Destiny® system backups

Establishing a backup and restore plan for Destiny

Overview

It is important to establish a backup and restore plan for your Destiny installation. The plan must be validated and monitored to ensure that your data is sufficiently backed up and can be recovered in the event of hardware failure or other disaster.

IMPORTANT Follett recommends deploying a comprehensive backup solution and testing and monitoring all backup processes.

There are tradeoff decisions to be made in the backup strategy that will be shaped by your organization's risk tolerance, technology constraints, and ability to absorb data loss or downtime. This document provides an overview of standard backup and restore for the Destiny system.

IMPORTANT This content does not cover high-availability configurations such as clustered servers or log shipping. Please contact Follett School Solutions, Inc. Technical Support should you have any questions about backing up your Destiny installation. For details, see Destiny® system backups.

Backing up Destiny: an overview

SQL database

The heart of the Destiny data resides in the SQL database. These are the main components of SQL data to be backed up:

- The *Destiny* SQL database. This database contains the main Destiny data. Proper SQL backup configuration of this database is essential.

  NOTE If your installation is a consortium, you will have to ensure a proper backup of multiple SQL databases (one per member).

- The Destiny SQL transaction log.
- The *Master* SQL database. This system database is useful when restoring to a replacement server.

It is not necessary to back up the *tempdb* database. This is a SQL Server work area and is recreated automatically.

IMPORTANT Follett recommends backing up the SQL data on a daily basis. Additionally you might want to implement the full recovery model, explained below, to safeguard transactions that occur between backups.

Application folders

The remaining Destiny content is located in the Destiny application folders. This content includes the application, configuration files, application logs, job and report output, export files, images (book covers, patron pictures, etc.), custom scripting, visual search configuration, and other files. This content should be backed up to ensure full recovery of the Destiny system.

IMPORTANT Follett recommends backing up the application folder daily.

The application folders and their content are located below the `\FSC-Destiny` folder on the application server. If backups are to occur while the Destiny application is running, this data must be backed up using *open file* backup software. If this is not an option, the Destiny service must be stopped while the backup occurs. The batch files `StopDestinyService.bat` and `RunDestinyService.bat` can be run via the Windows task scheduler to stop Destiny service prior to, and restart the service after, the backup. These files are located in the `\FSC-Destiny\fsc\bin` folder.

Some installations prior to Destiny 7.0 were configured with the `osql_backup.bat` file. This script was provided to the customer as an introductory means to backup Destiny data and is not intended as a comprehensive backup solution. As a rudimentary tool, it will extract the SQL data but does not include complete backup services such as support for remote or removable media, failure notification, and backup file management. It is highly recommended to implement a robust backup solution that includes these important best practices.

Restoring Destiny

Conceptually, restoring the Destiny application involves the following:

- Restore and verify the SQL database as described below
- Restore the `\FSC-Destiny application` folder and subfolders
- If restoring to a different server environment, update the Destiny configuration, register the service and verify database connectivity

Please contact Follett School Solutions, Inc. Technical Support for specific assistance should you need to restore your Destiny system.

SQL Server backup concepts

An understanding of the following concepts and terms is helpful when backing up the SQL database portion of Destiny data. The terms are associated with Microsoft SQL Server 2005, Microsoft SQL Server 2008, and Microsoft SQL Server 2008 R2, as well as SQL Express 2005, SQL Express 2008 and SQL Express 2008 R2 for smaller installations.

For more information and complete technical details on SQL Server backup and restore, see Microsoft SQL Books Online, as follows:

- Microsoft SQL Server 2005 - http://msdn.microsoft.com/en-us/library/ms187048(SQL.90).aspx
- Microsoft SQL Server 2008 - http://msdn.microsoft.com/en-us/library/ms187048(v=SQL.100).aspx
- Microsoft SQL Server 2008 R2 - http://msdn.microsoft.com/en-us/library/ms187048(v=SQL.105).aspx

Live backups vs. data files

A SQL database can be backed up with a SQL Server backup agent while the database and application are running. This is generally the safest, most comprehensive, and least disruptive method. Alternatively, the external data files can be backed up if the database and application are stopped. The information below applies to live SQL agent backups.

Full vs. differential database backups

Database backups can be *full* or *differential*. Each full database backup contains the whole database. Using full database backups is simple because there is only one element in the backup/restore process.

Full database backups are appropriate when:

- Backup media space is not an issue
- Backups do not take too much time
- The easiest restore process is desired

A differential backup contains only changes since the last full backup. This faster, smaller backup is designed to make it possible to take backups more frequently when media space is limited or backup times become too long.

Restoring from a differential database backup consists of restoring the most recent full backup followed by the single most recent differential backup.

Simple recovery model

The simple recovery model refers to backing up just the database, using full and optionally differential database backups as described above. Database recovery is to the point of most recent backup. This is the simplest recovery model to manage because only the database is involved. Transactions subsequent to a backup are not logged or backed up until the next database backup.

Because transactions are not logged, changes since the last database backup are lost when restoring from backup. The simple recovery model therefore has the greatest exposure to potential data loss.

The simple recovery model is appropriate to use when:

- The backup/restore process must be simple
- The database is relatively small, allowing for backups that are frequent enough for your organization
- There are not extensive data modifications between backups
- You can accept the loss of changes since the last backup

Restoring a SQL database in the simple recovery model

Restoring a SQL database in the simple recovery model involves the following steps.

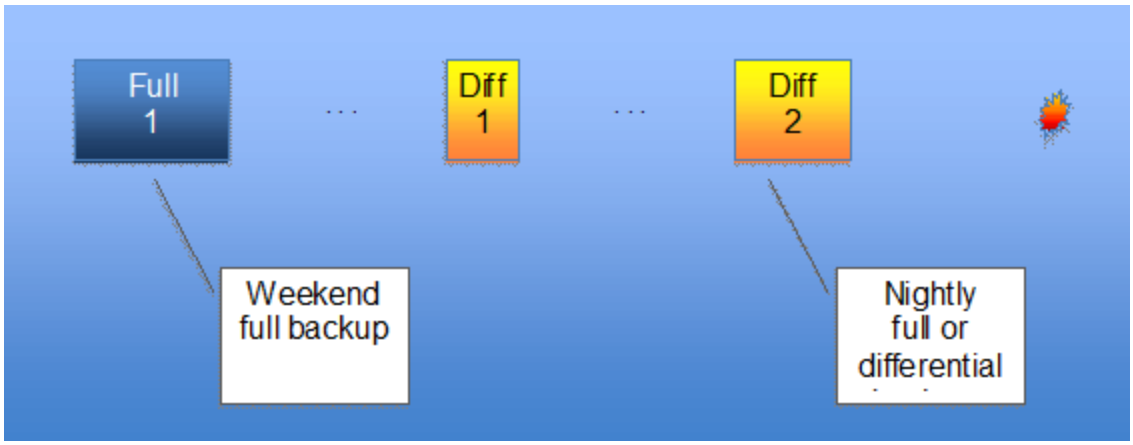
IMPORTANT If you are restoring to a new SQL Server installation, you need to restore and recover the *master* database.

To restore a database that is configured under the simple recovery model:

- Restore the most recent full database backup
- Restore the most recent differential database backup, if any
- Recover the database – Make it ready for use

Please contact Follett School Solutions, Inc. Technical Support for specific assistance you need to restore your Destiny system.

This illustration represents a schedule with full weekly backups and nightly differential backups:



To restore this SQL database

1. Restore in order:

- The latest full backup (Full 1)
- The latest differential backup, if any (Diff 2)

2. Recover the database. This finalizes the restore and makes the database ready for use.

Full recovery model

The full recovery model provides protection that is more complete. Database changes are logged to the transaction log. The transaction log is periodically backed up, allowing for full recovery of data to any point from the backup up to the time of failure. The tradeoff is that the backup and restore process is more complex.

The full recovery model is appropriate to use when:

- You need to ensure comprehensive data recovery up to the point of failure
- You cannot afford to lose transactions that occur between database backups
- You are willing to manage the additional complexity in the backup configuration and data recovery process

Transaction log

The transaction log is the key component in the full recovery model. This file, named `*.LDF`, saves transactions between database backups. When either a database backup or transaction log backup occurs, accumulated transactions in the log are emptied (truncated) and logging resumes.

Frequent transaction log backups provide recoverability if the transactions log is damaged. Because each log backup contains only new transactions since the last log backup, the series of all log backups after the last database backup—the *log chain*—must be applied to the database backup during recovery.

IMPORTANT Follett recommends hourly log backups while transactions are occurring.

Transaction log size settings

Frequent backups of the transaction log are absolutely essential to ensure that the transaction log does not grow too large. If the transaction log grows so large that it fills available disk space, you will not be able to perform transactions on your Destiny database. Additionally, SQL Server may mark the database as suspect. If you encounter this situation, see the Microsoft KnowledgeBase article, *A transaction log grows unexpectedly or becomes full on a computer that is running SQL Server*, at http://support.microsoft.com/kb/317375.

It is possible, but rare, for a transaction log to grow large even if there are frequent backups. This can occur in situations where there are extremely large or uncommitted transactions. If SQL Server is unable to grow the transaction log and/or reuse space in the log file, you can determine the cause by seeing the *log_reuse_wait* and *log_reuse_wait_desc* columns of the `sys.databases` catalog in SQL Server 2005, SQL Server 2008, or SQL Server 2008 R2.

When a backup occurs, the transaction log is truncated. SQL Server retains and reuses the transaction log space rather than shrinking the external LDF file. Transaction log auto-growth can be specified to grow by a percentage or by megabytes. This setting is database-specific and can be accessed in SQL Server Enterprise Manager or Management Studio. Because SQL Server leaves the file at its own high-water mark, it is generally acceptable to have the transaction log grow in absolute megabytes. Monitor the log file size on disk periodically to ensure backups are preventing nonstop growth. Regular backups will keep this file to a fraction of the size of your database.

Shrinking the transaction log file is not a part of routine database administration. It takes resources to grow the log file, so it is best left at its own high-water mark to avoid having to resize during high transaction activity. However, if you have had an unusual event that caused large growth, such as if backups failed to run for a period of time, then there is a mechanism in SQL Server to shrink the transaction log. For details on shrinking the log file, see http://support.microsoft.com/kb/907511.

Further transaction log growth information can be found in SQL Server Books Online and in these Microsoft KnowledgeBase articles:

- Books Online *Backing up and Restoring Databases in SQL Server*:
  - Microsoft SQL Server 2005 - http://msdn.microsoft.com/en-us/library/ms187048(SQL.90).aspx
  - Microsoft SQL Server 2008 - http://msdn.microsoft.com/en-us/library/ms187048(v=SQL.100).aspx
  - Microsoft SQL Server 2008 R2 - http://msdn.microsoft.com/en-us/library/ms187048(v=SQL.105).aspx
- To understand why the transaction log might fill up - http://support.microsoft.com/default.aspx/kb/110139.
- For information about full or unexpectedly growing log files - http://support.microsoft.com/default.aspx/kb/317375.
- For information about preventing unexpected log file growth - http://support.microsoft.com/default.aspx/kb/873235 .

Tail log backup

In the event of a failure, the current transaction log must be backed up to capture transactions logged since the last log backup. SQL Server introduces the term *tail log backup* to describe this step. The tail log backup is taken before any backups are restored, and is restored as the last item in the log chain.

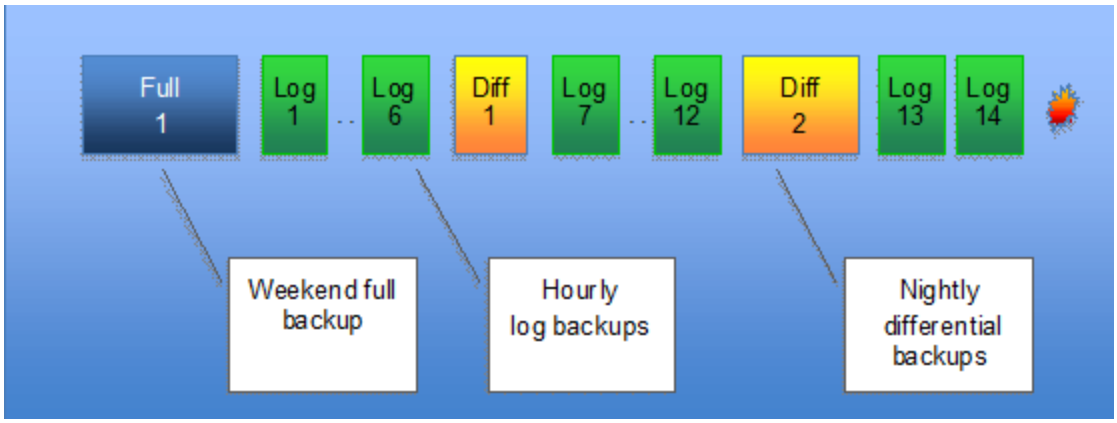
Where possible, it is recommended to keep the transaction log file on a separate physical drive from the database data file(s) so that a tail log backup can still be taken if the database drive fails.

Restoring an SQL database in the full recovery model

To restore a database that is configured under the full recovery model:

1. Perform a tail log backup to capture log entries since last log backup.
2. Restore the most recent full database backup.
3. Restore the most recent differential database backup, if any.
4. Restore the series of transaction log backups since the most recent database backup (full or differential).
5. Recover the database – Make it ready for use.

The following backup sequence represents a schedule with weekly full backups, nightly differential backups, and hourly transaction log backups.



NOTE Diff 2 already contains the transactions in Logs 1-12. Diff 1 would not be included when restoring this database.

To restore this SQL database

1. Create a tail log backup to backup transactions since the Log 14 backup occurred.
2. Restore in order:
   - The latest full backup (Full 1)
   - The latest differential backup, if any (Diff 2)
   - All log backups since the last database backup, in order:
     - Log 13
     - Log 14
   - The tail log backup

3. Recover the database. This finalizes the restore and makes the database ready for use.

It would have been possible to rely solely on weekly full backups plus hourly transaction log backups, skipping the nightly differential backups. However, restoring the database would involve a larger number of log backups and would significantly increase the total restoration time.

Backup managers

Third-party backup manager software can be used to help simplify and manage the configuration and execution of the entire backup/restore process. Examples include Symantec Backup Exec™, CA Arcserve® Backup, or EMC® Retrospect®. It is also possible to configure a SQL backup schedule through SQL Server's provided management tools such as the SQL Server Management Studio Suite.

Consult your chosen tool's documentation for details of implementing the options described above.

IMPORTANT Follett recommends: using a third party backup solution that includes a SQL Server agent for the database and open file management for the application folder and backing up to removable media or to a remote network location. Backup rotations should contain at least one full backup weekly and the subsequent daily backups may be full or differential. Follett advises against the use of drive imaging software as a backup mechanism for the Destiny system.

Site/small district SQL Express particulars

Destiny installations based on SQL Express are preset to the simple recovery model by default. Transactions will not be logged to the log file.

NOTE Database backups to a remote location or external media must still be configured and scheduled in the customer environment.

High Availability options

Failover It is possible to configure SQL Server using two clustered servers such that if one server fails, the system will automatically failover to the other server. See SQL Server Books Online if you wish to explore this advanced technique further.

Log shipping *Log shipping* refers to a process that copies logged transactions to a *warm* mirror database. See SQL Server Books Online if you wish to explore this advanced technique further.

Other SQL Server backup terms

The following terms are related to the SQL Server backup process but are not necessarily applicable to Destiny backups. They are included here for completeness in case these terms are encountered elsewhere.

Partial backups A partial backup backs up the primary file and all read/write files, plus any specified read-only files or file groups. It is intended to make backups more efficient in situations where a significant portion of the database is read-only and does not need to be included in backups. This is not applicable to Destiny.

File backups File backups and differential file backups back up one or more selected files or file groups within a database. A file or file group is a subset of a database. All files within the Destiny database must be backed up, so this tactic is not applicable to Destiny.

Copy-only backups Copy-only backups create a database backup without affecting the backup chain (without truncating the transaction log). Copy-only backups are independent of the regular backup sequence. If used, a copy-only backup would be performed outside a normal Destiny backup and restore process.

Technical Support

Destiny

Email techsupport@fsc.follett.com

Phone Support 800.323.3397

Follett Shelf

Email follettshelfsupport@follett.com

Phone Support 877.873.2764

Customer Service

Email customerservice@fsc.follett.com

Phone 800.323.3397 or 815.344.8700