates the creation of a new backup with verification of local backups and archive backups.

Example 3: `new-backup -svr 'VM-VS-1' -d 'VM-VS-1', '2', 'model', 'sm_test' -ver -verInst 'ZEUS-VM1\VERSERVER' -rtbackups 7 -lb -bksif -rtsifsnap 8 -trlog -noutm -gen -mgmt standard`

This example creates a new backup with the generic naming convention.

Example 4: `new-backup -svr 'VM-VS-1' -d 'VM-VS-1', '2', 'model', 'sm_test' -ver -verInst 'ZEUS-VM1\VERSERVER' -rtbackups 7 -lb -bksif -rtsifsnap 8 -trlog -noutm -mgmt standard`

This example creates a new backup with the unique naming convention.

Example 5: `new-backup -Server 'SNAPMGR-63' -Database 'SNAPMGR-63\SQL63INSTANCE1', '2', 'master', 'testdb2', 'SNAPMGR-63\SQL63INSTANCE2', '1', 'testdb1', 'SNAPMGR-19\SQLINSTANCE', '3', 'testdb1', 'testdb2', 'testdb3'`

This example creates a new backup with the federated backup feature.

Example 6: `new-backup -Server 'SNAPMGR-63' -Database 'SNAPMGR-63\SQL63INSTANCE1', '2', 'testdb4', 'testdb5'-FederatedGroups 2, 'SNAPMGR-63\SQL63INSTANCE1', '1', 'testdb1', 'SNAPMGR-19\SQLINSTANCE', '1', 'testdb2', 3, 'SNAPMGR-63\SQL63INSTANCE1', '2', 'testdb2', testdb3', 'SNAPMGR-63\SQL63INSTANCE2', '1', 'testdb1', 'SNAPMGR-19\SQLINSTANCE', '2', 'testdb1', 'testdb3',1, 'SNAPMGR-63\SQL63INSTANCE3', 0`

This example creates backups on all replicas.

Example 7: `new-backup -svr 'SQL2012HA2' -ag snapmgr2012 -prm -sec -mgmt standard`

This example creates backups on all replicas, because the default is all replicas.

Example 8: `new-backup -svr 'SQL2012HA2' -ag snapmgr2012 -mgmt standard`

This example creates backups on all replicas with backup priorities within the range of 50 to 70, because the default is all replicas.

Example 9: `new-backup -svr 'SQL2012HA2' -ag snapmgr2012 -prm -sec -bp 50,70 -mgmt standard`

This example creates a backup of the preferred replica.

Example 10: `new-backup -svr 'SQL2012HA2' -ag snapmgr2012 -PreferredBackupReplica -mgmt standard`

reseed-backup

Name

reseed-backup

Synopsis

This command enables you to reseed databases from SnapManager backups.

Syntax

```
reseed-backup [-Server <String>] [-UserName <String>] [-Password <String>] [-ServerInstance <String[]>] -Database <String[]> [-Backup <String>] [-RestoreLastBackup <Int32>] [-VerifyServerInstance <String>] [-VerSvrLogin <String>] [-VerSvrPassword <String>] [-VerifyDisable] [-DBCCOption <EnumDbccOption[]>] [-apicontext] [-PreCommand] [-PreCommandPath <String>] [-PreCommandArguments <String>] [-PreCommandHost <String>] [-PreCommandErrors <EnumHandleCmdError[]>] [-PostCommand] [-PostCommandPath <String>] [-PostCommandArguments <String>] [-PostCommandHost <String>] [-PostCommandErrors <EnumHandleCmdError[]>] [-AvailabilityGroup] [-RestoreArchivedBackup] [-SnapVaultSecondary] [-IgnoreRepLogs] [-WhatIf] [-Confirm] [<CommonParameters>]
```

Description

This cmdlet enables you to reseed a secondary database or a secondary Availability group replica. It has many other options. You can also implement these options with the SnapManager user interface.

Parameters

-Server <String> - Short Form: -svr

This parameter denotes the name of the host SQL Server on which the SQL Server instances reside. SnapManager takes the local computer name as the default server name.

Using this parameter, you can also specify a particular SQL server instance. The following examples illustrate the usage:

```
-svr win-225-161
```

```
-svr sql1
```

-Username <String> - Short Form: -usr

UserName is the SQL Server account name. It is specified if the SQL Server computer is accessed using a different account from that used to access the production SQL Server. If not specified, the Windows NT Authentication user name is used.

-Password <String> - Short Form: -pwd

This parameter is the SQL Server account password. SnapManager ignores this parameter if the parameter `-UserName` is not specified.

-ServerInstance <String[]> - Short Form: -inst

This parameter specifies the SQL Server instance on which the database is backed up originally. SnapManager takes the local computer name as the default server instance.

You can specify multiple server instance names in one list, separated by commas. If multiple databases reside on the same LUN but are owned by different SQL Server instances when you backed them up originally, use the following format:

```
-Inst "SQLServerInstance1","SQLServerInstance2"
```

The first database specified in the `-Database` parameter refers the first server instance in the `-ServerInstance` parameter, the second database in the `-Database` parameter refers to the second server instance in the `-ServerInstance` parameter, and so on.

-Database <String[]> - Short Form: -d

Use this parameter to specify the original database that you want to reseed. You can specify multiple database names using this option if the databases share a single LUN or multiple LUNs; also, the backups for multiple databases must all have the same name. Use the following format:

```
-Database "DatabaseName1"," DatabaseName2"
```

Note: All the databases selected should be present in the selected Snapshot copy.

-Backup <String> Short Form: -bkup

Use this option to specify the name of the backup set. The following example illustrates the usage:

```
-bkup sqlsnap__SYMNASQLDEV170_04-11-2007_15.22.27
```

-RestoreLastBackup <Int32> - Short Form: -lastBkup

Use this parameter to restore backups without specifying the name. If you try to use the `Backup` and `RestoreLastBackup` parameters together, SnapManager ignores the `RestoreLastBackup` parameter and uses the `Backup` parameter during restore operation. A typical usage example of the `restorelastbackup` parameter is as follows:

```
restore-backup -restorelastbackup 1
```

Note: If the value for `RestoreLastBackup` parameter is 0, SnapManager reseeds the latest backup. If the value is 1, SnapManager reseed the second-to-latest backup, and so on.

-VerifyServerInstance <String> - Short Form: -verInst

This parameter specifies the separate SQL Server that is used to run the Database Consistency Check (DBCC) utility. If you have not specified the `-verify` parameter, SnapManager ignores this parameter.

The following example illustrates the usage:

```
-verInst win-225-161
```

In this example, the SQL Server instance is the local or remote SQL Server instance to verify on. SnapManager takes the configured SQL Server instance that is used for verify in client configuration (registry) as the default SQL Server instance.

-VerSvrLogin <String> - Short Form: -verlogin

This parameter specifies that SQL Server authentication is used. If not specified, the default Windows NT Authentication mechanism is used.

-VerSvrPassword <String> - Short Form: -verpwd

SnapManager ignores this parameter if the parameter `-VerSvrLogin` is not specified.

-VerifyDisable - Short Form: -verDis

This parameter overrides verification and can disable verification even if the database was not verified after backup.

-DBCCOption <EnumDbccOption[]> - Short form: -dbccopt

This parameter specifies options to the DBCC SQL command that are used to validate and verify the database that is being processed. When you use this parameter, you are explicitly requesting DBCC options, and the system does read the registry to determine the default DBCC options. The security access issues for the registry are bypassed when you use this cmdlet option. The parameter uses the following values:

NOOPTION

NOINDEX

ALL_ERRORMSGS

NO_INFOMSGS (default)

TABLOCK

PHYSICAL_ONLY (default)

For more information about these options, see your Microsoft SQL Server documentation.

-apicontext - Short form: none

Use this parameter when calling the cmdlet as an API call.

-PreCommand <String> - Short form: -precmd

This parameter indicates to run a command before the current operation.

-PreCommandPath <String> - Short form: -precmdpath

This parameter specifies the operating system path to the command to be run before the SnapManager operation starts.

-PreCommandArguments <String> - Short form: -precmdargs

Use this parameter to specify a list of strings of SnapManager operation-specific information or user defined arguments to be passed to the program or script.

-PreCommandHost <String> - Short form: -precmdhost

Use this parameter to specify the host machine name on which the command is to be run before the operation starts.

-PreCommandErrors <EnumHandleCmdError[]> - Short form: -precmnderrors

Use this parameter to specify how to handle errors on the precommand and the following SMSQL operation. The `ContinueOnError` value indicates that the following SMSQL operation are to be executed anyway. The `StopOnPreCmdError` value indicates that if a precommand script get an error, the remaining SMSQL operation is not attempted.

-PostCommand - Short form: -postcmd

Use this parameter to run a command after the current operation.

-PostCommandPath <String> - Short form: -postcmdpath

Use this parameter to specify the operating system path to the command to be run after the SMSQL operation starts.

-PostCommandArguments <String> - Short form: -postcmdargs

Use this parameter to specify a list of strings of SnapManager operation-specific information or user-defined arguments to be passed to the program or script.

-PostCommandHost <String> - Short form: -postcmdhost

Use this parameter to specify the name of the host on which the command is to be run after the operation is complete.

-PostCommandErrors <EnumHandleCmdError[]> - Short form: -postcmderrors

Use this parameter to specify how to handle errors on the next postcommand run. The `ContinueOnError` value indicates that the following SMSQL operation is executed anyway. StopOnPostCmdError value indicates that if a postcommand script results in an error, the remaining SMSQL operation is not attempted.

-AvailabilityGroup <String> - Short form: -ag

This parameter specifies the name of the Availability Group to which the databases belong.

-RestoreArchivedBackup - Short Form: -rstarchbkup

Use this parameter to specify using a remote backup to reseed the database.

-SnapVaultSecondary - Short Form: -vaultsec

This optional parameter identifies the backup vault from which you want to reseed a database. If you do not specify this parameter, SnapManager chooses one of the backup vaults. You use this parameter in conjunction with the `-RestoreArchivedBackup` parameter. If you specify this parameter with the `-AvailabilityGroup` parameter, then the Availability Group databases must be spread across the same volumes. Otherwise, do not specify this parameter and SnapManager chooses one of the backup vaults. This parameter applies to clustered Data ONTAP only.

The syntax for this parameter is as follows, where n is the number of Vserver:volume pairs:

```
-SnapVaultSecondary n, Vserver:volume
```

Example: `-SnapVaultSecondary 3, Vserver1:volume1, Vserver2:volume2, Vserver3:volume3`

-IgnoreRepLogs - Short form: -nosharelogs

This parameter specifies that the log backups from the SnapManager Repository Share should be ignored.

-WhatIf - Short form: -wi

This parameter gives you a preview of an operation.

-Confirm - Short form: -cf

This parameter prompts you for confirmation before the actual deletion operation starts.

<CommonParameters>

This cmdlet supports the common parameters `Verbose`, `Debug`, `ErrorAction`, `ErrorVariable`, `WarningAction`, `WarningVariable`, `OutBuffer`, and `OutVariable`. For more information, see about_CommonParameters (http://technet.microsoft.com/library/hh847884.aspx).

Examples

Example 1: `reseed-backup -svr venudhar-2k8vm2 -inst venudhar-2k8vm2 -ag tstag1 -backup sqlsnap__VENUDHAR-2K8VM2_08-26-2012_20.46.42`

This command reseeds Availability Group `tstag1`. Note that only unhealthy databases or databases that are already dropped in the given Availability Group are reseeded.

Example 2: `reseed-backup -svr venudhar-2k8vm2 -inst venudhar-2k8vm2 -d db1,db2,db3 -backup sqlsnap__VENUDHAR-2K8VM2_08-26-2012_20.46.42`

This example reseeds the specific databases db1, db2, and db3.

Example 3: `reseed-backup -svr 'venudhar-2k8vm2' -inst 'venudhar-2k8vm2\heitz' -ag 'testag' -restorelastbackup 0`

This example reseeds all databases that belong to the Availability Group.

Example 4: `reseed-backup -svr 'venudhar-2k8vm2' -inst 'venudhar-2k8vm2' -ag 'testag1' -backup 'sqlsnap__VENUDHAR-2K8VM2_08-26-2012_20.46.42' -RestoreArchivedBackup`

This example reseeds databases that belong to the specified Availability Group from a SnapVault secondary volume chosen by SnapManager. The command does not specify a SnapVault secondary volume because the Availability Group databases are not spread across the same volumes.

restore-backup

Name

restore-backup

Synopsis

This cmdlet enables you to restore databases from SnapManager backups.

Syntax

```
restore-backup [-Server <String>] [-BackupServer <String>] [-UserName
<String>] [-Password <String>] [-ServerInstance <String[]>] -Database
<String[]> [-Backup <String>] [-RestoreLastBackup <Int32>] [-
TransLogsToApply <Int32[]>] [-ForceRestore [<Boolean>]] [-
VerifyServerInstance <String>] [-VerSvrLogin <String>] [-VerSvrPassword
<String>] [-VerDestVolume] [-VerifyOnDestVolumes <String[]>] [-
VerifyDisable] [-DBCCOption <EnumDbccOption[]>] [-TargetDatabase
<String[]>] [-TargetLocation] <String[]>] [-TargetServerInstance
<String[]>] [-PointInTime <String[]>] [-RestoreArchive] [-
RestoreFromUnmanagedMedia] [-SnapInfoDirectory <String>] [-MarkName
<String[]>] [-MarkTime <String[]>] [-RestoreBeforeMark [<Boolean>]] [-
RecoverDatabase <Boolean[]>] [-StandbyPath <String>] [-apicontext] [-
RestoreArchivedBackup] [-SnapVaultSecondary] [-NoAccessToRemoteBackup] [-
ProxyServer <String>] [-PreCommand] [-PreCommandPath <String>] [-
PreCommandArguments <String>] [-PreCommandHost <String>] [-PreCommandErrors
<EnumHandleCmdError[]>] [-PostCommand] [-PostCommandPath <String>] [-
PostCommandArguments <String>] [-PostCommandHost <String>] [-
PostCommandErrors <EnumHandleCmdError[]>] [-AvailabilityGroup] [-
IgnoreRepLogs] [-WhatIf] [-Confirm] [<CommonParameters>]
```

Description

This cmdlet enables you to restore a database. It also gives point-in-time restore, verification, force restore and many other options.

You can also implement these options with the SnapManager user interface.

Parameters

-Server <String> - Short Form: -svr

This parameter denotes the name of the host SQL Server on which the SQL Server instances reside. SnapManager takes the local computer name as the default server name.

Using this parameter, you can also specify a particular SQL server instance. The following examples illustrate the usage:

-BackupServer <String> - Short Form: -bkupsvr

Use this parameter to specify where the backup was originally created. Use the host name or cluster name where the SQL Server instance resides. This parameter cannot be an SQL Server instance name. This parameter is optional, and is mainly used for a restore backup created from a different server. For example, this parameter can be used for DR using SnapMirror. By default, the backup server is the server currently connected, specified by `-Server` parameter. For example:

```
-Server win2k8-248-137 -backupserver 'SQL2K8VI1' -inst 'SQL2K8VI1\DE1' -
TargetServerInstance win2k8-248-137 -SnapInfoDirectory 'H:\SMSQL_SnapInfo'
```

The server is connected to a new server where the restore will be performed. But the backup was originally created on 'SQL2K8VI1', and the instance was 'DE1'. The `-SnapInfoDirectory` parameter is required when you specify this parameter.

-Username <String> - Short Form: -usr

UserName is the SQL Server account name. It is specified if the SQL Server computer is accessed using a different account from that used to access the production SQL Server. If not specified the Windows NT Authentication username will be taken.

-Password <String> - Short Form: -pwd

This parameter is the SQL Server account password. SnapManager ignores this parameter if the parameter -UserName is not specified.

-ServerInstance <String[]> - Short Form: -inst

This parameter specifies the SQL Server instance where the database is backed up originally. SnapManager takes the local computer name as the default server instance.

You can specify multiple server instance names here as a comma-separated list. If multiple databases reside on the same LUN but are owned by different SQL Server instances when you backed them up originally, use the following format:

```
-Inst "SQLServerInstance1","SQLServerInstance2"
```

The first database specified in the -Database parameter refers the first server instance in the -ServerInstance parameter, the second database in the -Database parameter refers to the second server instance in the -ServerInstance parameter and so on.

-Database <String[]> - Short Form: -d

Use this parameter to specify the original database that you want to restore. You can also specify multiple database names only if the databases share a single LUN or multiple LUNs together. In this case, list the databases followed by -Database in the following format:

```
-Database "DatabaseName1"," DatabaseName2"
```

Note: All the databases selected should be present in the selected Snapshot copy. This is a required parameter.

-Backup <String> - Short Form: -bkup

Use this option to specify the name of the backup set. The following example illustrates the usage:

```
-bkup sqlsnap__SYMNASQLDEV170_04-11-2007_15.22.27
```

-RestoreLastBackup <Int32> - Short Form: -lastBkup

Use this parameter to restore backups without specifying the name. If you try to use the Backup and RestoreLastBackup parameters together, SnapManager ignores the RestoreLastBackup parameter and uses the Backup parameter during restore operation. A typical usage example of the restorelastbackup parameter is as follows:

```
restore-backup -restorelastbackup 1 -backup (backup name)
```

Note: If the value for RestoreLastBackup parameter is 0, SnapManager restores the latest backup. If the value is 1, SnapManager restores the second-to-latest backup and so on.

-TransLogsToApply <Int32[]> - Short Form: -translogs

This parameter specifies the list of transactions logs that need to be applied. SnapManager applies all transaction logs of the databases specified in the -Database parameter by default. You can specify the number of transaction logs to be applied for every database mentioned in the -Database parameter. The list of number of transaction logs that have to applied has to be listed in the same sequence as the databases listed in the -Database parameter. For example:

restore-backup -svr MACHINE1\INST1 -database db1,db2 -transLogsToApply 3,7

-ForceRestore [<Boolean>] - Short Form: -force

Use this parameter to force the restore of a database based on its state. SnapManager sets its value to "true" by default.

-VerifyServerInstance <String> - Short Form: -verInst

This parameter specifies the separate SQL Server that is used to run the Database Consistency Check (DBCC) utility. If you have not specified the -verify parameter, SnapManager ignores this parameter.

The following example illustrates the usage:

```
-verInst win-225-161
```

Here the SQL Server instance is the local or remote SQL Server instance to verify on. SnapManager takes the configured SQL Server instance that is used for verify in client configuration (registry) as the default SQL Server instance.

-VerSvrLogin <String> - Short Form: -verlogin

This parameter specifies that SQL Server authentication is used. If not specified, the default Windows NT Authentication mechanism is used.

-VerSvrPassword <String> - Short Form: -verpwd

This parameter is used to input the verification server password. SnapManager ignores this parameter if the parameter -VerSvrLogin is not specified.

-VerDestVolume - Short Form: -verdest

Use this parameter to verify the database on the SnapMirror destination volume. SnapManager sets it to "false" by default.

-VerifyOnDestVolumes <String[]> - Sort form: -verMirror

Specify this parameter to override the default SnapMirror relationships. Enter a comma-separated list of the source storage system, the source volume, the destination storage system, and the destination volume.

-VerifyDisable - Short Form: -verDis

This parameter overrides verification and can disable verification even if the database was not verified after backup.

-DBCCOption <EnumDbccOption[]> - Short form: -dbccopt

This parameter specifies options to the DBCC SQL command that are used to validate and verify the database that is being processed. When you use this parameter, you are explicitly requesting DBCC options, and the system does read the registry to determine the default DBCC options. The security access issues for the registry are bypassed when you use this cmdlet option. The parameter uses the following values:

NOOPTION

NOINDEX

ALL_ERRORMSGS

NO_INFOMSGS (default)

TABLOCK

PHYSICAL_ONLY (default)

For more information about these options, see your Microsoft SQL Server documentation.

-TargetDatabase <String[]> - Short Form: -tgDb

When you want to restore the database with a new name, use this parameter

-TargetLocation - Shot Form: -tgloc

This parameter defines the location to which you want to restore a database.

Syntax: `-TargetLocation Source_Database_Name, n, Logical_FileName_1, Desination_FilePath_1,....,Logical_FileName_n, Desination_FilePath_n`

Where, `Source_Database_Name` represents the source database, `n` represents the number of database files, `Logical_File_1` to `Logical_File_n` represents the database logical file names, `Destination_File_1` to `Destination_File_n` represents the corresponding destination file names for the logical file name, if you want to change the database file name at the target location.

For example, `restore-backup -Database db -TargetDatabase newDB -TargetLocation db,2, DB_FileName, K:\NewDB\NewDB.mdf, LOG_FileName, K:\NewDB\NewDB.ldf`

-TargetServerInstance <String[]> - Short Form: -tgInst

This parameter specifies the name of the new SQL server if you want to restore the database to a new SQL server. SnapManager takes the source SQL server instance as the default.

-PointInTime <String[]> - Short Form: -pit

Use this switch to restore databases until a specific point in time. The format for the point-in-time string is yyyy-mm-ddThh:mm:ss, with time specified in a 24-hour format.

In case of multiple databases you should specify the point-in-time values for every database separated by a comma. The number of values after the parameter name should equal the number of databases selected. The first value will be applied to the first database specified after the -Database parameter, the second value to the second database, and so on. The following example illustrates the usage:

```
-pit 2008-10-22T11:50:00, 2008-11-25T22:50:00
```

> **Note:** The parameter correspondence is one-to-one, that is, the first point-in-time parameter value specified after the parameter -pit is applied to the first database specified in the parameter -Database and the second point-in-time parameter value to second database and so on. The values should conform to the required PointInTime regular expression.

-RestoreArchive - Short Form: -rstarch

Use this parameter to restore a database from an archived backup.

-RestoreFromUnmanagedMedia - Short Form: -rstumm

Use this parameter if you are restoring databases from archived SnapManager backup sets.

-SnapInfoDirectory <String> - Short Form: -snapinfo

Use this parameter to specify the SnapInfo directory path of the archived backup set. Use the parameter only along with the -RestoreFromUnmanagedMedia parameter. This parameter is required when you specify the "-BackupServer" parameter.

-MarkName <String[]> - Short form: -mark

This parameter indicates the marked transaction at which to stop the transaction log recovery.

-MarkTime <String[]> - Short form: -mktm

This parameter specifies a unique timestamp to guarantee the uniqueness of the input restored mark.

-RestoreBeforeMark [<Boolean>] - Short form: -beforemk

This true or false value indicates whether the specified marked transaction log should be included in the restore.

-RecoverDatabase <Boolean[]> - Short Form: -recoverdb

This parameter indicates whether the database fully recovered or left in a partially recovered state after the cmdlet finishes, to facilitate future SQL transaction log restores. This is an array of booleans, so it must match the same number of elements of the -database array. If the it does not match the number of elements of the -database array, an error is given. This defaults to $true for all databases unless the -standbyPath is given, in which case it defaults to $false for all databases.

-StandbyPath <String> - Short Form: -standby

This parameter indicates the path to the standby recovery file where incomplete transactions are stored after restoring a full database and its transaction logs. There is no default if you specify this parameter. The path must be to the standby directory if more than one database shares a LUN. If the database is on a dedicated LUN, then it must be a specific file. If the -standbypath parameter is given, the -RecoveryDatabase given must be -RecoverDatabase $False, otherwise it defaults to $false for all databases if no _RecoverDatabase parameter is specified.

-apicontext - Short form: none

Use this parameter when calling the cmdlet as an API call.

-RestoreArchivedBackup - Short Form: -rstarchbkup

Use this parameter to restore database from an archived backup.

-SnapVaultSecondary - Short Form: -vaultsec

This optional parameter identifies the backup vault from which you want to restore a database. If you do not specify this parameter, SnapManager chooses one of the backup vaults. You use this parameter in conjunction with the `-RestoreArchivedBackup` parameter. If you specify this parameter with the `-AvailabilityGroup` parameter, then the Availability Group databases must be spread across the same volumes. Otherwise, do not specify this parameter and SnapManager will choose one of the backup vaults. This parameter applies to clustered Data ONTAP only.

The syntax for this parameter is as follows:

```
-SnapVaultSecondary n, Vserver:volume
```

Where n is the number of Storage Virtual Machine (SVM, formerly known as Vserver):volume pairs.

Example: `-SnapVaultSecondary 3, Vserver1:volume1, Vserver2:volume2, Vserver3:volume3`

-NoAccessToRemoteBackup - Short Form: -noaccessarchivebkup

This parameter specifies that there is no direct access to the secondary storage system. SnapManager uses the proxy server to access the secondary storage system.

-ProxyServer <String> - Short Form: -pxy

This parameter defines the name of the proxy server. Use it along with the parameter NoAccessToRemoteBackup.

-PreCommand <String> - Short form: -precmd

This parameter indicates to run a command before the current operation.

-PreCommandPath <String> - Short form: -precmdpath

This parameter specifies the operating system path to the command to be run before the SnapManager operation starts.

-PreCommandArguments <String> - Short form: -precmdargs

Use this parameter to specify a list of strings of SnapManager operation-specific information or user defined arguments to be passed to the program or script.

-PreCommandHost <String> - Short form: -precmdhost

Use this parameter to specify the host machine name on which the command is to be run before the operation starts.

-PreCommandErrors <EnumHandleCmdError[]> - Short form: -precmnderrors

Use this parameter to specify how to handle errors on the pre-command and the following SMSQL operation. ContinueOnError value indicates that the following SMSQL operation will be executed anyway. StopOnPreCmdError value indicates that if a pre-command script get an error, the remaining SMSQL operation will not be attempted.

-PostCommand - Short form: -postcmd

Use this parameter to indicate to run a command after the current operation.

-PostCommandPath <String> - Short form: -postcmdpath

Use this parameter to specify the operating system path to the command to be run after the SMSQL operation starts.

-PostCommandArguments <String> - Short form: -postcmdargs

Use this parameter to specify a list of strings of SnapManager operation-specific information or user defined arguments to be passed to the program or script.

-PostCommandHost <String> - Short form: -postcmdhost

Use this parameter to specify the host name on which the command is to be run after the operation is complete.

-PostCommandErrors <EnumHandleCmdError[]> - Short form: -postcmderrors

Use this parameter to specify how to handle errors on the following post-command run. ContinueOnError value indicates that the following SMSQL operation will be executed anyway. StopOnPostCmdError value indicates that if a post-command script get an error, the remaining SMSQL operation will not be attempted.

-AvailabilityGroup <String> - Short Form: -ag

Use this parameter to specify one or more names of Availability Groups for which this backup applies.

-IgnoreRepLogs - Short form: -nosharelogs

Use this parameter to ignore the transaction logbackups from SnapManager Repository Share.

-WhatIf - Short form: -wi

This parameter gives you a preview of an operation.

-Confirm - Short form: -cf

This parameter prompts you for confirmation before the actual deletion operation starts.

<CommonParameters>

This cmdlet supports the common parameters: Verbose, Debug, ErrorAction, ErrorVariable, WarningAction, WarningVariable, OutBuffer and OutVariable. For more information, see about_CommonParameters (http://go.microsoft.com/fwlink/?LinkID=113216).

Examples

Example 1: `restore-backup -Server sql1 -Database "Db1"`

This command restores the backup of database Db1 on SQL Server sql1.

Example 2: `restore-backup -svr 'VM-VS-1' -inst vm-vs-1 -d 'ds_test7' -backup sqlsnap__VM-VS-1_07-18-2008_03.19.14__Daily`

This example restores the specified backup on the given server instance.

Example 3: `restore-backup -inst 'SNAPMGR-65' -Database 'dbDef_1' -restorelastbackup 0 - standbypath u:\temp\standby -recoverdatabase $false`

This example specifies the path where incomplete transactions are stored after restoring a full database and its transaction logs.

Example 4: `restore-backup -Server snapmgr-62 -FederatedGroups 1, snapmgr-62\SQLEXPRESS62, 1, TestData -Mark mypsmark -MarkDesc "mymark description" -Logbkup`

This example restores the log to a marked transaction.

Example 5: `restore-backup -svr 'venudhar-2k8vm2' -inst 'venudhar-2k8vm2\heitz' -ag 'testag' -restorelastbackup 0`

This command restores all the databases belonging to the specified Availabilty group.

Example 6: `restore-backup -svr 'VM-VS-1' -inst vm-vs-1 -d 'ds_test7' -backup sqlsnap__VM-VS-1_07-18-2008_03.19.14__Daily -RestoreArchivedBackup -vaultsec 2,vserver1:volume1,vserver2:volume2`

This example restores the database ds_test7 from the specified SnapVault secondary volume on the specified server instance.

split-clone

Name

split-clone

Synopsis

This cmdlet enables you to split the cloned database from the parent database.

Syntax

`split-clone -Database` <string> [ ] *`Database_name`* [`-Server` <string> *`Server_name`*] [`-UserName` <string> *`User_name`*] [`-Password` <string> *`Password`*] [`-ServerInstance` <string>] [`-GetStatus`] [`-ClusterAware`] [`-apicontext`] [`-WhatIf`] [`-Confirm`] [`<CommonParameters>`]

Parameters

`-Database <String[]>` - Short Form: -d

Specifies the databases that you wanted to split. Use a comma-separated list of strings:

`-d Database 1, Database 2, Database 3, Database 4,....`

Multiple database names should be specified only if those databases share a single LUN or multiple LUNs together. For a multiple database restore, all the selected databases should be present in the selected Snapshot copy.

`-Server <String>` - Short Form: -svr

Denotes the name of the host SQL server on which the SQL server instances reside. SnapManager takes the local computer name as the default server name.

Using this parameter, you can also specify a particular SQL server instance. The following examples illustrate the usage:

`-svr win-225-161`

`-svr sql1`

`-UserName <string>` - Short Form: -usr

Denotes the SQL Server account name. If the login name is not specified, SnapManager uses Windows NT Authentication.

`-Password <string>` - Short Form: -pwd

Specifies the SQL Server account password. SnapManager ignores this parameter if the parameter `-UserName` is not specified.

`-ServerInstance <string>` - Short Form: -inst

Specifies the SQL Server instance where the database is backed up originally. SnapManager takes the local computer name as the default server instance. If multiple databases reside on the same LUN but are owned by different SQL server instances when you backed them up originally, you should use the following format:

```
-Inst "SQLServerInstance1","SQLServerInstance2"
```

The first database specified in the `-Database` parameter refers the first server instance in the `-ServerInstance` parameter, the second database in the `-Database` parameter refers to the second server instance in the `-ServerInstance` parameter and so on.

`-GetStatus` - Short Form: -gtst

Enables you to view the status of the split job. The split process might take time based on the size and load on the volume that is being split. You can check the status by using the `-GetStatus` option in the cmdlet. This provides you with the status of each of the volumes being split. Each mount point that has unique volumes is shown in the status; other mount points or disks from the same volume are ignored, because the split occurs at the volume level.

> **Note:** You are not able to see the status of the split operation through the graphical user interface (GUI).

`-ClusterAware` - Short Form: -cl

Specifies that the cmdlet runs solely on the active node in a cluster environment.

`-apicontext` - Short Form: none

Calls the cmdlet as an API call.

`-WhatIf` - Short Form: -wi

Provides you with a preview of an operation.

`-Confirm` - Short Form: -cf

Prompts you for confirmation before the actual operation starts.

`<CommonParameters>`

Supports the common parameters:Verbose, Debug, ErrorAction, ErrorVariable, WarningAction, WarningVariable, OutBuffer, and OutVariable. For more information, see about_CommonParameters (*http://go.microsoft.com/fwlink/?LinkID=113216*).

Example

```
split-clone -Server snapmgr-63 -Database 'DB1' -getstatus
```

verify-backup

Name

verify-backup

Synopsis

This cmdlet enables you to verify the SQL Server databases in SnapManager PowerShell command-line interface.

Syntax

```
verify-backup [-Server <String>] [-UserName <String>] [-Password <String>]
[-Database <String[]>] [-VerifyServerInstance <String>] [-VerSvrLogin
<String>] [-AttachDB] [-VerSvrPassword <String>] [-UpdateMirror] [-
VerDestVolume] [-VerifyOnDestVolumes <String[]>] [-ManagementGroup
<String>] [-BackupNo <Int32>] [-MountPointDir <String>] [-UseMountPoint] [-
UseDriveAvailable] [-Command] [-RunCommand <String>] [-CommandArguments
<String>] [-CommandServer <String>] [-PreCommand] [-PreCommandPath
<String>] [-PreCommandArguments <String>] [-PreCommandHost <String>] [-
PreCommandErrors <EnumHandleCmdError[]>] [-PostCommand] [-PostCommandPath
<String>] [-PostCommandArguments <String>] [-PostCommandHost <String>] [-
PostCommandErrors <EnumHandleCmdError[]>] [-DBCCOption <EnumDbccOption[]>]
[-apicontext] [-ArchiveBackup] [-VerifyArchiveBackup] [-
ArchivedBackupRetention <String>] [-ClusterAware] [-WhatIf] [-Confirm] [-
AvailabilityGroup] [-BackupPriority] [-Primary] [-Secondary] [-
PreferredBackupReplica] [<CommonParameters>]
```

Description

This cmdlet enables you to perform verification operations. You can mount the Snapshot copies, manage SnapMirror relationships and destinations, assign management groups for verification and so on.

You can also implement these options with the SnapManager user interface.

Parameters

-Server <String> - Short Form: -svr

This parameter denotes the name of the host SQL Server on which the SQL Server instances reside. SnapManager takes the local computer name as the default server name. If no default host exists, SnapManager attempts to use the following as the default:

- The VerifyServerInstance specified by the user
- The configured verification server for the current machine (in the registry) done in the configuration wizard, or backup verification settings
- The VerificationServerInstance from the SQL Server being backed up as the verification server
- The current machine

Using this parameter, you can also specify a particular SQL Server instance. The following examples illustrate the usage:

```
-svr win-225-161
```

```
-svr sql1
```

-Username <String> - Short Form: -usr

UserName is the SQL Server account name. It is specified if the SQL Server computer is accessed using a different account from that used to access the production SQL Server. If not specified the Windows NT Authentication username will be taken.

-Password <String> - Short Form: -pwd

This parameter is the SQL Server account password. SnapManager ignores this parameter if the parameter -UserName is not specified.

-Database <String[]> - Short Form: -d

Short Form: -d This parameter specifies the original databases to backup. If you specify multiple database names, the list is separated by commas. If you do not specify the database parameter, the cmdlet backs up all SQL server instances that are peer instances of the SQL server in the -Server parameter.

-VerifyServerInstance <String> - Short Form: -verInst

This parameter specifies the separate SQL Server that is used to run the Database Consistency Check (DBCC) utility. If you have not specified the -verify parameter, SnapManager ignores this parameter.

The following example illustrates the usage:

```
-verInst win-225-161
```

Here the SQL Server instance is the local or remote SQL Server instance to verify on. SnapManager takes the configured SQL Server instance that is used for verify in client configuration (registry) as the default SQL Server instance.

-VerSvrLogin <String> - Short Form: -verlogin

This parameter specifies that SQL Server authentication is used. If not specified, the default Windows NT Authentication mechanism is used.

-AttachDB - Short Form: -attdb

If the operation includes a database or transaction log verification, use this option when you want to specify that the databases are to be attached after the verification operation completes.

-VerSvrPassword <String> - Short Form: -verpwd

This parameter is used to input the verification server password. SnapManager ignores this parameter if the parameter -VerSvrLogin is not specified.

-UpdateMirror - Short Form: -updmir

Use this option to update the SnapMirror destination after the backup or verification operations are complete, if you are using backups that reside on volumes configured as SnapMirror sources.

-VerDestVolume - Short Form: -verdest

Use this parameter to verify the database on the SnapMirror destination volume. SnapManager sets it to false by default.

-VerifyOnDestVolumes <String[]> - Short form: -vermirror

Specify this parameter to override the default SnapMirror relationships. Enter a comma-separated list of the source storage system, the source volume, the destination storage system, and the destination volume.

-ManagementGroup <String> - Short form: -mgmt

This parameter denotes the backup or verify operation that SnapManager performs on daily, or weekly, or standard basis. The default management group is standard.

-BackupNo <Int32> - Short Form: -bkno

This option specifies the number of most recent unverified backups to verify. It is an integer with a default value of 1.

-MountPointDir <String> - Short Form: -mpdir

Use this parameter to specify the mount point directory on which a backup set is mounted during database verification. This parameter should be used along with the parameter -UseMountPoint.

Note: This option is valid only if you specify the parameter -BkupSIF.

-UseMountPoint - Short Form: -mp

This parameter specifies that the Snapshot copy must be mounted to an NTFS directory. During a SnapManager verification operation, Snapshot copies are mounted to the default NTFS directory for database verification. The option is effective when there are no available drive letters to mount the Snapshot copies. It overrides preconfigured SnapManager verification settings.

-UseDriveAvailable - Short Form: -drvavail

Use this parameter to specify the mount point with available drive on which a backup set will be mounted during database verification.

-Command - Short form: -cmd

This switch parameter that runs a command after the backup or verify operation.

-RunCommand - Short form: -runcmd

This parameter runs the specified command after the SnapManager backup or verification operation is complete. It defines the complete path for the command to be run after the backup or verify operation is complete. There is no default.

-CommandArguments <String> - Short form: -cmdargs

This option contains the string of SnapManager operation-specific information to be passed to your program or script. It is considered only if Command and RunCommand are specified. There is no default.

-CommandServer <String> - Short form: -cmdsvr

This obsolete parameter (now replaced by PostCommandHost) was used to indicate the machine where the desired command should run after the operation is complete. The default was to run on the current machine. This was only considered if -command and -RunCommand were specified.

-PreCommand <String> - Short form: -precmd

This parameter indicates to run a command before the current operation.

-PreCommandPath <String> - Short form: -precmdpath

This parameter specifies the operating system path to the command to be run before the SnapManager operation starts.

-PreCommandArguments <String> - Short form: -precmdargs

Use this parameter to specify a list of strings of SnapManager operation-specific information or user defined arguments to be passed to the program or script.

-PreCommandHost <String> - Short form: -precmdhost

Use this parameter to specify the host machine name on which the command is to be run before the operation starts.

-PreCommandErrors <EnumHandleCmdError[]> - Short form: -precmnderrors

Use this parameter to specify how to handle errors on the pre-command and the following SMSQL operation. ContinueOnError value indicates that the following SMSQL operation will be executed anyway. StopOnPreCmdError value indicates that if a pre-command script get an error, the remaining SMSQL operation will not be attempted.

-PostCommand - Short form: -postcmd

Use this parameter to indicate to run a command after the current operation.

-PostCommandPath <String> - Short form: -postcmdpath

Use this parameter to specify the operating system path to the command to be run after the SMSQL operation starts.

-PostCommandArguments <String> - Short form: -postcmdargs

Use this parameter to specify a list of strings of SnapManager operation-specific information or user defined arguments to be passed to the program or script.

-PostCommandHost <String> - Short form: -postcmdhost

Use this parameter to specify the host name on which the command is to be run after the operation is complete.

-PostCommandErrors <EnumHandleCmdError[]> - Short form: -postcmderrors

Use this parameter to specify how to handle errors on the following post-command run. ContinueOnError value indicates that the following SMSQL operation will be executed anyway. StopOnPostCmdError value indicates that if a post-command script get an error, the remaining SMSQL operation will not be attempted.

-DBCCOption <EnumDbccOption[]> - Short form: -dbccopt

This parameter specifies options to the DBCC SQL command that are used to validate and verify the database that is being processed. When you use this parameter, you are explicitly requesting DBCC options, and the system does read the registry to determine the default DBCC options. The security access issues for the registry are bypassed when you use this cmdlet option. The parameter uses the following values:

NOOPTION

NOINDEX

ALL_ERRORMSGS

NO_INFOMSGS (default)

TABLOCK

PHYSICAL_ONLY (default)

For more information about these options, see your Microsoft SQL Server documentation.

-apicontext - Short form: none

Use this parameter when calling the cmdlet as an API call.

-ArchiveBackup - Short form: -arch

Use this parameter to archive database to a secondary storage system.

-VerifyArchiveBackup - Short form: -verarch

Use this parameter to verify database archived at the secondary storage system.

-ArchivedBackupRetention <String> - Short form: -archret

Use this parameter to specify whether you want to retain backups at the archived location on a daily, hourly, weekly, monthly, or unlimited basis.

-ClusterAware - Short form: cl

Use this parameter to specify that the cmdlet runs solely on the active node in a cluster environment.

-WhatIf - Short form: -wi

This parameter gives you a preview of an operation.

-Confirm - Short form: -cf

This parameter prompts you for confirmation before the actual deletion operation starts.

-AvailabilityGroup <String> - Short Form: -ag

Use this parameter to specify one or more names of Availability Groups for which this backup applies.

-BackupPriority <Integer,Integer> - Short Form: -bp

Use this parameter to specify a set of secondary Availability Groups on a cluster by specifying a range of backup priorities. The operation applies to all replicas with backup priorities in that range. The maximum priority must be in the range of 1 to 100. The minimum backup priority must be less than or equal to the maximum priority.

-Primary - Short form: -prm

If this parameter is defined, then the backup is only taken on the primary replica. If BackupPriority is also defined, then the primary replica must also satisfiy the BackupPriority values.

-Secondary - Short form: -sec

If this parameter is defined, then the backup is taken on all secondary replicas. If BackupPriority is also defined, then the secondary replicas must also satisfiy the BackupPriority values.

-PreferredBackupReplica - Short form: -preferbkreplica

Use this parameter to specify that only the preferred backup replica is verified. The preferred backup replica is set from the Availability Group properties in the SQL Server 2012 Management Studio.

<CommonParameters>

This cmdlet supports the common parameters: Verbose, Debug, ErrorAction, ErrorVariable, WarningAction, WarningVariable, OutBuffer and OutVariable. For more information, see about_CommonParameters (http://go.microsoft.com/fwlink/?LinkID=113216).

Examples

Example 1: `verify-backup -svr 'VM-VS-1' -d 'VM-VS-1', '2', 'ds_test6', 'ds_test7' -verInst 'ZEUS-VM1\VERSERVER' -bkno 1 -mgmt standard -ArchiveBackup -VerifyArchiveBackup -ArchivedBackupRetention Daily`

This command initiates deferred verification for the specified database at the specified server, with one unverified most recent backup. The management groups are standard.

Copyright information

Copyright © 1994–2015 NetApp, Inc. All rights reserved. Printed in the U.S.

No part of this document covered by copyright may be reproduced in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system—without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.277-7103 (October 1988) and FAR 52-227-19 (June 1987).

Trademark information

NetApp, the NetApp logo, Go Further, Faster, ASUP, AutoSupport, Campaign Express, Cloud ONTAP, clustered Data ONTAP, Customer Fitness, Data ONTAP, DataMotion, Fitness, Flash Accel, Flash Cache, Flash Pool, FlashRay, FlexArray, FlexCache, FlexClone, FlexPod, FlexScale, FlexShare, FlexVol, FPolicy, GetSuccessful, LockVault, Manage ONTAP, Mars, MetroCluster, MultiStore, NetApp Insight, OnCommand, ONTAP, ONTAPI, RAID DP, SANtricity, SecureShare, Simplicity, Simulate ONTAP, Snap Creator, SnapCopy, SnapDrive, SnapIntegrator, SnapLock, SnapManager, SnapMirror, SnapMover, SnapProtect, SnapRestore, Snapshot, SnapValidator, SnapVault, StorageGRID, Tech OnTap, Unbound Cloud, and WAFL are trademarks or registered trademarks of NetApp, Inc., in the United States, and/or other countries. A current list of NetApp trademarks is available on the web at *http://www.netapp.com/us/legal/netapptmlist.aspx*.

Cisco and the Cisco logo are trademarks of Cisco in the U.S. and other countries. All other brands or products are trademarks or registered trademarks of their respective holders and should be treated as such.

How to send comments about documentation and receive update notification

You can help us to improve the quality of our documentation by sending us your feedback. You can receive automatic notification when production-level (GA/FCS) documentation is initially released or important changes are made to existing production-level documents.

If you have suggestions for improving this document, send us your comments by email to *doccomments@netapp.com*. To help us direct your comments to the correct division, include in the subject line the product name, version, and operating system.

If you want to be notified automatically when production-level documentation is released or important changes are made to existing production-level documents, follow Twitter account @NetAppDoc.

You can also contact us in the following ways:

- NetApp, Inc., 495 East Java Drive, Sunnyvale, CA 94089 U.S.
- Telephone: +1 (408) 822-6000
- Fax: +1 (408) 822-4501
- Support telephone: +1 (888) 463-8277

Index

.TRN files
    changing to .TRB *39*

A

application
    settings *85*
application settings
    restriction for configuring
        restriction for changing *85*
archived backup
    cloning databases from *57*
AutoSupport
    enabling *89*
Availability Group
    creating a clone replica *60*
Availability Group transaction log backup *29*

B

backing up databases
    for the first time *18*
    using a schedule *21*
backup
    cloning databases from *57*
backup deletion
    automatic setting for *26*
backup management group
    changing *29*
backup operations
    defining settings for *85*
backup set
    changing backup management group for *29*
backup sets
    deleting using SnapManager *27*
    how SnapManager deletes them *26*
    verifying the initial set *20*
    verifying with a schedule *24*
backup types
    overview of *15*
backups
    how SnapManager backs up
        stream-based method
            online Snapshot copy method *9*
    using the Find Backups wizard *41*
benefits and features
    overview of *6*
busy Snapshot
    deleting *30*
busy Snapshot copy
    avoiding during a SnapManager operation *30*
    deleting *30*

C

centralized transaction log backups
    setting up a network location *70*
    using SnapManager share *70*
clone replica
    creating *60*
cloned databases
    cloning *60*
    deleting *63*
    splitting *62*
cloning databases
    limitations for VMDKs *54*
    overview of *54*
    requirements for VMDKs *13*, *55*
cmdlets
    accessing *108*
    common parameters *108*
    guidelines *108*
    new-backup *146*
    split-clone *169*
commands
    arguments for preoperation and postoperation *90*
    reseed-backup *156*
    setting defaults for preoperation and postoperation *89*
comments
    how to send feedback about documentation *180*
Configuration wizard
    moving multiple SnapInfo directories to a single directory *68*
configuring
    email notifications
        configuring *65*
    fractional space reservation policies *98*
control file
    exporting *71*
    importing *71*

sample XML schema *72*
custom policies
configuring for fractional space-reserved LUNs *98*

D

database states
non-operational *40*
operational *40*
read-only *40*
database verification
modifying settings for *86*
databases
backing up for the first time *18*
backing up with a schedule *21*
cloning already cloned database *60*
cloning from local backup *57*, *60*
cloning from production database *59*
cloning, overview of *54*
configuring restore settings *87*
deleting clones *63*
destinations for a restore operation *38*
exporting to a control file *71*
how restoring works *34*
how SnapManager backs them up *9*
importing using a control file *71*
maximum number of *100*
migrating back to local disks *69*
modifying backup settings *85*
overview of backing up *8*
post-restore states *40*
removing layout restrictions *95*
requirements for restoring *40*
restoring from a remote backup set *44*
sources for a restore operation *38*
splitting cloned database *62*
strategy for backing up *15*
types of restore operations *36*
verifying the initial backup set *20*
verifying with a schedule *24*
DBCC
as used by SnapManager Backup *10*
verifying a backup, drive letters required for *10*
DBCC options
choosing for database verification *86*
disk requirements
in a Windows clustered environment *102*
documentation
how to receive automatic notification of changes to *180*
how to send feedback about *180*

E

email notifications
enabling *89*

F

features and benefits
overview of *6*
feedback
how to send comments about documentation *180*
file groups
maximum number of *100*
Find Backups wizard
restoring backups *41*
fractional space reservation
viewing status of *96*
fractional space-reserved LUNs
configuring policies for *98*
full database backup
what to do if the backup fails *30*

I

information
how to send feedback about improving documentation *180*

L

local backup sets
restoring SQL Server databases from *41*
local disks
migrating databases back to *69*
logging
enabling *89*
LUNs
maximum number of *100*
removing layout restrictions for *95*

M

management groups
overview of *15*
managing transaction log backups *29*
migration
migrating databases back to local disks *69*

moving multiple SnapInfo directories to a single directory *68*
modifying
registry keys on the primary server *51*
multiple instance cluster
disk requirements *102*

N

naming conventions
overview of *15*
new-backup cmdlet
syntax and description *146*

P

postoperation commands
arguments for *90*
setting defaults for *89*
PowerShell cmdlets
accessing *108*
common parameters
guidelines *108*
preoperation commands
arguments for *90*
setting defaults for *89*
preparing secondary site before *51*
primary server
modifying for primary server *51*
primary site
preparing for recovery *51*
product overview
features and benefits *6*
production databases
cloning *59*

R

recovering databases
from secondary site *52*
recovering databases on VMDKs
using SnapMirror *51*
Recovery Point Objective
meeting *22*
registry keys
modifying on primary server *51*
reinstalling
SnapManager *106*
remote verification server
DBCC, drive letters required for *10*
repairing
SnapManager *106*
report directory
changing location of *66*
reports
default location of *66*
deleting *64*
overview *64*
viewing *64*
reseed-backup command
syntax and description *156*
restore operation *51*
restoring
replicated, publisher, and subscriber databases using SnapManager *46*
restoring databases
configuring default settings *87*
destinations for a restore operation *38*
from a remote backup set *44*
how it works *34*
post-restore states *40*
requirements *40*
sources for a restore operation *38*
using SnapMirror *48*
using the Restore wizard *44*
restoring operations
types of restore operations *36*
restoring SQL Server databases
from a local backup set *41*
using SnapManager for SQL *48*
restoring using SnapMirror *48*

S

secondary site
preparing for recovery *51*
service accounts
requirements for 7-Mode SnapVault *101*
single-instance cluster
disk requirements *102*
SnapInfo directories
moving to a single directory *68*
SnapInfo directory
how SnapManager updates it *10*
what it contains *10*
what the subdirectories contain *10*
SnapManager
user interface
command-line interface (CLI)
clone-backup *109*

SnapManager for Microsoft SQL Server
backup overview *8*
backup strategy *15*
features and benefits *6*
reinstalling *106*
repairing *106*
reports *64*
supported configurations *100*
uninstalling *107*
upgrading
interactively *104*
prerequisites *104*
upgrading silently *104*
SnapManager PowerShell
accessing *108*
common parameters *108*
SnapMirror
requirements for VMDK cloning *13*, *55*
requirements for VMDK verification *13*, *55*
using to restore databases *48*
SnapMirror destination volume
choosing for database verification *86*
Snapshot copies per volume *25*
SnapVault
service account requirements for 7-Mode *101*
SnapVault destination volume
choosing for database verification *86*
space consumptionvolumes
viewing available space *96*
viewing status of *96*
split-clone cmdlet
description and parameters *169*
SQL Server database
Auto Shrink option *30*
SQL Server databases
restoring from a local backup set *41*
restoring using archives *48*
SQL Server instances
maximum number of *100*
SQL Server Management Studio
restoring transaction logs from *39*
storage layouts
allowing on any LUN or VMDK configuration *95*
suggestions
how to send feedback about documentation *180*

T

transaction log backup
what to do if the backup fails *30*
transaction log backups
centralizing *70*
managing *29*
using a schedule *22*
using SnapManager *70*
transaction logs
backing up with a schedule *22*
restoring from SQL Server Management Studio backups *39*
troubleshooting
reinstalling SnapManager *106*
repairing SnapManager *106*
twitter
how to receive automatic notification of documentation changes *180*

U

uninstalling
SnapManager for Microsoft SQL Server *107*
upgrading SnapManager
interactively *104*
prerequisites *104*
silently *104*

V

verification
modifying settings for *86*
verification server
choosing *86*
verifying backup sets
requirements for VMDKs *13*, *55*
VMDKs
cloning limitations of *54*
maximum number of *100*
removing layout restrictions for *95*
requirements for cloning from destination volumes *13*, *55*
requirements for verification from destination volumes *13*, *55*
volumes
maximum number of *100*
Vservers
See SVMs

W

Windows clustered environment
disk requirements *102*

Windows host system requirements
    DBCC, drive letters required for *10*

X

XML scheme
    sample of *72*