

# Dell™ NetVault™ Backup Plug-in for MySQL 4.4

User's Guide



© 2014 Dell Inc.  
ALL RIGHTS RESERVED.

This guide contains proprietary information protected by copyright. The software described in this guide is furnished under a software license or nondisclosure agreement. This software may be used or copied only in accordance with the terms of the applicable agreement. No part of this guide may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording for any purpose other than the purchaser's personal use without the written permission of Dell Inc.

The information in this document is provided in connection with Dell products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Dell products. EXCEPT AS SET FORTH IN THE TERMS AND CONDITIONS AS SPECIFIED IN THE LICENSE AGREEMENT FOR THIS PRODUCT, DELL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL DELL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF DELL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Dell makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Dell does not make any commitment to update the information contained in this document.

If you have any questions regarding your potential use of this material, contact:

Dell Inc.  
Attn: LEGAL Dept  
5 Polaris Way  
Aliso Viejo, CA 92656

Refer to our web site ([software.dell.com](http://software.dell.com)) for regional and international office information.


#### Patents


This product is protected by U.S. Patents #7,814,260; 7,913,043; 7,979,650; 8,086,782; 8,145,864; 8,171,247; 8,255,654; 8,271,755; 8,311,985; 8,452,731; and 8,544,023. Protected by Japanese, E.U., French, and UK patents 1615131 and 05250687.0, and German patent DE602004002858. Additional patents pending. For more information, go to <http://software.dell.com/legal/patents.aspx>.


#### Trademarks

Dell, the Dell logo, and NetVault are trademarks of Dell Inc. and/or its affiliates. FreeBSD is a registered trademark of The FreeBSD Foundation. Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both. Mac and OS X are trademarks of Apple Inc., registered in the U.S. and other countries. MySQL is a registered trademark of MySQL AB in the United States, the European Union and other countries. Sun, Oracle, Java, and Oracle Solaris are trademarks or registered trademarks of Oracle and/or its affiliates in the United States and other countries. SPARC is a registered trademark of SPARC International, Inc. in the United States and other countries. Products bearing the SPARC trademarks are based on an architecture developed by Oracle Corporation. Windows is a registered trademark of Microsoft Corporation in the United States and/or other countries. Other trademarks and trade names may be used in this document to refer to either the entities claiming the marks and names or their products. Dell disclaims any proprietary interest in the marks and names of others.

#### Legend

 **CAUTION:** A CAUTION icon indicates potential damage to hardware or loss of data if instructions are not followed.

 **WARNING:** A WARNING icon indicates a potential for property damage, personal injury, or death.

 **IMPORTANT NOTE, NOTE, TIP, MOBILE, or VIDEO:** An information icon indicates supporting information.

NetVault Backup Plug-in for MySQL User's Guide  
Updated - April 2014  
Software Version - 4.4  
MYG-101-4.4-EN-01

# Contents

<b>Introducing Dell™ NetVault™ Backup Plug-in for MySQL</b> . . . . .	<b>5</b>
Dell NetVault Backup Plug-in for MySQL - at a Glance . . . . .	5
Key benefits . . . . .	5
Feature summary . . . . .	6
Target audience . . . . .	7
Recommended additional reading . . . . .	7
<b>Installing and removing the plug-in</b> . . . . .	<b>8</b>
Installation prerequisites . . . . .	8
Enabling the Binary Log on the MySQL Server (Standard/Community option only) . . . . .	9
Reviewing the recommended configuration . . . . .	10
Installing the plug-in . . . . .	10
Removing the plug-in . . . . .	11
Removing a specific MySQL Instance . . . . .	11
<b>Configuring the plug-in</b> . . . . .	<b>12</b>
Configuring default settings . . . . .	12
Updating the configuration of an existing instance . . . . .	14
Setting default actions for error conditions (optional) . . . . .	15
<b>Backing up data</b> . . . . .	<b>16</b>
Backing up data - an overview . . . . .	16
Important notes for MySQL Standard/Community . . . . .	16
Important notes for MySQL Enterprise Backup . . . . .	17
Defining a backup strategy . . . . .	17
Performing a backup . . . . .	21
Selecting data for a backup . . . . .	21
Setting backup options . . . . .	22
Finalizing and submitting the job . . . . .	24
<b>Restoring data</b> . . . . .	<b>25</b>
Restoring data - an overview . . . . .	25
Reviewing the available restore methods for MySQL Standard/Community . . . . .	25
Restoring data in MySQL . . . . .	26
Selecting data for restore . . . . .	26
Setting restore options . . . . .	27
Finalizing and submitting a job . . . . .	33
Examples of restore scenarios for MySQL Standard/Community . . . . .	33
Examples of restore scenarios for MySQL Enterprise Backup . . . . .	54
Using advanced restore procedures for MySQL Standard/Community . . . . .	54
Renaming a database during a restore . . . . .	54
Restoring to a different MySQL Instance on the same server . . . . .	55
Recovering to an alternate MySQL Server . . . . .	57

<b>Working with native MySQL replication</b> . . . . .	<b>60</b>
Using the plug-in in a native environment - an overview . . . . .	.60
Enabling replication support . . . . .	.60
Backing up replication servers . . . . .	.61
Replication configuration backups . . . . .	.61
Restoring replication servers . . . . .	.61
<b>Using the plug-in in a Failover Cluster environment</b> . . . . .	<b>62</b>
MySQL Server Failover Clustering - an overview . . . . .	.62
Important considerations . . . . .	.62
Installing the plug-in . . . . .	.63
Installation prerequisites . . . . .	.63
Installing the software . . . . .	.63
Licensing the plug-in . . . . .	.64
Configuring the plug-in . . . . .	.64
Backing up data . . . . .	.64
Restoring data . . . . .	.65
<b>Troubleshooting</b> . . . . .	<b>66</b>
<b>About Dell</b> . . . . .	<b>68</b>
Contacting Dell . . . . .	.68
Technical support resources . . . . .	.68

# Introducing Dell™ NetVault™ Backup Plug-in for MySQL

- [Dell NetVault Backup Plug-in for MySQL - at a Glance](#)
- [Key benefits](#)
- [Feature summary](#)
- [Target audience](#)
- [Recommended additional reading](#)

## Dell NetVault Backup Plug-in for MySQL - at a Glance

The **Dell NetVault Backup (NetVault Backup) Plug-in for MySQL (Plug-in for MySQL)** consolidates the backup and recovery of multiple MySQL storage engines into a single job without complex scripting. If you use the **MySQL Enterprise Backup** option (MEB-based method), the plug-in supports hot backups of InnoDB tables during backup. If you use the **MySQL Standard/Community** option (mysqldump-based method), the plug-in supports warm backups of all tables while keeping data online with read-only access. Additionally, using the **MySQL Standard/Community** option, Plug-in for MySQL offers improved point-in-time (PIT) functionality to perform more granular restores—which lets you restore to a precise point and reduce data loss.

## Key benefits

- **Increases confidence and reduces risk while deploying MySQL** - Plug-in for MySQL eliminates the need to create complex backup scripts and is flexible enough to account for a large number of recovery scenarios. Users no longer have to worry about understanding MySQL internals before being able to implement a backup policy that prevents committed transactions from being lost during backup and understanding the proper time to purge the Binary Logs. This knowledge is built into the plug-in.

Plug-in for MySQL's flexible backup features also include:

- Full, Incremental, and Differential Backups while data is online and accessible
- Common user interface across multiple storage engines
- Protection down to the table level
- Consolidation of multiple storage engines into a single job

By relying on Plug-in for MySQL to implement backup policies, users can be freed to focus on more critical tasks without risking the ability to recover exactly what is needed in the unfortunate event of a failure. In addition, the IT manager's confidence is increased by knowing that MySQL data is protected, no matter what.

- **Speeds up restores to reduce downtime** - With Plug-in for MySQL, you select what needs to be restored, the backup set to restore from, and, if applicable, the time or position point to restore to; and

the plug-in automatically performs the restore. Restores are faster due to minimized human interaction, and the potential for syntax errors is eliminated.

Additional Plug-in for MySQL restore features include:

- Full, Incremental, and both time-based and position-based PIT Restores
- Restores of complete instances, individual databases, or individual tables
- Rename of databases during restores
- Restores to alternate MySQL Instances
- **Ensure business continuity with automatic integration of backup devices** - With offsite backups being an important part of the data protection plan for any mission-critical application, Plug-in for MySQL takes advantage of NetVault Backup's fundamental integration with a wide range of backup devices, which includes: Dell NetVault SmartDisk, tape libraries, stand-alone tape devices, virtual tape libraries (VTLs), and shared VTLs. NetVault Backup lets you select which backup device to store the backup on. You can store the backup online in a virtual tape library and duplicate the job to physical tape libraries that are shared by multiple MySQL Instances or other proprietary databases, or used for general backup purposes.
- **Supports advanced MySQL replication techniques** -As detailed in the *MySQL Reference Guide*, MySQL features support one-way, asynchronous replication, in which one server acts as the master, while one or more other servers act as slaves.

In single-master replication, the master server writes updates to its Binary Logs and maintains an index of those files to keep track of log rotation. The Binary Logs serve as a record of updates to be sent to any slave servers. When a slave connects to its master, it informs the master of the position up to which the slave read the logs at its last successful update. The slave receives any updates that have taken place since that time, and then blocks and waits for the master to notify it of new updates.

Plug-in for MySQL gives you the confidence that your email MySQL environments are being protected and stored offsite for disaster-recovery purposes. At the same time, it frees administrators from having to be available 24x7 because less experienced MySQL administrators can initiate restores with the confidence that they are performed correctly and as quickly as possible, thereby reducing downtime and improving business continuity.

## Feature summary

- Support for the following with the **MySQL Standard/Community** option:
  - Full and Incremental Backups
  - Differential Backups
  - Individual Database/Table Copy Only Backups
  - InnoDB, MyISAM, MERGE (also known as MRG\_MyISAM), Memory/Heap, Federated, Berkeley DB (BDB), Archive, and CSV Storage Engines
  - Common user interface across Storage Engines
  - Time- and position-based PIT Restores
  - PIT Restores prior to and after data corruption
  - Restoration of individual tables or databases, or entire instances
  - Rename databases during restore
  - Restore to alternate instances
  - Native MySQL Replication Slave and Master Instance Backups
- Support for the following with the **MySQL Enterprise Backup** option:
  - Full and Incremental Backups

- InnoDB, MyISAM, MERGE (also known as MRG\_MyISAM), Archive, and CSV Storage Engines
- Hot backups of the InnoDB tables
- Common user interface across Storage Engines
- Restoration of individual tables or databases, or entire instances

## Target audience

While advanced MySQL database administrator (DBA) skills are generally not required to create and execute routine backup operations, they are required for defining the MySQL backup-and-recovery strategy and performing advanced recovery scenarios.

## Recommended additional reading

Dell recommends that you have the following documentation available for reference while setting up and using this plug-in.

- **MySQL <X> Reference Guide** (where <X> refers to the version of MySQL installed on the MySQL Server):
  - **MySQL ver. 5.0:** <http://dev.mysql.com/doc/refman/5.0/en/index.html>
  - **MySQL ver. 5.1:** <http://dev.mysql.com/doc/refman/5.1/en/index.html>
  - **MySQL ver. 5.5:** <http://dev.mysql.com/doc/refman/5.5/en/index.html>
  - **MySQL ver. 5.6:** <http://dev.mysql.com/doc/refman/5.6/en/index.html>
- **NetVault Backup documentation:**
  - *Dell NetVault Backup Installation Guide* - This guide provides complete details on installing the NetVault Backup Server and Heterogeneous Client software.
  - *Dell NetVault Backup Administrator's Guide* - This guide explains how to use NetVault Backup and describes the functionality common to all plug-ins.
  - *Dell NetVault Backup Command Line Interface Reference Guide* - This guide provides a detailed description of the command line utilities.

You can download these guides from <http://software.dell.com/support/>.

- ① **IMPORTANT:** Starting with 10.0, NetVault Backup provides a web-based user interface (WebUI) to configure, manage, and monitor your NetVault Backup system and installed plug-ins. The procedures described in the user's guide for this version of the plug-in are intended for the new WebUI. For procedures based on the NetVault Backup Console (user interface available with NetVault Backup 9.x and 8.x), refer to the documentation for an earlier version of the plug-in.

# Installing and removing the plug-in

- [Installation prerequisites](#)
- [Reviewing the recommended configuration](#)
- [Installing the plug-in](#)
- [Removing the plug-in](#)
- [Removing a specific MySQL Instance](#)

## Installation prerequisites

Before installing Plug-in for MySQL, make sure that the following software is installed and properly configured on the machine that is to serve as the MySQL Server:

- **NetVault Backup Server/Client software** - At a minimum, the NetVault Backup Client must be installed on the machine configured as the MySQL Server.
- **MySQL Database software**
- **Enable the Binary Log on the MySQL Server (MySQL Standard/Community option only)** - This allows for support of **Point-in-Time (PIT)** backups and restores of the MySQL Server. For more information, see [Enabling the Binary Log on the MySQL Server \(Standard/Community option only\)](#).
- **Proper version of the MySQL Database Client package** - The plug-in interacts with components installed with the MySQL Client package and lets you access more functionality with the plug-in. The version of the components installed with this package must be compatible with the installed version of MySQL. Primarily, these two MySQL components should be installed and their version verified:
  - **mysqldump** - This utility lets you perform backups and restores of multiple types of MySQL storage engines. Verify that the version of this component is compatible with the current version of MySQL, and that it is *not* the version provided with an earlier version of Plug-in for MySQL.
  - **mysqlbinlog** - This utility lets you use PIT backups and restores. Ensure that the proper version of this component is available for use with the installed version of MySQL.
- **MySQL Enterprise Backup** - If you want to use the **MySQL Enterprise Backup** option, your environment must meet the following requirements:
  - For Linux/UNIX environments, your MySQL Server must use version 5.5 or 5.6.
  - For Windows environments, your MySQL Server must use version 5.5.
  - Version 3.8.1 of the MySQL Enterprise Backup product must be installed. MySQL Enterprise Backup is available with MySQL Enterprise Edition and with select Commercial Editions. For installation instructions, refer to: <http://dev.mysql.com/doc/mysql-enterprise-backup/3.8/en/installing.html>



# Enabling the Binary Log on the MySQL Server (Standard/Community option only)

Prior to configuring support for PIT backups and restores with the **MySQL Standard/Community** option, you must enable the MySQL Binary Log.

## Enabling the log on a Linux- or UNIX-based MySQL Server

- 1 Access the MySQL installation directory, and locate the MySQL configuration file (for example, “my.cnf”).

The name and location of the file are contingent on your MySQL configuration. For more information, refer to your MySQL documentation.

- 2 Using a text editor, open the file, and locate the “[mysqld]” section.
- 3 To use the default MySQL directory to house the MySQL Binary Log, add the following entry:

```
log-bin
```

- IMPORTANT:** If desired, the “log-bin” entry added to the my.cnf file can be set up using the following syntax to specify a different file to house the Binary Log:

```
log-bin=<NameOfDestinationFile>
```

When you specify the name of the destination file for the Binary Log, use only the exact name of the file itself; do not include full path information or the file-name extension. For more information on enabling the Binary Log, refer to the MySQL Reference Guide before continuing with installation of the plug-in.

- 4 To enable the changes, restart the MySQL Server.

## Enabling the log on a Windows-based MySQL Server

- 1 Launch the **MySQL Administrator** application (for more information, refer to the relevant MySQL documentation).

- IMPORTANT:** If you do not have the MySQL Administrator installed, you must update the configuration file on a Linux/UNIX system and then stop and restart the MySQL service to enable binary logging.

- 2 In the **MySQL Administrator** window, click **Startup Variables** in the left pane.
- 3 In the right pane, select the **Log Files** tab.
- 4 Select the **Binary Logfile Name** check box, and either enter a unique name for the file, or leave the field blank to use the default value of **log-bin**.

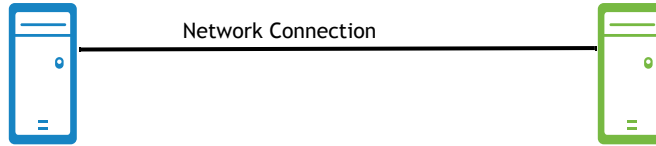
- IMPORTANT:** When you specify the name of the destination file for the Binary Log, use only the exact name of the file itself; do not include full path information or the file-name extension. For more information on enabling the Binary Log, refer to the MySQL Reference Guide before continuing with installation of the plug-in.

- 5 Exit the **MySQL Administrator** application.
- 6 To enable the changes, restart the MySQL Server.

# Reviewing the recommended configuration

While you can set up a single machine as both the NetVault Backup Server and the MySQL server (that is, all software installation and configuration requirements are performed on a single machine), Dell recommends that these two entities exist on *separate* machines.

Table 1. Recommended configuration



MySQL Server machine	NetVault Backup Server machine
<b>Software installed/configuration</b> <ul style="list-style-type: none"><li>• MySQL Database software (ver 5.0 or later)</li><li>• NetVault Backup Server/Client software</li><li>• Plug-in for MySQL</li><li>• Binary Log enabled (MySQL Standard/Community option only)</li><li>• mysqldump/mysqlbinlog utilities (compatible with the version of MySQL installed)</li><li>• mysqlbackup utility (MySQL Enterprise Backup option only)</li></ul>	<b>Software installed/configuration</b> <ul style="list-style-type: none"><li>• NetVault Backup Server software</li><li>• MySQL Server added as a NetVault Backup Client (For details on adding a Client machine to the NetVault Backup Server, refer to the <i>Dell NetVault Backup Administrator's Guide</i>.)</li></ul>


**IMPORTANT:** Sample images and procedures throughout this guide assume that you are using this *two-machine environment* and that configuration requirements have been met.

## Installing the plug-in

### To install the plug-in

- 1 Access the **NetVault Configuration Wizard** or **Manage Clients** page.

**NOTE:** You can use the configuration wizard to install the plug-in on multiple clients at the same time (provided that the selected clients are all the same type). When you select multiple clients, make sure that the plug-in binary file is compatible with the OS and platforms of the target clients. From the **Manage Clients** page, you can only select one client for plug-in installation.

- To access the **NetVault Configuration Wizard** page:
    - a In the Navigation pane, click **Guided Configuration**.
    - b On the **NetVault Configuration Wizard** page, click **Install Plugins**.
    - c On the next page, select the applicable clients.
  - To access the **Manage Clients** page:
    - a In the Navigation pane, click **Manage Clients**.
    - b On the **Manage Clients** page, select the applicable machine, and click **Manage**.
    - c On the **View Client** page, click the **Install Plug-in** button (  ).
- 2 Click **Choose Plug-in File**, navigate to the location of the “.npx” installation file for the plug-in (on the installation CD or the directory to which the file was downloaded from the web site).

Based on the operating system (OS) in use, the path for this software may vary on the installation CD.

- 3 Select the file entitled “**mys-x-x-x-x.npk**” (where **xxxx** represents the platform and version number), and click **Open**.
- 4 Click **Install Plugin** to begin installation.

After the plug-in is successfully installed, a message is displayed.

## Removing the plug-in

### *To remove the plug-in*

- 1 In the Navigation pane, click **Manage Clients**.
- 2 On the **Manage Clients** page, select the applicable client, and click **Manage**.
- 3 In the **Installed Software** table on the **View Client** page, select **Plug-in for MySQL**, and click the **Remove Plugin** button (🗑️).
- 4 In the **Confirm** dialog, click **Remove**.

## Removing a specific MySQL Instance

After a MySQL Instance has been successfully configured and added to the plug-in, you can also remove it.

📘 **IMPORTANT:** Use this procedure with caution. However, you can add the instance again by following the steps in [Configuring the plug-in](#).

### *To remove the an instance*

- 1 In the Navigation Pane, click **Create Backup Job**, and click **Create New** next to the **Selections** list.
- 2 In the selection tree, open the applicable client node.
- 3 Open **Plug-in for MySQL**, and select the applicable instance.
- 4 From the **Actions** list, select **Remove Server**.

No confirmation dialog appears after this command is used.

## Configuring the plug-in

- [Configuring default settings](#)
- [Setting default actions for error conditions \(optional\)](#)

### Configuring default settings

Plug-in for MySQL supports multiple MySQL Instances from a single MySQL Server, and each individual instance must be specifically configured for use. Note that the configuration options available will vary, based on the OS in use on the MySQL Server and whether you use the **MySQL Standard/Community** option or **MySQL Enterprise Backup** option.

#### To configure the plug-in

- 1 In the Navigation Pane, click **Create Backup Job**, and click **Create New** next to the **Selections** list.
- 2 In the selection tree, open the applicable client node, and then select **Plug-in for MySQL**.
- 3 From the **Actions** list, select **Add New Server**.
- 4 On the **Configure** dialog, complete the applicable fields:
  - **MySQL Instance Name** (required) - Enter the applicable name for the instance of MySQL (by default, the local host name is used). This value will be used in the NetVault Backup WebUI.
  - **MySQL Edition** (required) - Select the applicable option: **MySQL Standard/Community** or **MySQL Enterprise Backup**. The version that you are using determines which options are modifiable on this dialog.
  - **Username and Password** (required) - Enter the user name and password that allows sufficient rights to read and write to the tables in the MySQL Instance's database that is to be targeted for backup and restore (for example, an account with administrator privileges).
    - ① **IMPORTANT:** NetVault Backup references **Username** and **Password** values each time it attempts to access the MySQL database for a selected instance (that is, for both backups and restores). If either of these values is changed for the MySQL Instance, it *must* be updated in these fields; otherwise, NetVault Backup will be unable to access the instance and job requests will fail.
  - **MySQL Base Directory** (required) - Enter the complete path to the base directory where the MySQL program files reside.
    - **Windows-based Server** - To locate the directory on Windows, query the Windows Registry for the "Location" value.
    - **Linux/UNIX-based Server** - With a default installation of MySQL, the directory is located as follows:
 

```
"/var/lib/mysql"
```
  - **MySQL Bin Directory** - Enter the complete path to the directory that contains the MySQL executable files on the MySQL Server. By default, the directory is as follows:
    - **Linux/UNIX-based Server:**

```
"<MySQLbaseDirectory>/bin"
```

- **Windows-based Server:**

```
"<MySQLbaseDirectory>\bin"
```

- **Mysqldump Path** - Enter the complete path and file name for the `mysqldump` utility, which is used during the backup and restore process. If a default installation of MySQL was performed for the target instance, the default value might appear based on the OS in use on the MySQL Server:

- **Linux/UNIX-based Server:**

```
"<MySQLbaseDirectory>/bin/mysqldump"
```

- **Windows-based Server:**

```
"<MySQLbaseDirectory>\bin\mysqldump.exe"
```

**IMPORTANT:** If either the MySQL Bin Directory or the `mysqldump` path is set to the defaults listed earlier, the field can be left blank.

- **TCP Port** (for Windows-based Servers only) - Each instance of MySQL requires its own port value for proper access. Default installations of MySQL use port **3306**, which appears in this field by default. If a different port has been set up for the selected instance, enter the correct value.

**IMPORTANT:** If multiple instances exist on a single MySQL Server, each is assigned its own port value, and this value must be entered in the **Port Number** field. This value is equivalent to what is revealed for the “`port=`” value, as found in the “`my.ini`” file for each instance.

- **Socket File Path** (for Linux/UNIX-based Servers only) - Enter the path and file name of the MySQL socket file. If a default installation of MySQL was performed, a default value will appear in this field. By default, the socket file is located in the following directory:

```
"/tmp/mysql.sock"
```

**IMPORTANT:** If a standard installation of MySQL was performed on the MySQL Server and the default directory was used, the **Socket File** path should remain at its default setting. However, if a different directory was selected during installation, you must enter the correct location in this field. To determine this path, issue the following command from a terminal session prompt on the MySQL server:

```
“ show variables like ‘socket’ ”
```

If the correct Socket File variable is not entered, the plug-in will not perform backups and restores.

- **Default Character Set** - The default character set is latin1. If you want to use a different character set for encoding (such as UTF-8), select it from the list.

5 If you are using **MySQL Standard/Community**, complete the following fields:

- **MyISAM Backup Method** - If your environment uses the MyISAM storage engine or table type, select this check box, and then select the applicable suboption.
  - **Lock & Copy Table Files** (default selection) - Select this option to use the standard method of locking, flushing, and copying the table files.
  - **Mysqldump** - If the tables are subject to heavy use or loading, select this option to use the `mysqldump` utility instead of copying the tables. Be aware that this might impact performance.
- **Enable MySQL Replication** - If native MySQL Replication is enabled for this instance, select this check box. For more information, see [Working with native MySQL replication](#).
  - **Slave Instance** - For replication-enabled instances, select this option if this instance is configured as a slave.
  - **Master Instance** - For replication-enabled instances, select this option if this instance is configured as the master.

ⓘ | **IMPORTANT:** Do *not* select this option if you do not intend to configure replication; otherwise, backups will fail.

- **Enable Point-In-Time Recovery** - If you want to enable PIT backups and restores, select this check box. This lets you perform a recovery up to a point prior to, or after, data corruption occurs. To use this feature, you must enable the MySQL Binary Log as outlined in [Enabling the Binary Log on the MySQL Server \(Standard/Community option only\)](#).
- **Binary Log Index Path** - If you selected the **Enable Point in Time Recovery** check box, use this field to specify the complete path to the Binary Log Index file. By default, the path and file are as follows:

- **Linux/UNIX server:**

```
<MySQLbaseDirectory>/data/<instanceName>-bin.index
```

- **Windows server:**

```
<MySQLbaseDirectory>\data\<instanceName>-bin.index
```

ⓘ | **IMPORTANT:** The plug-in is able to determine if a specified file in the Binary Log Index Path exists during configuration. However, it is unable to determine if the file specified is actually the Binary Log Index until the backup job is submitted. If it determines that the specified file is invalid, the job fails.

- **Relay Log Index Path** - If you are configuring a Slave Instance, enter the complete path to the Relay Log Index file to include it in backups.

- **Linux/UNIX server:**

```
<MySQLbaseDirectory>/data/<instanceName>-relay-bin.index
```

- **Windows server:**

```
<MySQLbaseDirectory>\data\<instanceName>-relay-bin.index
```

6 If you are using **MySQL Enterprise Backup**, complete the required **Mysqlbackup Path** field by entering the complete path to the directory where the **mysqlbackup** utility resides.

7 Click **OK** to save the settings.

## Updating the configuration of an existing instance

After an instance of MySQL has been successfully configured and added to the plug-in for use, you can edit its configuration options by performing the following steps:

- 1 In the Navigation Pane, click **Create Backup Job**, and click **Create New** next to the **Selections** list.
- 2 In the selection tree, open the applicable client node.
- 3 Open the **Plug-in for MySQL** node, and select the applicable instance.
- 4 From the **Actions** list, select **Configure**.

The **Configure** dialog appears with all previous settings revealed, allowing you to make any necessary modifications.

ⓘ | **IMPORTANT:** In the **Edit** version, the **MySQL Instance Name** field is dimmed. This field is for information only and displays the name of the selected instance.

# Setting default actions for error conditions (optional)

MySQL backup jobs typically include multiple storage engines, databases, and/or tables. Occasionally during the execution of a backup job, an unsupported storage engine will be encountered or a database/table will be inaccessible, which prevents one or more items from being backed up successfully, while the remaining items selected in the backup job *are* backed up. A MySQL DBA needs to determine what action should be taken when these conditions are encountered:

- Should the backup job complete with warnings, complete without warnings, or fail?
- Should the backup of items that completed successfully be retained or discarded?


The plug-in lets you set default options for backup and restore jobs. You can override these options on a per-job basis.

## To configure default settings

- 1 In the Navigation pane, click **Change Settings**.
- 2 On the **Configuration** page, click **Server Settings** or **Client Settings**, as applicable.
- 3 If you selected **Client Settings**, select the applicable client, and click **Next**.
- 4 On the **NetVault Server Settings** or **Client Settings** page, click **Plugin Options**.

In the **Plug-in for MySQL** section, the following items are listed:

- **Locked Table** - This occurs when a table selected for inclusion in the backup is locked by a client session other than the plug-in.
  - **Manually Selected Table Unavailable** - This occurs when an individual table is unavailable for backup for any reason, such as being dropped since the backup job was defined
  - **Manually Selected Database Unavailable** - This occurs when an individual database is unavailable for backup for any reason, such as being dropped since the backup job was defined
  - **Unsupported Storage Engine** - This occurs when a table is encountered during the backup that has a type of storage engine that is not currently supported by the plug-in.
- 5 For each of these conditions, select one of the following settings:
    - **Complete with Warnings - Saveset Retained** - The job returns a status of “**Backup Completed with warnings**” and a backup saveset is created that includes the items that were successfully backed up.
    - **Complete without Warnings - Saveset Retained** - The job completes and returns a status of “**Backup Completed.**” While errors are logged in the NetVault Backup binary logs, the errors are essentially ignored in the **Job Status** page and a backup saveset is created that includes the items that were successfully backed up.
    - **Fail - Saveset Retained** - The job returns a status of “**Backup Failed.**” However, a backup saveset is generated that includes the items that were successfully backed up.
    - **Fail - No Saveset Retained** - The job returns a status of “**Backup Failed**” and no saveset of backed up objects is kept. That is, even if some of the objects were successfully backed up, the saveset is discarded.

 **IMPORTANT:** You can override the selected default action selected at the individual backup job level.

- 6 Click **Apply** to save the settings.

# Backing up data

- [Backing up data - an overview](#)
- [Performing a backup](#)

## Backing up data - an overview

Before completing a backup, review the information in the following sections:

- [Important notes for MySQL Standard/Community](#)
- [Important notes for MySQL Enterprise Backup](#)
- [Defining a backup strategy](#)

## Important notes for MySQL Standard/Community

If your environment uses databases whose names contain special characters, such as hyphens (all characters except alphanumeric characters and underscores are considered special characters), be aware of the following limitations:

- All table types except MyISAM are backed up, even if the database name contains hyphens. This is because the `mysqldump` command is always used for these table types.
- If the database name contains hyphens, MyISAM tables are backed up if the **MyISAM Backup Method** is set to the **Mysqldump** option introduced in version 4.2. Note that performance of backups and restores might be negatively affected.
- If the **MyISAM Backup Method** is set to use the default **Lock & Copy Table Files** option and the database name contains hyphens, MyISAM tables are *not* backed up because the plug-in bypasses MySQL commands and tries to copy the table files directly. The plug-in will log an error message indicating that the table file cannot be located, and then fail the backup job without creating a saveset.

With previous versions, the plug-in would attempt to verify the existence of the database directory, log a warning message when it failed to do so, and continue with backup of the next database. The backup would complete with warnings and create a saveset that included all other databases.

If you want to maintain the original behavior and still use the **Lock & Copy Table Files** option for any reason, such as less-than-optimal performance impact when using the **Mysqldump** option, you can manually set the `ValidateDatabaseDirectory` parameter to `TRUE` in the plug-in configuration file, “`nvmysql.cfg`,” as follows:

```
[MySQL:ValidateDatabaseDirectory]  
Value=TRUE
```

If you subsequently decide to use the new behavior instead, you can change the parameter to `FALSE` or remove the parameter from the “`nvmysql.cfg`” file.



# Important notes for MySQL Enterprise Backup

If you intend to use the **MySQL Enterprise Backup** option, review the following guidelines and information:

- MySQL recommends that you use InnoDB tables for critical data because the backup process is faster and the reliability and scalability features are significant. MySQL Enterprise Backup lets you back up various kinds of MySQL tables, and it is optimized for backing up InnoDB tables. This option performs a hot backup of all InnoDB tables. Because hot backups are performed while the database is running, the backup does not stop ongoing database operations. Additionally, any database changes made during the backup process are included. This is important if your environment requires that the database remains online while also supporting its growth, which impacts the time required to complete a backup.
- When you use this option, the MyISAM table and other non-InnoDB tables are backed up last, using a warm backup. In a warm backup, the database continues to run, but the tables are set to read-only access while the backup is completed.
- If you want to ensure that the bulk of your data is backed up during the hot-backup phase, consider making InnoDB the default storage engine for new tables and converting existing tables to use the InnoDB storage engine. (In MySQL Server 5.5 and later, InnoDB is the default.)
- An Incremental Backup is primarily intended for InnoDB tables and non-InnoDB tables that are read-only or updated infrequently. For non-InnoDB files, the entire file is included if it changed since the last backup.
- When using the plug-in, all InnoDB tables in a MySQL instance are backed up if either of the following conditions is met:

- Only tables are explicitly selected for backup, and none of the tables are of type or storage engine InnoDB.

**Example:** You have a MySQL Instance with two databases (DB1 and DB2). Each database contains two tables: DB1 has T1\_InnoDB and T1\_MyISAM and DB2 has T2\_InnoDB and T2\_MyISAM. If you back up T1\_MyISAM and T2\_MyISAM, T1\_InnoDB and T2\_InnoDB are also backed up. If you include one of the InnoDB tables, only that InnoDB table is backed up. If you select one of the databases, only the tables in the database are backed up.

- Some or all databases are selected for backup and all associated InnoDB tables are excluded from the backup.

**Example:** You have a MySQL Instance with two databases (DB1 and DB2). Each database contains two tables: DB1 has T1\_InnoDB and T1\_MyISAM and DB2 has T2\_InnoDB and T2\_MyISAM. If you back up DB1 and DB2 and exclude T1\_InnoDB and T2\_InnoDB, T1\_InnoDB and T2\_InnoDB are also backed up. If you exclude only one of the two InnoDB tables, only the other InnoDB table is backed up.

This reflects the current behavior of MySQL Enterprise Backup (**mysqlbackup** utility); this behavior might change in a future release (post-3.8.1) of MySQL.

- In MySQL 5.5, the **innodb\_file\_per\_table** configuration option is *disabled* by default. In MySQL 5.6, it is *enabled* by default. Any InnoDB tables that are created with the **innodb\_file\_per\_table** option disabled are stored in the InnoDB system tablespace; they cannot be omitted from the backup. If you need to place an InnoDB table outside the tablespace, create it while the **innodb\_file\_per\_table** option is enabled in MySQL. Each .ibd file will contain the data and indexes of only one table.

## Defining a backup strategy

When defining a MySQL backup strategy, you must answer the following questions:

- Do I want to use the **MySQL Standard/Community** or **MySQL Enterprise Backup** option? Even if you have both versions implemented in your environment, you can only use one strategy with the plug-in. You must use either the MEB-based method or the mysqldump-based method; you cannot use a mixture of the two.

If you use the MEB-based option, the **mysqlbackup** command is executed once for all the database objects that you select for backup, and a **mysqlbackup** output log is included in the job log. Backing up

data involves two stages. In the first stage, all InnoDB tables are copied. In the second stage, all other types of tables are copied. In addition to providing support for hot backup of InnoDB tables, the MEB-based option supports improved backup performance.

If you use the `mysqldump`-based option, the command is executed once for each table, trigger, and stored procedure. (Hot backups are not supported.)

- Understanding that read-only access across the entire instance is required during Full Backups, how often should full-backups be taken?
- What is more important: quicker backups or quicker restores?
- What is the maximum amount of acceptable data loss?

Answering these questions will help you define the type and frequency of backups that should be implemented.

- [Reviewing the backup types for MySQL Standard/Community](#)
- [Reviewing the backup types for MySQL Enterprise Backup](#)
- [Examples of backup sequences for MySQL Standard/Community](#)

## Reviewing the backup types for MySQL Standard/Community

If you use the `MySQL Standard/Community` option, the plug-in uses `mysqldump` to provide the following types of backup:

- **Full Backup**
- **Incremental Backup**
- **Differential Backup**
- **Individual Database/Table Copy Only Backup**

Understanding how these backups differ is the first step in selecting a suitable backup sequence that matches the data protection requirements for each MySQL Instance.

### Full Backups

In a Full Backup for the `MySQL Standard/Community` option, the plug-in uses the `mysqldump` utility to back up *every database included in the instance*. Full Backups are the foundation of any backup strategy because they provide the starting point for almost every restore scenario. Full Backups generated with the plug-in can be used to restore an entire instance, individual or multiple databases, and individual or multiple tables.

The **Purge Binary Logs after Full or Incremental Backup** option, which is enabled by default when the **Enable MySQL Replication** is disabled and **Enable Point-In-Time Recovery** is enabled, ensures that Binary Logs are purged after a Full or Incremental Backup.

If you do not select the **Purge Binary Logs...** option, the plug-in keeps track of the **Last Log Backed Up** in its configuration file, and you can manually purge Binary Logs at your discretion. For example, if you are using a MySQL replication environment where you do not want to purge Binary Logs from the Master Instance until they have been replicated to the Slave Instance, you are responsible for manually purging the Binary Logs.

### Incremental Backups

An Incremental Backup backs up the transaction logs that were generated since the last Full or Incremental Backup, followed by purging the Binary Logs. Because the Binary Logs are instance-based, the transaction logs for every database are backed up and purged as a unit.

Incremental Backups are essential in reducing data loss after a media failure or data corruption because they can be used to restore to a time prior to and after a data corruption, such as incorrect update or dropped table. Unlike Full Backups, Incremental Backups do not require read-only access during the backup.

MySQL Incremental Backups require the MySQL Instance to be started with the `“-log-bin”` option, which enables Binary Logging. This procedure is outlined in [Enabling the Binary Log on the MySQL Server \(Standard/Community option only\)](#). For more information, refer to the Binary Log section in the *MySQL Reference Guide*.

As described earlier, the **Purge Binary Logs after Full or Incremental Backup** option, ensures that Binary Logs are purged after a Full or Incremental Backup. If you do not use this option, the plug-in keeps track of the **Last Log Backed Up** in its configuration file, and you can manually purge Binary Logs at your discretion.

## Differential Backups

A Differential Backup backs up the transaction logs that were generated since the last Full or Incremental Backup. However, the Binary Logs are *not purged* at the completion of the backup. Therefore, subsequent Differential Backups increase in size and in duration because each backup of this type includes the Binary Logs that were also included in the previous Differential Backup *as well as* the Binary Logs that have been generated since the previous Differential Backup. For example, if a Full Backup was taken on Sunday with Differential Backups scheduled Monday through Saturday, Monday's Differential will include the Binary Logs generated since the Full Backup on Sunday, while Tuesday's Differential will include the Binary Logs generated on Monday as well as those generated on Tuesday. Wednesday's Differential will include the Binary Logs for Monday, Tuesday, and Wednesday, and so on.

Similar to an Incremental Backup, a Differential can also be used to reduce data loss after a media failure or data corruption, with the ability to restore to a time prior to, and after the failure/corruption. Unlike a Full Backup, a Differential does *not* require read-only access during the backup.

Differential Backups require the MySQL Instance to be started with the “-log-bin” option, which enables Binary Logging. This procedure is outlined in [Enabling the Binary Log on the MySQL Server \(Standard/Community option only\)](#). For more information, refer to the Binary Log section in the *MySQL Reference Guide*.

## Incremental vs. Differential Backups

Because Incremental Backups purge the Binary Logs after they are backed up, subsequent Incrementals are quicker because only the Binary Logs that have been created since the last Incremental Backup are backed up. However, restore sequences that use Incremental Backups require that every Incremental taken between the Full Backup and the point of failure must be restored in succession. This can lead to longer restores due to the increased DBA intervention needed to initiate the multiple restore jobs.

Because Differential Backups do not purge the Binary Logs after they are backed up, each subsequent Differential Backup takes longer because all the Binary Logs since the last Full Backup are included in the backup. Nevertheless, restore sequences that use Differential Backups require that only one Differential Backup be restored after the Full Backup is restored. This results in quicker restores because less DBA intervention is required during the restore process.

## Individual Database/Table Copy Only Backups

Sometimes a backup must be taken for a special purpose and should not affect the overall backup and restore procedures for a complete database. For example, backups can be a source for a test environment or as an initial synchronization for a replication slave instance. Individual Database/Table Copy Only Backups are designed for these special purposes, in that they allow you to “copy” a MySQL environment. “**Copy Only**” backups are independent of an established sequence of backups and do not affect the recoverability of Full, Incremental, or Differential Backups. However, they *cannot* be used as a replacement for a Full Backup.

## Reviewing the backup types for MySQL Enterprise Backup

For the **MySQL Enterprise Backup** option, the plug-in executes the `mysqlbackup` command once for all selected database objects to achieve the following types of backup: Full and Incremental.

### Full Backups

In a Full Backup for the **MySQL Enterprise Backup** option, the plug-in uses the `mysqlbackup` utility to back up *every selected database object included in the instance*. Full Backups are the foundation of any backup strategy because they provide the starting point for almost every restore scenario. Full Backups generated with the plug-in can be used to restore an entire instance, individual or multiple databases, and individual or multiple tables.

## Incremental Backups

For an InnoDB table, only data that changed since the last Full or Incremental Backup is backed up. For a non-InnoDB table, the entire table is backed up if anything has changed in the table since the last Full or Incremental Backup.

## Examples of backup sequences for MySQL Standard/Community

The following sections provide examples of the various sequences.

### Full Backups Only

When business requirements guarantee data protection up to the previous day and daily read-only access is permissible (for example, after regular business hours), performing only Full Backups on a daily basis should be sufficient. While DBAs are only guaranteed to be able to recover the database up to the point of the last Full Backup, they do have the option to perform a PIT Recovery using the Binary Logs that currently exist on the MySQL server.

### Full and Incremental Backups

When business requirements guarantee data protection up to the previous day, but read-only access to the target MySQL Instance is only permissible at intermittent times (for example, after regular business, on only a weekly or bi-weekly basis), and *backup time should be as fast as possible*, Full Backups coupled with Incremental Backups is the best combination. For example, Full Backups are performed every Sunday night at 11:00 P.M., while Incremental Backups are performed Monday through Saturday at 11:00 P.M. Each Incremental Backup will include the Binary Logs that were generated since the previous night's backup, whether it be the Sunday evening Full Backup or one of the Incremental Backups.

Restoring this type of backup sequence is more time-consuming. For example, if recovery is performed on Tuesday, only Sunday's Full Backup and Monday's Incremental Backup must be restored; whereas, if recovery is performed on Thursday, Sunday's Full Backup followed by Monday's, Tuesday's, and Wednesday's Incremental Backups must be restored. Even though the backups are quicker, the restores can take longer due to the additional intervention that is required to execute multiple restore jobs.

### Full and Differential Backups

When business requirements guarantee data protection up to the previous day, but read-only access to the target MySQL Instance is only permissible at intermittent times (for example, after regular business hours, on only a weekly or bi-weekly basis), and *restore time should be as fast as possible*, Full Backups coupled with Differential Backups is the best combination. For example, Full Backups are performed every Sunday night at 11:00 P.M., while Differential Backups are performed Monday through Saturday at 11:00 P.M. Each Differential Backup will include the Binary Logs that were generated since the last Full Backup. As noted earlier, this requires more overall backup time.

Regardless of the time to which recovery is necessary, the same number of restore jobs is required. For example, if recovery is performed on Tuesday, Sunday's Full Backup and Monday's Differential Backup must be restored; whereas, if recovery is performed on Thursday, Sunday's Full Backup followed Wednesday's Differential Backup must be restored. Even though subsequent Differential Backups will increase not only in size but in duration, restores are quicker due to the fewer number of restore jobs that must be executed.

# Performing a backup

A backup using Plug-in for MySQL includes the steps outlined in the following sections.

- [Selecting data for a backup](#)
- [Setting backup options](#)
- [Finalizing and submitting the job](#)

## Selecting data for a backup

You must use sets (Backup Selection Set, Backup Options Set, Schedule Set, Target Set, and Advanced Options Set) to create a backup job.

Backup Selection Sets are essential for Incremental and Differential Backups. You must create the Backup Selection Set during a Full Backup, and use it for Full, Incremental, and Differential Backups. The backup job will report an error if you do not use a Selection Set for the Incremental or Differential Backup. For more information, refer to the *Dell NetVault Backup Administrator's Guide*.

### To create a Backup Selection Set

- 1 In the Navigation pane, click **Create Backup Job**.

You can also start the wizard from the Guided Configuration link. In the Navigation pane, click **Guided Configuration**. On the **NetVault Configuration Wizard** page, click **Create backup jobs**.

- 2 In **Job Name**, specify a name for the job.

Assign a descriptive title that lets you easily identify the job for monitoring its progress or restoring data. The job name can contain alphanumeric and non-alphanumeric characters, but it cannot include non-English characters. On Linux, the names can have a maximum of 200 characters. On Windows, there is no length restriction. However, a maximum of 40 characters is recommended on all platforms.

- 3 Click **Create New** next to the **Selections** list.

- 4 On the **NetVault Backup Selections** page, enter a name for the set in the **Backup Selection Set** box, and then open the NetVault Backup Client on which the plug-in is installed.

- 5 In the list of plug-ins, open **Plug-in for MySQL** to display the MySQL Servers.

- 6 Select the applicable data:

- To include all MySQL databases in the selected instance in the backup job, select the **All Databases** node.
- For a more granular selection, open the **All Databases** node to display the individual databases. In addition, you can open each individual database to display its individual tables, which can be selected as desired for inclusion in a backup job.
- To explicitly omit items from a backup, select a parent-level item, and click the applicable child-level item to replace the green check mark with a red "X" (cross), which marks it as omitted.

**IMPORTANT:** If you select a granular set of data for backup with the **MySQL Standard/Community** option, you must select **Individual Database/Table Copy Only** as the backup type on the **Backup Options** tab. If any other form of backup is selected (that is, **Full**, **Incremental**, or **Differential Backup**), granular selections will be ignored, and the entire database will be backed up. For MySQL 5.0.x and later, stored procedures, functions, and triggers are automatically backed up with **Full** and **Individual Database/Table Copy Only** backups for the **MySQL Standard/Community** option.

For MySQL 5.0.x and later, the "information\_schema" database will be displayed in the selection tree but it will not be available for selection. This is because all the data contained in this database is generated dynamically, and does not exist in any permanent sense. Therefore, the plug-in automatically excludes the information\_schema database from all backups.

- 7 Click **Save** to save the set.

### To use an existing Backup Selection Set

- 1 In the Navigation pane, click **Create Backup Job**.

You can also start the wizard from the Guided Configuration link. In the Navigation pane, click **Guided Configuration**. On the **NetVault Configuration Wizard** page, click **Create backup jobs**.

- 2 In the **Selections** list, select an existing Backup Selection Set.

## Setting backup options

The next step involves creating the Backup Options Set or selecting an existing one. The settings available on the Backup Options tab depend on whether you use the **MySQL Standard/Community** or **MySQL Enterprise Backup** option.

### Setting backup options for MySQL Standard/Community

With the applicable items selected for backup, you can select the type of backup to perform and select a different behavior in case of failure.

#### To create a Backup Options Set

- 1 Click **Create New** next to the **Plugin Options** list.
- 2 Select the applicable option:

**IMPORTANT:** If you specified the target instance of MySQL as a “**Replication Master Instance**” (that is, the **Enable MySQL Replication** and **Master Instance** options were selected in the **Configure** dialog for this instance), the **Full**, **Incremental**, and **Differential** forms of backup will be *unavailable* for selection. For more information, see [Working with native MySQL replication](#).

- **Full Backup for All Databases** (default selection) - To perform a complete backup of every database included in the current MySQL Instance, select this option.
- **Incremental Backup** - To back up only the transaction logs that were generated since the last Full or Incremental Backup, select this option.
- **Differential Backup** - To back up all the transaction logs that have been generated since the last Full or Incremental Backup, select this option. Each time a subsequent Differential is performed, it will contain all Binary Log files generated since the original Full was performed. After the backup is finished, the Binary Logs will be *kept* on the MySQL Instance.
- **Individual Database/Table Copy Only** - To create a special-purpose copy of a MySQL environment that will not affect the overall backup and restore procedures for the database (for example, creating a test environment), select this option. When used, these copy backups will not affect the sequence established by a Full and Incremental/Differential MySQL backup scenario (that is, these backups have no affect on Binary Log records). This form of backup is independent of what should be established as a normal sequence of backups through the use of a Full and Incremental/Differential MySQL backup scenario. In addition, a copy backup *cannot* be used as a replacement for a Full Backup.
- **Lock All Tables with Read Access...** - If you selected **Full Backup** and you want to prevent the loss of transactions by locking all the databases in the instance that currently have read-only access, select this check box. When this option is selected, users will be unable to insert, update, or delete data across the entire instance while the Full Backup is in progress. When this option is cleared, Plug-in for MySQL locks each table during the backup process *only* when the table is being backed up; therefore, if the instance contains tables that are related, Dell strongly recommends that you select this option to ensure that all tables are locked during the process.

- **Purge Binary Logs...** - This option is selected by default if **Enable MySQL Replication** is *not* selected and **Enable Point-In-Time Recovery** is selected. Dell recommends that you use this option, but you can decide how much control you want over the binary logs.
- 3 Select the applicable action for each condition (for more information, see [Setting default actions for error conditions \(optional\)](#)).

Each of these conditions lets select an action to take for this job, even if you set default actions that are different:

- **Complete with Warnings - Saveset Retained** - The job returns a status of “**Backup Completed with warnings**” and a backup saveset is created that includes the items that were successfully backed up.
  - **Complete without Warnings - Saveset Retained** - The job completes and returns a status of “**Backup Completed**” and a backup saveset is created that includes the items that were successfully backed up.
  - **Fail - Saveset Retained** - The job returns a status of “**Backup Failed.**” However, a backup saveset is generated that includes the items that were successfully backed up.
  - **Fail - No Saveset Retained** - The job returns a status of “**Backup Failed**” and no saveset of backed up objects is kept.
- 4 In the **Mysqldump options** box, list the command options that you want the **mysqldump** utility to use for the job.

The options must begin with a dash or double-dash, and they cannot contain invalid characters (; | < >).

These options are added to the **mysqldump** command first, followed by the options generated internally by the plug-in. Because of this, an option that you enter here that contradicts one generated internally will be overridden by the one generated by the plug-in.

Errors detected by **mysqldump** that cause the job to fail will be embedded in the error log message in the job log.

If you previously set up MySQL Option files to accomplish this task, the options that you enter here are appended to the options specified in the Option files. If you want the plug-in to ignore existing MySQL Option files, enter **--no-defaults** as the first option in this box.

For information on the options supported by your version of **mysqldump**, refer to the applicable MySQL documentation.

**⚠ WARNING:** Be aware that you should *not* use the **--routines (-R)** or **--triggers** option with this feature. (Using these options interferes with backups of database tables, and while backups will succeed, restores might fail.) If there are stored procedures and triggers that need to be backed up for a database, the plug-in will generate **mysqldump** commands internally with the **--routines** and **--triggers** options.

- 5 Click **Save** to save the set.
- 6 In the **Create New Set** dialog, specify a name for the set, and click **Save**.

The name can contain alphanumeric and non-alphanumeric characters, but it cannot include non-English characters. On Linux, the names can have a maximum of 200 characters. On Windows, there is no length restriction. However, a maximum of 40 characters is recommended on all platforms.

### ***To use an existing Backup Options Set***

In the **Plugin Options** list, select the existing Backup Options Set that you want to use.

## Setting backup options for MySQL Enterprise Backup

With the applicable items selected for backup, you can select the type of backup to perform and select a different behavior in case of failure.

### *To create a Backup Options Set*

- 1 Click **Create New** next to the **Plugin Options** list.
- 2 Select the applicable option:
  - **Full Backup** (default selection) - To back up every database and table selected in the current MySQL Instance, select this option.
  - **Incremental Backup** - To back up only the data (for InnoDB tables) or complete tables (for non-InnoDB tables) changed since the last Full or Incremental Backup, select this option.
- 3 Click **Save** to save the set.
- 4 In the **Create New Set** dialog, specify a name for the set, and click **Save**.


The name can contain alphanumeric and non-alphanumeric characters, but it cannot include non-English characters. On Linux, the names can have a maximum of 200 characters. On Windows, there is no length restriction. However, a maximum of 40 characters is recommended on all platforms.

### *To use an existing Backup Options Set*

In the **Plugin Options** list, select the existing Backup Options Set that you want to use.

## Finalizing and submitting the job

### *To finalize and submit a backup job*

- 1 Use the **Schedule**, **Target Storage**, and **Advanced Options** lists to configure any additional required options.
  - 2 Click **Save** or **Save & Submit**, whichever is applicable.
-  **TIP:** To run a job that you have already created and saved, select **Manage Job Definitions** in the Navigation pane, select the applicable job, and click **Run Now**.



# Restoring data

- [Restoring data - an overview](#)
- [Restoring data in MySQL](#)
- [Using advanced restore procedures for MySQL Standard/Community](#)

## Restoring data - an overview

This section outlines the plug-in's restore process and describes all the functionality available for use. In addition, [Examples of restore scenarios for MySQL Standard/Community](#) and [Examples of restore scenarios for MySQL Enterprise Backup](#) offer examples of the various types of restore. Dell recommends that you review these sections to ensure that you understand the functionality available and how it applies to the various types of restore.

## Reviewing the available restore methods for MySQL Standard/Community

To perform a restore successfully, you must understand the types of restore that are available for use.

### Full or Individual Database/Table Copy Only Restores

When Plug-in for MySQL performs a Full or Individual Database/Table Copy Only backup, MySQL's `mysqldump` utility is used to stream the SQL statements that are used to create and populate the tables, directly to the backup media. When the plug-in restores one of these forms of backup, the SQL statements are read directly from the backup media, and executed automatically.

### Incremental or Differential Restores

When the plug-in performs Incremental or Differential Backups, MySQL's Binary Log Index is used to determine which Binary Logs need to be copied to the backup media. When these backups are restored, the Binary Logs are restored to a temporary directory, "`NETVAULT_HOME/tmp/MySQL`". The `mysqlbinlog` utility is then used to generate SQL statements for each transaction that was recorded in the Binary Logs. These statements are then executed automatically. This process is referred to as "applying Binary Logs."

During Incremental and Differential Restores, all transactions recorded in the Binary Logs can be applied, or they can be applied up to a particular time (PIT Recovery). PIT Recovery is useful when trying to recover up to the point directly preceding data corruption, such as a developer accidentally dropping a table or executing an incorrect update.

### Time-based Point-in-Time (PIT) Recovery

PIT Recovery can be performed on the Binary Logs that are to be restored during an Incremental or Differential Restore. Time-based PIT Recovery is useful when the time that the data corruption occurred is known. For example, if a developer dropped a table at 6:00 A.M., PIT Recovery can be performed with a stop time of 5:55 A.M.

Time-based PIT Recovery is typically a one-step process:

Restore the Binary Logs from the Incremental or Differential Backup by selecting the **Restore and Apply Binary Logs** option on the **Options** tab, and specifying the stop time to be just *before* the unwanted transaction.

## Position-based Point-in-Time (PIT) Recovery

When the actual time of the data corruption is unknown, or a more precise recovery is required, a position-based PIT Recovery should be used. For example, if a developer dropped a table from the database, but does not know the exact time when the table was dropped, a position-based PIT Recovery should be used.

Position-based PIT Recovery is a three-step process:

- 1 Restore the Binary Logs from the Incremental or Differential Backup to a temporary directory on the MySQL server by selecting the **Restore Binary Logs to Temporary Directory to Identify Time or Position** option on the **Options** tab.
- 2 Use MySQL's `mysqlbinlog` utility to identify the specific position of the unwanted transaction. For more information, refer to the Point-in-Time Recovery section of the *MySQL Reference Guide*.
- 3 Restore the same Incremental or Differential Backup again; however, select the **Apply Binary Logs from Temporary Directory** restore option, and specify the stop position that exists right before the unwanted transaction.

# Restoring data in MySQL

A standard restore with Plug-in for MySQL includes the steps outlined in the following sections.

- [Selecting data for restore](#)
- [Setting restore options](#)
- [Finalizing and submitting a job](#)
- [Examples of restore scenarios for MySQL Standard/Community](#)
- [Examples of restore scenarios for MySQL Enterprise Backup](#)

## Selecting data for restore

### *To select data to restore*

- 1 In the Navigation pane, click **Create Restore Job**.
- 2 On the **Create Restore Job - Choose Saveset** page, select **Plug-in for MySQL** from the **Plugin Type** list.
- 3 To filter the items displayed in the saveset table further, use the **Client**, **Date**, and **Job ID** lists.  
The table displays the saveset name (Job Title and Saveset ID), creation date and time, and size. By default, the list is sorted alphabetically by saveset name.
- 4 In the saveset table, select the applicable item.  
When you select a saveset, the following details are displayed in the **Saveset Information** area: Job ID, Job Title, name of the NetVault Backup Server, name of the client from which the data was backed up, plug-in used to create the saveset, saveset creation date and time, saveset retirement setting, whether it is an Incremental Backup, whether it is an Archive, and saveset size.
- 5 To continue, click **Next**.
- 6 On the **Create Selection Set** page, select the data that you want to restore.

The first selectable node for inclusion in the restore will vary, based on the type of backup that is being recovered:

- **Full or Individual Database/Table Copy Only Backups** - The root node is listed as “**All Databases**” –because the actual database/table data was included in the backup.
    - ① **IMPORTANT:** Although the root node is entitled “**All Databases,**” it does not account for all the databases that currently exist for a target MySQL Instance. Selection of this will only restore all the data items that were actually selected for the backup job (that is, by selecting this node for a restore, you will not be performing a restore of all the databases that currently exist in a MySQL Instance—only those actually included in the backup).
  - **Incremental or Differential Backups** - The root node is listed as “**Binary Logs**” –because the transactions (Binary Logs) that occurred since the previous backup was performed are included in this form of backup.
- 7 If a more granular restore is desired, double-click the root node to open it and reveal the individual databases that were included in the backup.

As well, an individual database can be opened to reveal its tables for selection.

- ① **IMPORTANT:** MySQL uses multiple file-formats to storage database information. Make sure that you include the .frm files in the restore process to ensure that the restored database is functional.

## Setting restore options

The options displayed on the **Options** tab depends on whether you use the **MySQL Standard/Community** option or **MySQL Enterprise Backup** option.

- [Setting restore options for MySQL Standard/Community](#)
- [Setting restore options for MySQL Enterprise Backup](#)

## Setting restore options for MySQL Standard/Community

On the **Create Selection Set** page, click **Edit Plugin Options**, and configure the applicable parameters on the **Point-in-Time Recovery** and **Restore Destination** tabs. The options displayed depend on the type of backup selected for restore.

- [Full or Individual database restore options](#)
- [Incremental or Differential database restore options](#)

### Full or Individual database restore options

*To restore either a Full Backup or an Individual Database/Table Copy Only Backup*

- 1 Use the following guidelines to select the applicable options on the **Point-in-Time Recovery** tab.
  - **Perform PIT Recovery on Current Binary Logs** - Select this option to perform a **Point-in-Time** form of restore of the selected data objects using the Binary Logs currently residing in the MySQL Binary Log directory on the MySQL server. After this option is enabled, all remaining options on this tab will be made available.
  - **Point In Time Type** - In this section, select the applicable form of PIT Recovery:
    - **Time Based PIT** (default selection) - Select this option to restore the selected data *to a specified time* (described in [Time-based Point-in-Time \(PIT\) Recovery](#)). With this option selected, the **Time Based PIT Details** section is made available.
    - **Position Based PIT** - Select this option to restore the selected data to a *specific stop position that exists right before an unwanted transaction* (described in [Position-based Point-in-Time \(PIT\) Recovery](#)). With this option selected, the **Position Based PIT Details** section is made available.

- **Time Based PIT Details** - If you selected **Time Based PIT**, select the applicable options:
  - **Enable Recovery Prior to Erroneous/Bad SQL Statement(s)** - To restore all transactions that occurred *prior* to the unwanted transaction, select this option. If you select only this option, all transactions that occurred *after* the time specified here will be lost. Using a 24-hour time format, specify the applicable date and time in the associated **Stop Date/Time** fields.
  - **Enable Recovery After Erroneous/Bad SQL Statement(s)** - To restore all transactions that occurred *after* to the unwanted transaction, select this option. If you select only this option, all transactions that occurred *before* the time specified here will be lost. Using a 24-hour time format, specify the applicable date and time in the associated **Start Date/Time** fields. With a specific start date and time selected, you can also set a stop date and time for transactions:
    - **None** (default selection) - Leave this option selected if you want to recover all transactions that occurred after the specified date and time.
    - **Specific Date** - If you only want to include transactions that occurred during a specific range of time, select this option, and enter the applicable stop time in the associated fields (using the 24-hour time format).

① **IMPORTANT:** When PIT Recovery is enabled on both restored and current Binary Logs, you do not have to determine whether the stop time is located in the restored binary logs or the current binary logs. MySQL will automatically stop and start at the specified time and ignore all the Binary Logs after the specified stop time.

You can use both of these options, especially if there is a specific time range in which unwanted transactions occurred. For example, if data that was collected between 11:00 A.M. and 11:15 A.M. on January 29, 2007, was not wanted, the **Enable Recovery Prior to ...** option would be enabled, and “11:00” - “29 Jan 2007” would be input as the **Stop Date/Time**. In addition, the **Enable Recovery After ...** option would be enabled and “11:15” - “29 Jan 2007” would be input in as the **Start Date/Time**. As a result, all transactions that occurred between 11:00 and 11:15 on January 29, 2007, would be omitted from the restore.

- **Position Based PIT Details** - If you selected **Position Based PIT**, select the applicable options:
  - **Enable Recovery Prior to Erroneous/Bad SQL Statement(s)** - To restore all transactions that occurred *prior* to the unwanted transaction, select this option. If you select only this option, all transactions that occurred *after* the position specified here will be lost. This option offers the following associated options:
    - **Stop Position** - Enter the position in the Binary Log *prior* to the unwanted transaction. For example, if the position of the unwanted transaction is 805, enter 804.
    - **Binary Log Containing Stop Position** - Use this list to select the specific Binary Log that contains the stop position specified in the **Stop Position**. If a different file is desired (or the applicable file is not listed), select **OTHER**, and enter the applicable file name in the text box.
  - **Enable Recovery After Erroneous/Bad SQL Statement(s)** - To restore all transactions that occurred *after* the unwanted transaction, select this option. If you select only this option, all transactions that occurred *before* the position specified here will be lost. This option also offers the following associated options:
    - **Start Position** - Enter the position in the Binary Log *after* the unwanted transaction. For example, if the position of the unwanted transaction is 805, enter 806.
    - **Binary Log Containing Start Position** - Use this list to select the specific Binary Log that contains the start position specified in the **Start Position**. If a different file is desired (or the applicable file is not listed), select **OTHER**, and enter the applicable file name in the text box.

- **Stop Position: None** (default selection)- Leave this option selected if you want *all* transactions recovered that occurred after the specified **Start Position**.
- **Stop Position: Specific Position** - If you only want to include transactions that occurred between a specific range of Binary Log positions, select this option, enter the applicable stop position, and select the applicable Binary Log in the **Binary Log Containing Stop Position** list (if a different file should be used, select **OTHER**, and enter the file name). Only transactions that occurred between the positions specified in the **Start Position** and the **Specific Position** fields will be included in the restore.

**IMPORTANT:** You can use both of these options, especially if there is a specific range of positions in which unwanted transactions occurred. For example, if data that was collected between position 805 and 810 contained unwanted transactions, the **Enable Recovery Prior to ...** option would be enabled, and “805” would be input as the **Stop Position** (and its associated options would be configured to call out the Binary Log). In addition, the **Enable Recovery After ...** option would be enabled and “810” would be input in as the **Start Position** (and its associated options would be configured to call out the Binary Log). As a result, all transactions that were logged in the specified Binary Log between 805 and 810 would be omitted from the restore. Also, Stop and Start positions must be *actual positions* listed in a Binary Log, not arbitrary numbers that are greater than the position of the unwanted transaction.

- 2 Use the following guidelines to select the applicable options on the **Restore Destination** tab.
  - **Restoring to the same MySQL Instance** - If the restore targets the same instance that was originally backed up, leave these fields blank. NetVault Backup will use the values set in the **Configure** dialog (for more information, see [Configuring the plug-in](#)).
  - **Restoring to a different MySQL Instance** - If you intend to relocate a restore of the selected data to a different instance, enter the applicable information in the **Username** and **Password** fields that will allow access to the new instance. Additionally, enter the NetVault Backup name established for the new instance in the **Instance Name** field (this is the name established as the **MySQL Instance Name** in the **Configure** dialog; for more information, see [Configuring the plug-in](#)).

**IMPORTANT:** Before attempting a relocation restore to a different MySQL Instance, review [Recovering to an alternate MySQL Server](#).

## Incremental or Differential database restore options

### To restore either an Incremental or Differential Backup

- 1 Use the following guidelines to select the applicable options on the **Point-in-Time Recovery** tab.
  - **Perform PIT Recovery** - Select this option to perform a **Point-in-Time** form of restore of the selected data items. After this option is enabled, all remaining options on this tab will be made available.

Incremental and Differential Restores use the Binary Logs to complete a restore. Therefore, when restoring this form of backup, you must determine how the Binary Logs associated with the selected databases are to be recovered. Select one of the following methods:

- **Restore and Apply Binary Logs (Used when Time or Position is already known)** - If the time or position at which corruption occurred is known, select this option to restore the Binary Logs from the backup device *and* apply the recorded transactions in one restore job. If you also want to perform a PIT Recovery on the Binary Logs currently residing in the MySQL Binary Log directory, select the **Include Current Binary Logs** check box. This will be performed *after* any Binary Log transactions that were saved in the Incremental/Differential Backup are restored and applied.
- **Restore Logs to Temporary Directory to Identify Time or Position** - To restore only the Binary Logs associated with the selected Incremental/Differential Backup to a *temporary directory* on the MySQL Server (that is, “NETVAULT\_HOME/tmp/MySQL/”), select this

option. This lets you use the `mysqlbinlog` utility to review the recovered logs to identify the time/position of the data corruption.

- **Apply Binary Logs from Temporary Directory** - If you previously used the **Restore Logs to Temporary Directory to Identify Time or Position** option, and you used the `mysqlbinlog` utility to identify the corrupted data that is to be omitted from the restore, select this option. This will apply the Binary Logs that were restored to the temporary directory. If you also want to perform a PIT Recovery on the Binary Logs currently residing in the MySQL Binary Log directory, select the **Include Current Binary Logs** check box. This will be performed *after* the Binary Log transactions that exist in the temporary directory are restored and applied.
- **Point In Time Type** - With the **Perform PIT Recovery** option enabled, you must select the applicable form of PIT Recovery:
  - **Time Based PIT** (default selection) - Select this option to restore the selected data to a *specified time* (described in [Time-based Point-in-Time \(PIT\) Recovery](#)). With this option selected, the **Time Based PIT Details** section is made available.
  - **Position Based PIT** - Select this option to restore the selected data to a *specific stop position that exists right before an unwanted transaction* (described in [Position-based Point-in-Time \(PIT\) Recovery](#)). With this option selected, the **Position Based PIT Details** section is made available.
- **Time Based PIT Details** - If you selected **Time Based PIT**, select the applicable options:
  - **Enable Recovery Prior to Erroneous/Bad SQL Statement(s)** - To restore all transactions that occurred *prior* to the unwanted transaction, select this option. If you select only this option, all transactions that occurred *after* the time specified here will be lost. Using a 24-hour time format, specify the applicable date and time in the associated **Stop Date/Time** fields.
  - **Enable Recovery After Erroneous/Bad SQL Statement(s)** - To restore all transactions that occurred *after* the unwanted transaction select this option. If you select only this option, all transactions that occurred *before* the time specified here will be lost. Using a 24-hour time format, specify the applicable date and time in the associated **Start Date/Time** fields. With a specific start date and time selected, you can also set a stop date and time for transactions:
    - **None** (default selection) - Leave this option selected if you want to recover all transactions that occurred after the specified date and time.
    - **Specific Date** - If you only want to include transactions that occurred during a specific range of time, select this option, and enter the applicable stop time in the associated fields (using the 24-hour time format).

① **IMPORTANT:** You can use both of these options, especially if there is a specific time range in which unwanted transactions occurred. For example, if data that was collected between 11:00 A.M. and 11:15 A.M. on January 29, 2007, was not wanted, the **Enable Recovery Prior to ...** option would be enabled, and “11:00” - “29 Jan 2007” would be input as the **Stop Date/Time**. In addition, the **Enable Recovery After ...** option would be enabled and “11:15” - “29 Jan 2007” would be input in as the **Start Date/Time**. As a result, all transactions that occurred between 11:00 and 11:15 on January 29, 2007, would be omitted from the restore.

- **Position Based PIT Details** - If you selected **Position Based PIT**, select the applicable options:
  - **Enable Recovery Prior to Erroneous/Bad SQL Statement(s)** - To restore all transactions that occurred *prior* to the unwanted transaction, select this option. If you select only this option, all transactions that occurred *after* the position specified here will be lost. This option offers the following associated options:
    - **Stop Position** - Enter the position in the Binary Log *prior* to the unwanted transaction. For example, if the position of the unwanted transaction is 805, enter 804.

- **Binary Log Containing Stop Position** - Use this list to select the specific Binary Log that contains the stop position specified in the **Stop Position**. If a different file is desired (or the applicable file is not listed), select **OTHER**, and enter the applicable file name in the text box.
- **Enable Recovery After Erroneous/Bad SQL Statement(s)** - To restore all transactions that occurred *after* to the unwanted transaction, select this option. If you select only this option, all transactions that occurred *before* the position specified here will be lost. This option also offers the following associated options:
  - **Start Position** - Enter the position in the Binary Log *after* the unwanted transaction. For example, if the position of the unwanted transaction is 805, enter 806.
  - **Binary Log Containing Start Position** - Use this list to select the specific Binary Log that contains the start position specified in the **Start Position**. If a different file is desired (or the applicable file is not listed), select **OTHER**, and enter the applicable file name in the text box.
  - **Stop Position: None** (default selection) - Leave this option selected if you want *all* transactions recovered that occurred after the specified **Start Position**.
  - **Stop Position: Specific Position** - If you only want to include transactions that occurred between a specific range of Binary Log positions, select this option, enter the applicable stop position, and select the applicable Binary Log in the **Binary Log Containing Stop Position** list (if a different file should be used, select **OTHER**, and enter the file name). Only transactions that occurred between the positions specified in the **Start Position** and the **Specific Position** fields will be included in the restore.

① **IMPORTANT:** You can use both of these options, especially if there is a specific range of positions in which unwanted transactions occurred. For example, if data that was collected between position 805 and 810 contained unwanted transactions, the **Enable Recovery Prior to ...** option would be enabled, and “805” would be input as the **Stop Position** (and its associated options would be configured to call out the Binary Log). In addition, the **Enable Recovery After ...** option would be enabled and “810” would be input in as the **Start Position** (and its associated options would be configured to call out the Binary Log). As a result, all transactions that were logged in the specified Binary Log between 805 and 810 would be omitted from the restore. Also, Stop and Start positions must be *actual positions* listed in a Binary Log, not arbitrary numbers that are greater than the position of the unwanted transaction.

## 2 Use the following guidelines to select the applicable options on the **Restore Destination** tab.

This tab contains the **Restore Destination Details** section which is comprised of three fields. These fields are used to input account information to allow restore access to the target instance of MySQL. Based on the desired restore type, use these options as follows:

- **Restoring to the same MySQL Instance** - If the restore targets the same instance that was originally backed up, leave these fields blank. NetVault Backup will use the values set in the **Configure** dialog (for more information, see [Configuring the plug-in](#)).
- **Restoring to a different MySQL Instance** - If you intend to relocate a restore of the selected data to a different instance, enter the applicable information in the **Username** and **Password** fields that will allow access to the new instance. Additionally, enter the NetVault Backup name established for the new instance in the **Instance Name** field (this is the name established as the **MySQL Instance Name** in the **Configure** dialog; for more information, see [Configuring the plug-in](#)).

① **IMPORTANT:** Before attempting a relocation restore to a different MySQL Instance, review [Recovering to an alternate MySQL Server](#).

# Setting restore options for MySQL Enterprise Backup

On the **Create Selection Set** page, click **Edit Plugin Options**, and configure the applicable parameters on the **Options** tab:

**IMPORTANT:** Before you perform a restore, make sure that the default NetVault Backup **Temporary Directory** has sufficient space to accommodate (at least temporarily) all the data included in a Full Backup that was created using the **MySQL Enterprise Backup** option. You can use the **General** option to change the default setting to a location that provides sufficient space; you can even use a mapped drive, network file system (NFS), or SMB mount. In the Navigation pane, click **Change Settings**, click **Server Settings**, and then click **General** in the **System and Security** section.

- **Full Restore** - Select the applicable options.
  - **Restore, Extract Raw Full Backup...** (default selection) - Select this option to restore a Full Backup to a temporary location that mirrors the MySQL Server data repository directory hierarchy. This option assumes that you know which backup to restore; if you do not, you can use the next two options.
  - **Restore Full Backup Image to Temp File** - Select this option if you need to list the contents of the backup to determine which backup you need to execute the next option.
  - **Extract Raw Full Backup from Temp File...** - Select this option after you have used the results the preceding option, **Restore Full Backup Image to Temp File**, to determine which backup you need to restore. This option restores the Full Backup to a temporary location that mirrors the MySQL Server data repository directory hierarchy.
  - **Shutdown MySQL Server and Copy Back...** - Select this option when you are ready to shut down the MySQL Server and copy the restored contents from the temporary location back to the original location.
  - **Validate Backup Image** - To instruct the plug-in to run the validate command against the extracted data, select this check box.
  - **List Backup Image** - To list the contents of the backup in the output log, select this check box.
- **Incremental Restore** - Select the applicable options.
  - **Restore, Extract Incremental Backup...** (default selection) - Select this option to restore an Incremental Backup. This option assumes that you know which backup to restore; if you do not, you can use the next two options.
  - **Restore Incremental Backup Image to Temp File** - Select this option if you need to list the contents of the backup to determine which backup you need to execute the next option.
  - **Extract Incremental Backup from Temp File...** - Select this option after you have used the results the preceding option, **Restore Incremental Backup Image to Temp File**, to determine which backup you need to restore.
  - **Shutdown MySQL Server and Copy Back...** - Select this option when you are ready to shut down the MySQL Server and copy the restored contents from the temporary location back to the original location.
  - **Validate Backup Image** - To instruct the plug-in to run the validate command against the extracted data, select this check box.
  - **List Backup Image** - To list the contents of the backup in the output log, select this check box.



## Finalizing and submitting a job

The final steps include setting additional options on the Schedule, Source Options, and Advanced Options pages, submitting the job, and monitoring the progress via the Job Status and View Logs pages. These pages and options are common to all NetVault Backup Plug-ins. For more information, refer to the *Dell NetVault Backup Administrator's Guide*.

### To finalize and submit a restore job

- 1 Click **Ok** to save the settings, and then click **Next**.

- 2 In **Job Name**, specify a name for the job if you do not want to use the default setting.

Assign a descriptive title that lets you easily identify the job for monitoring its progress. The job name can contain alphanumeric and non-alphanumeric characters, but it cannot include non-English characters. On Linux, the names can have a maximum of 200 characters. On Windows, there is no length restriction. However, a maximum of 40 characters is recommended on all platforms.

- IMPORTANT:** Do not use special characters that are not supported in a file name on the target OS. For example, the characters /, \, \*, @, and so on, should not be used on Windows. (This is because Plug-in for MySQL tries to create a folder with the same name as the Job Title for restoring data temporarily.)

- 3 In the **Target Client** list, select the machine on which you want to restore the data.

- TIP:** You can also click **Choose**, and then locate and select the applicable client in the **Choose the Target Client** dialog.

- 4 Use the **Schedule**, **Source Options**, and **Advanced Options** lists to configure the any additional required options.

- 5 Click **Submit** to submit the job for scheduling.

You can monitor progress on the **Job Status** page and view the logs on the **View Logs** page. For more information, refer to the *Dell NetVault Backup Administrator's Guide*.

## Examples of restore scenarios for MySQL Standard/Community

To recover successfully from a failure or data corruption, various settings must be made when setting up the job in regards to data selected for restore and options available on the **Options** tab. The following sections offer examples of various types of recovery and cover the specific options required.

- [Full Backup only restore scenarios](#)
- [Full and Incremental Backup restore scenarios](#)
- [Full and Differential Backup restore scenarios](#)

### Full Backup only restore scenarios

In the following examples, the MySQL DBA has established a backup policy in which Full Backups are performed daily at 11:00 P.M.

#### Full Backup restore & time-based Point-in-Time Recovery

On Monday at 9:00 A.M., the DBA learns that users are encountering “**table not found**” errors on the **Orders** table. The DBA subsequently learns that the table no longer exists because a developer unknowingly dropped it at **6:00 A.M. on Monday** prior to the DBA’s arrival at work.

## Method 1: Recovery *prior* to erroneous statement

The DBA decides to recover up to the time *right before* the Drop Table command was issued. This means that the DBA must restore the Sunday's Full Backup, and perform PIT Recovery on the current Binary Logs.

- 1 **Select the Full Restore from Sunday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Sunday's Full Backup.
- 2 **Set specific options on the restore-related Options tab** - The DBA sets the following options:
  - **Perform PIT Recovery on Current Binary Logs** - Selected to enable this form of restore and all associated options.
  - **Time Based PIT** - Selected as the type.
  - **Enable Recovery Prior to Erroneous/Bad SQL Statement(s)** - Selected this option, and set the **Stop Date/Time** to "5:59" and "8 Jan 2007" (that is, one minute prior to 6:00 A.M. on Monday).
- 3 **Submit the job.**

## Method 2: Recovery *prior* to and *after* erroneous statement

The DBA decides to recover up to the time *right before* the Drop Table command was issued. The DBA also wants to recover the transactions that occurred to the remaining tables from the time *after* the erroneous statement was issued, and up until the end of the current Binary Logs. This will ensure that he has recovered as many of the transactions as feasible, in addition to recovering the dropped table.

- 1 **Select the Full Restore from Sunday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Sunday's Full Backup.
- 2 **Set specific options on the restore-related Options tab** - The DBA sets the following options:
  - **Perform PIT Recovery on Current Binary Logs** - Selected to enable this form of restore all associated options.
  - **Time Based PIT** - Selected as the type.
  - **Enable Recovery Prior to Erroneous/Bad SQL Statement(s)** - Selected this option, and set the **Stop Date/Time** to "5:59" and "8 Jan 2007" (that is, one minute prior to 6:00 A.M. on Monday).
  - **Enable Recovery After Erroneous/Bad SQL Statements** - Selected to recover transactions that occurred after the Order table was dropped, and entered a *later* time and date in the **Start Date/Time**. Finally, because the recovery is to be performed to the end of the named Binary Log, the **None** option was selected for the **Stop Date/Time**.

## Full restore and position-based Point-in-Time Recovery

On Monday at 9:00 A.M., the DBA learns that users are encountering "table not found" errors on the **Orders** table. The DBA subsequently learns that the table no longer exists because a developer unknowingly dropped it at **6:00 A.M. on Monday** prior to the DBA's arrival at work.

### Method 1: Recovery *prior* to erroneous statement

The DBA decides to recover up to the time *right before* the Drop Table command was issued. Furthermore, because the DBA wants a more precise recovery than estimating the time at which the developer dropped the table, the DBA chooses to use a position-based recovery. To accomplish this, the DBA must restore Sunday's Full Backup and perform a PIT Recovery on the current Binary Logs.

- 1 **Use the mysqlbinlog utility against the current Binary Logs** - This is performed outside of NetVault Backup to identify the *position* of the Drop Table command that the DBA does not want to restore. (For information on this utility and process, refer to the *MySQL Reference Guide*.) In this process, the DBA identified the Drop Table command as log position "805" in the "MYSQLSVR-bin.000009" Binary Log.
- 2 **Select the Full Restore from Sunday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Sunday's Full Backup.
- 3 **Set specific options on the restore-related Options tab** - The DBA sets the following options:

- **Perform PIT Recovery on Current Binary Logs** - Selected to enable this form of restore and all associated options.
- **Position Based PIT** - Selected as the type.
- **Enable Recovery Prior to Erroneous/Bad SQL Statement(s)** - Selected this option, and set the **Stop Position** to “804” (the position *before* the one identified using `mysqlbinlog`). Set **Binary Log Containing Stop Position** to **OTHER FILE**, and entered the name of the target binary file in the text box (for example, “`MYSQLSVR-bin.000009`”).

ⓘ | **IMPORTANT:** Stop and Start positions must be *actual positions* listed in a Binary Log, not arbitrary numbers that are greater than the position of the unwanted transaction.

4 Submit the job.

## Method 2: Recovery *prior* to and *after* erroneous statement

The DBA decides to recover up to the time *right before* the Drop Table command was issued. The DBA also wants to recover the transactions that occurred to the remaining tables from the time *after* the Orders table was dropped, and up until the end of the current Binary Logs. This will ensure that he has recovered as many of the transactions as feasible, in addition to recovering the dropped table. Furthermore, the DBA wants a more precise recovery, so he decides to use a position-based recovery. To accomplish this, the DBA must restore Sunday’s Full Backup and perform a PIT Recovery on the current Binary Logs.

- 1 **Use the `mysqlbinlog` utility against the current Binary Logs** - This is performed outside of NetVault Backup to identify the *position* of the Drop Table command that the DBA does not want to restore. (For information on this utility and process, refer to the *MySQL Reference Guide*.) In this process, the DBA identified the Drop Table command as log position “805” in the “`MYSQLSVR-PM-bin.000009`” Binary Log.
- 2 **Select the Full Restore from Sunday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Sunday’s Full Backup.
- 3 **Set specific options on the restore-related Options tab** - The DBA sets the following options:
  - **Perform PIT Recovery on Current Binary Logs** - Selected to enable this form of restore and all associated options.
  - **Position Based PIT** - Selected as the type.
  - **Enable Recovery Prior to Erroneous/Bad SQL Statement(s)** - Selected this option, and set the **Stop Position** to “804” (the position *before* the one identified using `mysqlbinlog`). Set **Binary Log Containing Stop Position** to **OTHER FILE**, and entered the name of the target binary file in the text box (for example, “`MYSQLSVR-PM-bin.000009`”).
  - **Enable Recovery After to Erroneous/Bad SQL Statement(s)** - Selected this option, and set the **Start Position** to “806” (the position *after* the one identified using `mysqlbinlog`). Set **Binary Log Containing Start Position** to **OTHER FILE**, and entered the name of the target binary file in the text box (for example, “`MYSQLSVR-bin.000009`”). Finally, because the recovery is to be performed to the end of the named Binary Log, the **None** option was selected for the **Stop Position**.

ⓘ | **IMPORTANT:** Stop and Start positions must be *actual positions* listed in a Binary Log, not arbitrary numbers that are greater than the position of the unwanted transaction.

4 Submit the job.

## Full and Incremental Backup restore scenarios

The DBA has established a backup policy in which **Full Backups** are performed every **Sunday at 11:00 P.M.** and **Incremental Backups** are performed **Monday through Saturday, at 11:00 P.M.** Because the DBA is performing Incremental Backups, the Binary Logs are *deleted* after each Incremental Backup. This makes the overall backup faster, but requires more time and steps when performing the restore.

## Full and Incremental restore only

On Thursday at 9:00 A.M., the DBA learns that users are encountering “**table not found**” errors for the **Orders** table. The DBA subsequently learns that the table no longer exists because a developer unknowingly dropped the table sometime early Thursday, prior to the DBA’s arrival at work.

The DBA decides to perform a complete recovery up to the point of the last Incremental Backup—the backup performed on **Wednesday** night.

### Phase 1: Full restore from Sunday

- 1 **Select the Full Backup performed Sunday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Sunday’s Full Backup.
- 2 **Leave all restore-related Options at their default** - None of these are used.
- 3 **Submit the job, and wait for it to complete.**

### Phase 2: Incremental restore from Monday

- 1 **Select the Incremental Backup performed Monday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Monday’s Incremental Backup.
- 2 **Leave all restore-related Options at their default** - None of these are used.
- 3 **Submit the job, and wait for it to complete.**

### Phase 3: Incremental restore from Tuesday

- 1 **Select the Incremental Backup performed Tuesday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Tuesday’s Incremental Backup.
- 2 **Leave all restore-related Options at their default** - None of these are used.
- 3 **Submit the job, and wait for it to complete.**

### Phase 4: Incremental restore from Wednesday

- 1 **Select the Incremental Backup performed Wednesday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Wednesday’s Incremental Backup.
- 2 **Leave all restore-related Options at their default** - None of these are used.
- 3 **Submit the job.**

## Full restore & time-based Point-in-Time Recovery

In the following examples, a Full/Incremental Backup scenario is in place, and the DBA wants to recover data to a specific time.

### Method 1: Recovery *Prior* to Erroneous Statement Using Restored Binary Logs Only

On Thursday at 9:00 A.M., the DBA learns that users are encountering “**table not found**” errors on the **Orders** table. The DBA subsequently learns that the table no longer exists because a developer unknowingly dropped it at **8:00 P.M. on Wednesday**.

The DBA needs to perform a recovery that restores the database up to the time **right before** the developer dropped the table at **8:00 P.M. on Wednesday**. Therefore, the following phases would be performed:

### Phase 1: Full restore from Sunday

- 1 **Select the Full Backup performed Sunday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Sunday’s Full Backup.
- 2 **Leave all restore-related Options at their default** - None of these are used.
- 3 **Submit the job, and wait for it to complete.**

## Phase 2: Incremental restore from Monday

- 1 Select the **Incremental Backup performed Monday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Monday's Incremental Backup.
- 2 Leave all restore-related **Options at their default** - None of these are used.
- 3 Submit the job, and wait for it to complete.

## Phase 3: Incremental restore from Tuesday

- 1 Select the **Incremental Backup performed Tuesday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Tuesday's Incremental Backup.
- 2 Leave all restore-related **Options at their default** - None of these are used.
- 3 Submit the job, and wait for it to complete.

## Phase 4: Time-based PIT restore from Wednesday

- 1 Select the **Incremental Backup performed Wednesday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Wednesday's Incremental Backup.
- 2 Set specific options on the restore-related **Options tab** - The DBA sets the following options:
  - **Perform PIT Recovery** - Selected to specify PIT Recovery and enable all associated options.
  - **Restore and Apply Binary Logs (Used when Time or Position is already known)** - Selected to specify the Binary Log included in the backup for use.
  - **Time Based PIT** - Selected as the type.
  - **Enable Recovery Prior to Erroneous/Bad SQL Statement(s)** - Selected this option, and set the **Stop Date/Time** to "19:59" and "10 Jan 2007" (that is, one minute prior to 8:00 P.M. on Wednesday).
- 3 Submit the job.

## Method 2: Recovery *prior to and after* erroneous statement using restored Binary Logs only

On Thursday at 9:00 A.M., the DBA learns that users are encountering "table not found" errors on the **Orders** table. The DBA subsequently learns that the table no longer exists because a developer unknowingly dropped it at **8:00 P.M. on Wednesday**.

The DBA decides to recover up to the time *right before* the Drop Table command was issued at 8:00 P.M. The DBA also wants to recover the transactions that occurred to the remaining tables from the time *after* the Orders table was dropped, and up until the end of the backed-up Binary Logs. This will ensure that he has recovered as many of the transactions as feasible, in addition to recovering the dropped table. Therefore, the following phases would be performed:

### Phase 1: Full restore from Sunday

- 1 Select the **Full Backup performed Sunday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Sunday's Full Backup.
- 2 Leave all restore-related **Options at their default** - None of these are used.
- 3 Submit the job, and wait for it to complete.

### Phase 2: Incremental restore from Monday

- 1 Select the **Incremental Backup performed Monday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Monday's Incremental Backup.
- 2 Leave all restore-related **Options at their default** - None of these are used.
- 3 Submit the job, and wait for it to complete.

### Phase 3: Incremental restore from Tuesday

- 1 Select the **Incremental Backup performed Tuesday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Tuesday's Incremental Backup.
- 2 Leave all restore-related **Options at their default** - None of these are used.
- 3 Submit the job, and wait for it to complete.

### Phase 4: Time-based PIT restore from Wednesday

- 1 Select the **Incremental Backup performed Wednesday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Wednesday's Incremental Backup.
- 2 Set specific options on the restore-related **Options tab** - The DBA sets the following options:
  - **Perform PIT Recovery** - Selected to specify PIT Recovery and enable all associated options.
  - **Restore and Apply Binary Logs (Used when Time or Position is already known)** - Selected to specify the Binary Log included in the backup for use.
  - **Time Based PIT** - Selected as the type.
  - **Enable Recovery Prior to Erroneous/Bad SQL Statement(s)** - Selected this option, and set the **Stop Date/Time** to "19:59" and "10 Jan 2007" (that is, one minute prior to 8:00 P.M. on Wednesday).
  - **Enable Recovery After Erroneous/Bad SQL Statements** - Selected to recover transactions that occurred *after* the *Order* table was dropped, entered a *later* time and date in the **Start Date/Time**. Finally, because the recovery is to be performed to the end of the Binary Log included in the backup, the **None** option was selected for the **Stop Date/Time**.
- 3 Submit the job.

### Method 3: Recovery *prior* to erroneous statement using restored and current Binary Logs

On Thursday at 9:00 A.M., the DBA learns that users are encountering "table not found" errors on the **Orders** table. The DBA subsequently learns that the table no longer exists because a developer unknowingly dropped it at **6:00 A.M. on Thursday**.

The DBA needs to perform a recovery that restores the database up to the time *right before* the developer dropped the table at **6:00 A.M. on Thursday**.

### Phase 1: Full restore from Sunday

- 1 Select the **Full Backup performed Sunday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Sunday's Full Backup.
- 2 Leave all restore-related **Options at their default** - None of these are used.
- 3 Submit the job, and wait for it to complete.

### Phase 2: Incremental restore from Monday

- 1 Select the **Incremental Backup performed Monday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Monday's Incremental Backup.
- 2 Leave all restore-related **Options at their default** - None of these are used.
- 3 Submit the job, and wait for it to complete.

### Phase 3: Incremental restore from Tuesday

- 1 Select the **Incremental Backup performed Tuesday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Tuesday's Incremental Backup.
- 2 Leave all restore-related **Options at their default** - None of these are used.
- 3 Submit the job, and wait for it to complete.

#### Phase 4: Time-based PIT restore from Wednesday

- 1 **Select the Incremental Backup performed Wednesday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Wednesday's Incremental Backup.
- 2 **Set specific options on the restore-related Options tab** - The DBA sets the following options:
  - **Perform PIT Recovery** - Selected to specify PIT Recovery and enable all associated options.
  - **Restore and Apply Binary Logs (Used when Time or Position is already known)** - Selected to indicate that the Binary Log included in the backup is to be used.
  - **Include Current Binary Logs** - Selected to use the current Binary Logs to apply entries that occurred between the time the backup was completed on Wednesday, and the issuance of the Drop Table command.
  - **Time Based PIT** - Selected as the type.
  - **Enable Recovery Prior to Erroneous/Bad SQL Statement(s)** - Selected this option, and set the **Stop Date/Time** to "05:59" and "11 Jan 2007" (that is, one minute prior to 6:00 A.M. on Thursday).
- 3 **Submit the job.**

#### Method 4: Recovery *prior* to and *after* erroneous statement using restored and current Binary Logs

On Thursday at 9:00 A.M., the DBA learns that users are encountering "table not found" errors on the **Orders** table. The DBA subsequently learns that the table no longer exists because a developer unknowingly dropped it at **6:00 A.M. on Thursday**.

The DBA decides to recover up to the time *right before* the Drop Table command was issued. The DBA also wants to recover the transactions that occurred to the remaining tables from the time *after* the Orders table was dropped, and up until the end of the current Binary Logs. This will ensure that he has recovered as many of the transactions as feasible, in addition to recovering the dropped table. Therefore, the following phases would be performed:

#### Phase 1: Full restore from Sunday

- 1 **Select the Full Backup performed Sunday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Sunday's Full Backup.
- 2 **Leave all restore-related Options at their default** - None of these are used.
- 3 **Submit the job, and wait for it to complete.**

#### Phase 2: Incremental restore from Monday

- 1 **Select the Incremental Backup performed Monday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Monday's Incremental Backup.
- 2 **Leave all restore-related Options at their default** - None of these are used.
- 3 **Submit the job, and wait for it to complete.**

#### Phase 3: Incremental restore from Tuesday

- 1 **Select the Incremental Backup performed Tuesday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Tuesday's Incremental Backup.
- 2 **Leave all restore-related Options at their default** - None of these are used.
- 3 **Submit the job, and wait for it to complete.**

#### Phase 4: Time-based PIT restore from Wednesday

- 1 **Select the Incremental Backup performed Wednesday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Wednesday's Incremental Backup.

- 2 **Set specific options on the restore-related Options tab** - The DBA sets the following options:
  - **Perform PIT Recovery** - Selected to specify PIT Recovery and enable all associated options.
  - **Restore and Apply Binary Logs (Used when Time or Position is already known)** - Selected to indicate that the Binary Log included in the backup is to be used.
  - **Include Current Binary Logs** - Selected to use the current Binary Logs to apply entries that occurred between the time the backup was completed on Wednesday, and the issuance of the Drop Table command.
  - **Time Based PIT** - Selected as the type.
  - **Enable Recovery Prior to Erroneous/Bad SQL Statement(s)** - Selected this option, and set the **Stop Date/Time** to “05:59” and “11 Jan 2007” (that is, one minute prior to 6:00 A.M. on Thursday).
  - **Enable Recovery After Erroneous/Bad SQL Statements** - Selected to recover transactions that occurred *after* the *Order* table was dropped, entered a *later* time and date in the **Start Date/Time**. Finally, because the recovery is to be performed to the end of the *current* Binary Log, the *None* option was selected for the **Stop Date/Time**.
- 3 **Submit the job.**

## Full restore & position-based Point-in-Time Recovery

In the following examples, a Full/Incremental Backup scenario is in place, and the DBA wants to recover data to a specific time, but use a more definitive method to define the time. This is done using identified “position values” that exist in the MySQL Binary Logs.

### Method 1: Recovery *prior* to erroneous statement using restored Binary Logs only

On Thursday at 9:00 A.M., the DBA learns that users are encountering “**table not found**” errors on the **Orders** table. The DBA subsequently learns that the table no longer exists because a developer unknowingly dropped it at **8:00 P.M. on Wednesday**.

The DBA decides to recover up to the time *right before* the Drop Table command was issued. Furthermore, the DBA wants a more precise recovery, so he decides to use a position-based recovery. To accomplish this, the DBA must restore Sunday’s Full Backup and the subsequent Incrementals performed on Monday and Tuesday, and then perform a position-based PIT Recovery using Wednesday’s Incremental Backup. The following phases illustrate this:

#### Phase 1: Full restore from Sunday

- 1 **Select the Full Backup performed Sunday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Sunday’s Full Backup.
- 2 **Leave all restore-related Options at their default** - None of these are used.
- 3 **Submit the job, and wait for it to complete.**

#### Phase 2: Incremental restore from Monday

- 1 **Select the Incremental Backup performed Monday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Monday’s Incremental Backup.
- 2 **Leave all restore-related Options at their default** - None of these are used.
- 3 **Submit the job, and wait for it to complete.**

#### Phase 3: Incremental restore from Tuesday

- 1 **Select the Incremental Backup performed Tuesday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Tuesday’s Incremental Backup.
- 2 **Leave all restore-related Options at their default** - None of these are used.
- 3 **Submit the job, and wait for it to complete.**



#### Phase 4: Restore backed-up Binary Logs to determine the position of the erroneous statement

In this phase, only the Binary Logs recorded in Wednesday night's Incremental Backup are restored to a temporary location. This lets the DBA locate the specific position in the log that marks when the Orders table was dropped.

- 1 **Select the Incremental Backup performed Wednesday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Wednesday's Incremental Backup.
- 2 **Set specific options on the restore-related Options tab** - The DBA sets the following options:
  - **Perform PIT Recovery** - Selected to enable this form of restore and all associated options.
  - **Restore Logs to Temporary Directory to Identify Time or Position** - Selected to restore only the Binary Logs included in Wednesday night's Incremental Backup.
  - **Time Based PIT** - Selected as the type, but left all options in the **Time Based PIT Details** section *cleared*.
- 3 **Submit the job, and wait for it to complete.**

#### Phase 5: Identify the position of the Drop Table command in the restored Binary Logs

**Use the mysqlbinlog utility against the restored Binary Logs** - This is performed outside of NetVault Backup to identify the *position* of the Drop Table command that the DBA does not want to restore. (For information on this utility and process, refer to the *MySQL Reference Guide*.) In this process, the DBA identified the Drop Table command as log position "805" in the "MYSQLSVR-bin.000009" Binary Log that was restored to the temporary location on the MySQL Server (and both values were noted).

#### Phase 6: Perform the position-based PIT restore

With the position identified from the restored Binary Log, a PIT restore is then performed using the Wednesday's Incremental Backup.

- 1 **Select the Incremental Backup performed Wednesday night** - The DBA again selects the backup saveset on the **Create Restore Job - Choose Saveset** page that corresponds to Wednesday's Incremental Backup.
- 2 **Set specific options on the restore-related Options tab** - The DBA sets the following options:
  - **Perform PIT Recovery** - Selected to enable this form of restore and all associated options.
  - **Apply Binary Logs from Temporary Directory** - Selected to target the Binary Logs that were restored to the temporary location in the last phase of this procedure. Because the restored Binary Log was used to identify the specific position that the Drop Table command occupied, this option is selected to tell the plug-in to use this same Binary Log.
  - **Enable Recovery Prior to Erroneous/Bad SQL Statement(s)** - Selected this option, and set the **Stop Position** to "804" (the position in the Binary Logs that exists *before* the Drop Table command position identified using `mysqlbinlog`). The **Binary Log Containing Stop Position** option was used to select the Binary Log ("MYSQLSVR-bin.000009") that was restored to the temporary directory.
- 3 **Submit the job.**

#### Method 2: Recovery *prior* to and *after* erroneous statement using restored Binary Logs only

On Thursday at 9:00 A.M., the DBA learns that users are encountering "table not found" errors on the Orders table. The DBA subsequently learns that the table no longer exists because a developer unknowingly dropped it at **8:00 P.M. on Wednesday**.

The DBA decides to recover up to the time *right before* the Drop Table command was issued. The DBA also wants to recover the transactions that occurred to the remaining tables from the time *after* the Orders table was dropped, and up until the end of the backed-up Binary Logs. Furthermore, the DBA wants a more precise recovery, so he decides to use a position-based recovery. To accomplish this, the DBA must restore Sunday's Full Backup and the subsequent Incrementals performed on Monday and Tuesday, and then perform a position-based PIT Recovery using Wednesday's Incremental Backup. The following phases illustrate this:

### Phase 1: Full restore from Sunday

- 1 Select the **Full Backup performed Sunday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Sunday's Full Backup.
- 2 Leave all restore-related **Options at their default** - None of these are used.
- 3 Submit the job, and wait for it to complete.

### Phase 2: Incremental restore from Monday

- 1 Select the **Incremental Backup performed Monday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Monday's Incremental Backup.
- 2 Leave all restore-related **Options at their default** - None of these are used.
- 3 Submit the job, and wait for it to complete.

### Phase 3: Incremental restore from Tuesday

- 1 Select the **Incremental Backup performed Tuesday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Tuesday's Incremental Backup.
- 2 Leave all restore-related **Options at their default** - None of these are used.
- 3 Submit the job, and wait for it to complete.

### Phase 4: Restore backed-up Binary Logs to determine the position of the erroneous statement

In this phase, only the Binary Logs recorded in Wednesday night's Incremental Backup are restored to a temporary location. This allows the DBA to locate the specific position in the log that marks when the Orders table was dropped.

- 1 Select the **Incremental Backup performed Wednesday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Wednesday's Incremental Backup.
- 2 Set specific options on the restore-related **Options tab** - The DBA sets the following options:
  - **Perform PIT Recovery** - Selected to enable this form of restore and all associated options.
  - **Restore Logs to Temporary Directory to Identify Time or Position** - Selected to restore only the Binary Logs included in Wednesday night's Incremental Backup.
  - **Time Based PIT** - Selected as the type, but left all options in the **Time Based PIT Details** section *cleared*.
- 3 Submit the job, and wait for it to complete.

### Phase 5: Identify the position of the Drop Table command in the restored Binary Logs


Use the `mysqlbinlog` utility against the restored Binary Logs - This is performed outside of NetVault Backup to identify the *position* of the Drop Table command that the DBA does not want to restore. (For information on this utility and process, refer to the *MySQL Reference Guide*.) In this process, the DBA identified the Drop Table command as log position "805" in the "MYSQSVR-bin.000009" Binary Log that was restored to the temporary location on the MySQL Server (and both values were noted).

### Phase 6: Perform the position-based PIT restore

With the position identified from the restored Binary Logs, the PIT restore is then performed using Wednesday's Incremental Backup.

- 1 Select the **Incremental Backup performed Wednesday night** - The DBA again selects the backup saveset on the **Create Restore Job - Choose Saveset** page that corresponds to Wednesday's Incremental Backup.
- 2 Set specific options on the restore-related **Options tab** - The DBA sets the following options:
  - **Perform PIT Recovery** - Selected to enable this form of restore and all associated options.

- **Apply Binary Logs from Temporary Directory** - Selected to target the Binary Logs that were restored to the temporary location in the last phase of this procedure. Because the restored Binary Log was used to identify the specific position that the Drop Table command occupied, this option is selected to tell the plug-in to use this same Binary Log.
- **Enable Recovery Prior to Erroneous/Bad SQL Statement(s)** - Selected this option, and set the **Stop Position** to “804” (the position in the Binary Logs that exists *before* the Drop Table command position identified using `mysqlbinlog`). The **Binary Log Containing Stop Position** option was used to select the Binary Log (“MYSQSVR-bin.000009”) that was restored to the temporary directory.
- **Enable Recovery After to Erroneous/Bad SQL Statement(s)** - Selected this option, and set the **Start Position** to “806” (the position in the Binary Logs that exists *after* the Drop Table command position identified using `mysqlbinlog`). The **Binary Log Containing Stop Position** option was used to select the Binary Log (“MYSQSVR-bin.000009”) that was restored to the temporary directory. Finally, because the recovery is to be performed to the end of the named Binary Log, the **None** option was selected for the **Stop Date/Time**.

 **IMPORTANT:** Stop and Start positions must be *actual positions* listed in a Binary Log, not arbitrary numbers that are greater than the position of the unwanted transaction.

3 Submit the job.

### Method 3: Recovery *prior* to erroneous statement using restored and Current Binary Logs

On Thursday at 9:00 A.M., the DBA learns that users are encountering “table not found” errors for the **Orders** table. The DBA subsequently learns that the table no longer exists because a developer unknowingly dropped it at **6:00 A.M. on Thursday**.

The DBA needs to perform a recovery that restores the database up to the time *right before* the developer dropped the table at **6:00 A.M. on Thursday**. Furthermore, the DBA wants a more precise recovery, so he decides to use a position-based recovery. To accomplish this, the DBA must restore Sunday’s Full Backup and the subsequent Incrementals performed on Monday and Tuesday, and then perform a position-based PIT Recovery using Wednesday’s Incremental Backup. The following phases illustrate this:

#### Phase 1: Full restore from Sunday

- 1 **Select the Full Backup performed Sunday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Sunday’s Full Backup.
- 2 **Leave all restore-related Options at their default** - None of these are used.
- 3 **Submit the job, and wait for it to complete.**

#### Phase 2: Incremental restore from Monday

- 1 **Select the Incremental Backup performed Monday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Monday’s Incremental Backup.
- 2 **Leave all restore-related Options at their default** - None of these are used.
- 3 **Submit the job, and wait for it to complete.**

#### Phase 3: Incremental restore from Tuesday

- 1 **Select the Incremental Backup performed Tuesday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Tuesday’s Incremental Backup.
- 2 **Leave all restore-related Options at their default** - None of these are used.
- 3 **Submit the job, and wait for it to complete.**

#### Phase 4: Identify the position of the Drop Table command in the current Binary Logs

**Use the `mysqlbinlog` utility against the current Binary Logs** - This is performed outside of NetVault Backup to identify the *position* of the Drop Table command that the DBA does not want to restore. (For information on

this utility and process, refer to the *MySQL Reference Guide*.) In this process, the DBA identified the Drop Table command as log position “805” in the current Binary Log (“MYSQLSVR-bin.000009”).

### Phase 5: Perform the position-based PIT restore

With the position identified from the restored Binary Logs, the PIT restore is then performed using Wednesday’s Incremental Backup.

- 1 **Select the Incremental Backup performed Wednesday night** - The DBA again selects the backup saveset on the **Create Restore Job - Choose Saveset** page that corresponds to Wednesday’s Incremental Backup.
- 2 **Set specific options on the restore-related Options tab** - The DBA sets the following options:
  - **Perform PIT Recovery** - Selected to enable this form of restore and all associated options.
  - **Restore and Apply Binary Logs (Used when Time or Position is already known)** - Selected to tell the plug-in to use the Binary Log included in the backup.
  - **Include Current Binary Logs** - Selected to tell NetVault Backup to use the current Binary Logs to apply all database transactions that occurred *after* Wednesday night’s Incremental Backup. This will recover all transactions that occurred between the completion of the Incremental Backup on Wednesday night, and the time the Drop Table command was issued.
  - **Enable Recovery Prior to Erroneous/Bad SQL Statement(s)** - Selected this option, and set the **Stop Position** to “804” (the position in the current Binary Log that exists *before* the Drop Table command position identified using `mysqlbinlog`). Set **Binary Log Containing Stop Position** to **OTHER FILE**, and entered the name of the current binary file in the text box (for example, “MYSQLSVR-bin.000009”).

### Method 4: Recovery *prior to and after* erroneous statement using restored and current Binary Logs

On Thursday at 9:00 A.M., the DBA learns that users are encountering “**table not found**” errors for the **Orders** table. The DBA subsequently learns that the table no longer exists because a developer unknowingly dropped it at **6:00 A.M. on Thursday**.

The DBA needs to perform a recovery that restores the database up to the time *right before* the developer dropped the table at **6:00 A.M. on Thursday**. Furthermore, the DBA wants a more precise recovery, so he decides to use a position-based recovery. To accomplish this, the DBA must restore Sunday’s Full Backup and the subsequent Incrementals performed on Monday and Tuesday, and then perform a position-based PIT Recovery using Wednesday’s Incremental Backup. The following phases illustrate this:

#### Phase 1: Full restore from Sunday

- 1 **Select the Full Backup performed Sunday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Sunday’s Full Backup.
- 2 **Leave all restore-related Options at their default** - None of these are used.
- 3 **Submit the job, and wait for it to complete.**

#### Phase 2: Incremental restore from Monday

- 1 **Select the Incremental Backup performed Monday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Monday’s Incremental Backup.
- 2 **Leave all restore-related Options at their default** - None of these are used.
- 3 **Submit the job, and wait for it to complete.**

#### Phase 3: Incremental restore from Tuesday

- 1 **Select the Incremental Backup performed Tuesday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Tuesday’s Incremental Backup.
- 2 **Leave all restore-related Options at their default** - None of these are used.

- 3 Submit the job, and wait for it to complete.


#### Phase 4: Identify the position of the Drop Table command in the current Binary Logs

Use the `mysqlbinlog` utility against the current Binary Logs - This is performed outside of NetVault Backup to identify the *position* of the Drop Table command that the DBA does not want to restore. (For information on this utility and process, refer to the *MySQL Reference Guide*.) In this process, the DBA identified the Drop Table command as log position “805” in the current Binary Log (“MYSQSVR-bin.000009”).

#### Phase 5: Perform the position-based PIT restore

With the position identified from the restored Binary Logs, the PIT restore is then performed using Wednesday’s Incremental Backup.

- 1 **Select the Incremental Backup performed Wednesday night** - The DBA again selects the backup saveset on the **Create Restore Job - Choose Saveset** page that corresponds to Wednesday’s Incremental Backup.
- 2 **Set specific options on the restore-related Options tab** - The DBA sets the following options:
  - **Perform PIT Recovery** - Selected to enable this form of restore and all associated options.
  - **Restore and Apply Binary Logs (Used when Time or Position is already known)** - Selected to tell the plug-in to use the Binary Log included in the backup.
  - **Include Current Binary Logs** - Selected to tell NetVault Backup to use the current Binary Logs to apply all database transactions that occurred *after* Wednesday night’s Incremental Backup. This will recover all transactions that occurred between the completion of the Incremental Backup on Wednesday night, and the time the Drop Table command was issued.
  - **Enable Recovery Prior to Erroneous/Bad SQL Statement(s)** - Selected this option, and set the **Stop Position** to “804” (the position in the current Binary Log that exists *before* the Drop Table command position identified using `mysqlbinlog`). Set **Binary Log Containing Stop Position** to **OTHER FILE**, and entered the name of the current binary file in the text box (for example, “MYSQSVR-bin.000009”).
  - **Enable Recovery After to Erroneous/Bad SQL Statement(s)** - Selected this option, and set the **Start Position** to “806” (the position in the current Binary Log that exists *after* the Drop Table command position that was identified using `mysqlbinlog`). Set **Binary Log Containing Stop Position** to **OTHER FILE**, and entered the name of the current binary file in the text box (for example, “MYSQSVR-bin.000009”). Finally, because the recovery is to be performed to the end of the current Binary Log, the **None** option was selected for the **Stop Position**.

 **IMPORTANT:** Stop and Start positions must be *actual positions* listed in a Binary Log, not arbitrary numbers that are greater than the position of the unwanted transaction.

## Full and Differential Backup restore scenarios

The DBA has established a backup policy in which **Full Backups** are performed every **Sunday at 11:00 P.M.** and **Differential Backups** are performed **Monday through Saturday, at 11:00 P.M.** Because the DBA is performing Differential Backups, the Binary Logs are kept after each form of this backup—which creates a longer backup, but allows for a faster overall restore.

### Full & Differential restore only

On Thursday at 9:00 A.M., the DBA learns that users are encountering “**table not found**” errors on the **Orders** table. The DBA subsequently learns that the table no longer exists because a developer unknowingly dropped it sometime early Thursday prior to the DBA’s arrival at work.

The DBA decides to perform a complete recovery up to the point of the last Differential Backup—the backup performed on **Wednesday** night.

### Phase 1: Full restore from Sunday

- 1 Select the **Full Backup performed Sunday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Sunday's Full Backup.
- 2 Leave all restore-related **Options at their default** - None of these are used.
- 3 Submit the job, and wait for it to complete.

### Phase 2: Incremental restore from Wednesday

- 1 Select the **Differential Backup performed Wednesday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Wednesday's Differential Backup.
- 2 Leave all restore-related **Options at their default** - *None* of the options available on the **Options** tab are used.
- 3 Submit the job.

**IMPORTANT:** The DBA does *not* need to restore **Monday** and **Tuesday** night's Differential Backups because, by selecting to perform Differential Backups, each night's backup is cumulative, back to Sunday night's Full Backup (that is, Wednesday night's backup includes all the Binary Logs that were generated on Monday, Tuesday, *and* Wednesday – back to Sunday's Full Backup).

## Full restore & time-based Point-in-Time Recovery

In the following examples, a Full/Differential Backup scenario is in place, and the DBA wants to recover data to a specific time.

### Method 1: Recovery Prior to Erroneous Statement Using Restored Binary Logs Only

On Thursday at 9:00 A.M., the DBA learns that users are encountering “**table not found**” errors on the **Orders** table. The DBA subsequently learns that the table no longer exists because a developer unknowingly dropped it at **8:00 P.M. on Wednesday**.

The DBA needs to perform a recovery that restores the database up to the time *right before* the developer dropped the table at **8:00 P.M. on Wednesday**. Therefore, the following phases would be performed:

### Phase 1: Full Restore from Sunday

- 1 Select the **Full Backup performed Sunday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Sunday's Full Backup.
- 2 Leave all restore-related **Options at their default** - None of these are used.
- 3 Submit the job, and wait for it to complete.

### Phase 2: Time-based PIT restore from Wednesday

- 1 Select the **Differential Backup performed Wednesday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Wednesday's Differential Backup.

**IMPORTANT:** The DBA does *not* need to restore **Monday** and **Tuesday** night's Differential Backups because, by selecting to perform Differential Backups, each night's backup is cumulative, back to Sunday night's Full Backup (that is, Wednesday night's backup includes all the Binary Logs that were generated on Monday, Tuesday, *and* Wednesday – back to Sunday's Full Backup).

- 2 Set specific options on the restore-related **Options** tab - The DBA sets the following options:
  - **Perform PIT Recovery** - Selected to specify PIT Recovery and enable all associated options.
  - **Restore and Apply Binary Logs (Used when Time or Position is already known)** - Selected to specify the Binary Log included in the backup for use.
  - **Time Based PIT** - Selected as the type.

- **Enable Recovery Prior to Erroneous/Bad SQL Statement(s)** - Selected this option, and set the **Stop Date/Time** to “19:59” and “10 Jan 2007” (that is, one minute prior to 8:00 P.M. on Wednesday).

3 Submit the job.

### Method 2: Recovery *prior* to and *after* erroneous statement using restored Binary Logs only

On Thursday at 9:00 A.M., the DBA learns that users are encountering “table not found” errors on the **Orders** table. The DBA subsequently learns that the table no longer exists because a developer unknowingly dropped it at **8:00 P.M. on Wednesday**.

The DBA decides to recover up to the time *right before* the Drop Table command was issued at 8:00 P.M. The DBA also wants to recover the transactions that occurred to the remaining tables from the time *after* the Orders table was dropped, and up until the end of the backed-up Binary Logs. This will ensure that he has recovered as many of the transactions as feasible, in addition to recovering the dropped table. Therefore, the following phases would be performed:

#### Phase 1: Full restore from Sunday

- 1 **Select the Full Backup performed Sunday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Sunday’s Full Backup.
- 2 **Leave all restore-related Options at their default** - None of these are used.
- 3 **Submit the job, and wait for it to complete.**

#### Phase 2: Time-based PIT restore from Wednesday

- 1 **Select the Differential Backup performed Wednesday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Wednesday’s Differential Backup.

**IMPORTANT:** The DBA does *not* need to restore **Monday** and **Tuesday** night’s Differential Backups because, by selecting to perform Differential Backups, each night’s backup is cumulative, back to Sunday night’s Full Backup (that is, Wednesday night’s backup includes all the Binary Logs that were generated on Monday, Tuesday, *and* Wednesday—back to Sunday’s Full Backup).

- 2 **Set specific options on the restore-related Options tab** - The DBA sets the following options:
  - **Perform PIT Recovery** - Selected to specify PIT Recovery and enable all associated options.
  - **Restore and Apply Binary Logs (Used when Time or Position is already known)** - Selected to specify the Binary Log included in the backup for use.
  - **Time Based PIT** - Selected as the type.
  - **Enable Recovery Prior to Erroneous/Bad SQL Statement(s)** - Selected this option, and set the **Stop Date/Time** to “19:59” and “10 Jan 2007” (that is, one minute prior to 8:00 P.M. on Wednesday).
  - **Enable Recovery After Erroneous/Bad SQL Statements** - Selected to recover transactions that occurred *after* the **Order** table was dropped, entered a *later* time and date in the **Start Date/Time**. Finally, because the recovery is to be performed to the end of the restored Binary Log, the **None** option was selected for the **Stop Date/Time**.

3 Submit the job.

### Method 3: Recovery *prior* to erroneous statement using restored and current Binary Logs

On Thursday at 9:00 A.M., the DBA learns that users are encountering “table not found” errors on the **Orders** table. The DBA subsequently learns that the table no longer exists because a developer unknowingly dropped it at **6:00 A.M. on Thursday**.

The DBA needs to perform a recovery that restores the database up to the time *right before* the developer dropped the table at **6:00 A.M. on Thursday**.

### Phase 1: Full restore from Sunday

- 1 Select the **Full Backup performed Sunday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Sunday's Full Backup.
- 2 Leave all restore-related **Options at their default** - None of these are used.
- 3 Submit the job, and wait for it to complete.

### Phase 2: Time-based PIT restore from Wednesday

- 1 Select the **Differential Backup performed Wednesday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Wednesday's Differential Backup.

**IMPORTANT:** The DBA does *not* need to restore **Monday** and **Tuesday** night's Differential Backups because, by selecting to perform Differential Backups, each night's backup is cumulative, back to Sunday night's Full Backup (that is, Wednesday night's backup includes all the Binary Logs that were generated on Monday, Tuesday, *and* Wednesday—back to Sunday's Full Backup).

- 2 Set specific options on the restore-related **Options tab** - The DBA sets the following options:
  - **Perform PIT Recovery** - Selected to specify PIT Recovery and enable all associated options.
  - **Restore and Apply Binary Logs (Used when Time or Position is already known)** - Selected to indicate that the Binary Log included in the backup is to be used.
  - **Include Current Binary Logs** - Selected to use the current Binary Logs to apply entries that occurred between the time the backup was completed on Wednesday, and the issuance of the Drop Table command.
  - **Time Based PIT** - Selected as the type.
  - **Enable Recovery Prior to Erroneous/Bad SQL Statement(s)** - Selected this option, and set the **Stop Date/Time** to "05:59" and "11 Jan 2007" (that is, one minute prior to 6:00 A.M. on Thursday).
- 3 Submit the job.

### Method 4: Recovery *prior* to and *after* erroneous statement using restored and current Binary Logs

On Thursday at 9:00 A.M., the DBA learns that users are encountering "**table not found**" errors on the **Orders** table. The DBA subsequently learns that the table no longer exists because a developer unknowingly dropped it at **6:00 A.M. on Thursday**.

The DBA decides to recover up to the time *right before* the Drop Table command was issued. The DBA also wants to recover the transactions that occurred to the remaining tables from the time *after* the Orders table was dropped, and up until the end of the current Binary Logs. This will ensure that he has recovered as many of the transactions as feasible, in addition to recovering the dropped table. Therefore, the following phases would be performed:

### Phase 1: Full restore from Sunday

- 1 Select the **Full Backup performed Sunday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Sunday's Full Backup.
- 2 Leave all restore-related **Options at their default** - None of these are used.
- 3 Submit the job, and wait for it to complete.

### Phase 2: Time-based PIT restore from Wednesday

- 1 Select the **Differential Backup performed Wednesday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Wednesday's Differential Backup.



**IMPORTANT:** The DBA does *not* need to restore **Monday** and **Tuesday** night's Differential Backups because, by selecting to perform Differential Backups, each night's backup is cumulative, back to Sunday night's Full Backup (that is, Wednesday night's backup includes all the Binary Logs that were generated on Monday, Tuesday, *and* Wednesday—back to Sunday's Full Backup).

2 **Set specific options on the restore-related Options tab** - The DBA sets the following options:

- **Perform PIT Recovery** - Selected to specify PIT Recovery and enable all associated options.
- **Restore and Apply Binary Logs (Used when Time or Position is already known)** - Selected to indicate that the Binary Log included in the backup is to be used.
- **Include Current Binary Logs** - Selected to use the current Binary Logs to apply entries that occurred between the time the backup was completed on Wednesday, and the issuance of the Drop Table command.
- **Time Based PIT** - Selected as the type.
- **Enable Recovery Prior to Erroneous/Bad SQL Statement(s)** - Selected this option, and set the **Stop Date/Time** to "05:59" and "11 Jan 2007" (that is, one minute prior to 6:00 A.M. on Thursday).
- **Enable Recovery After Erroneous/Bad SQL Statements** - Selected to recover transactions that occurred *after* the **Order** table was dropped, entered a *later* time and date in the **Start Date/Time**. Finally, because the recovery is to be performed to the end of the current Binary Log, the **None** option was selected for the **Stop Date/Time**.

3 **Submit the job.**

## Full restore & position-based Point-in-Time Recovery

In the following examples, a Full/Incremental Backup scenario is in place, and the DBA wants to recover data to a specific time, but use a more definitive method to define the time. This is done using identified "position values" that exist in the MySQL Binary Logs.

### Method 1: Recovery *prior* to erroneous statement using restored Binary Logs only

On Thursday at 9:00 A.M., the DBA learns that users are encountering "**table not found**" errors on the **Orders** table. The DBA subsequently learns that the table no longer exists because a developer unknowingly dropped it at **8:00 P.M. on Wednesday**.

The DBA decides to recover up to the time *right before* the Drop Table command was issued. Furthermore, the DBA wants a more precise recovery, so he decides to use a position-based recovery. The following phases illustrate this:

#### Phase 1: Full restore from Sunday

- 1 **Select the Full Backup performed Sunday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Sunday's Full Backup.
- 2 **Leave all restore-related Options at their default** - None of these are used.
- 3 **Submit the job, and wait for it to complete.**

#### Phase 2: Restore backed-up Binary Logs to determine the position of the erroneous statement

In this phase, only the Binary Logs recorded in Wednesday night's Differential Backup are restored to a temporary location. This lets the DBA locate the specific position in the log that marks when the Orders table was dropped.

- 1 **Select the Differential Backup performed Wednesday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Wednesday's Differential Backup.
- 2 **Set specific options on the restore-related Options tab** - The DBA sets the following options:
  - **Perform PIT Recovery** - Selected to enable this form of restore and all associated options.

- **Restore Logs to Temporary Directory to Identify Time or Position** - Selected to restore only the Binary Logs included in Wednesday night's Differential Backup.
- **Time Based PIT** - Selected as the type, but left all options in the **Time Based PIT Details** section *cleared*.

3 Submit the job, and wait for it to complete.

### Phase 3: Identify the position of the Drop Table command in the restored Binary Logs

Use the `mysqlbinlog` utility against the restored Binary Logs - This is performed outside of NetVault Backup to identify the *position* of the Drop Table command that the DBA does not want to restore. (For information on this utility and process, refer to the *MySQL Reference Guide*.) In this process, the DBA identified the Drop Table command as log position "805" in the "MYSQLSVR-bin.000009" Binary Log that was restored to the temporary location on the MySQL Server (and both values were noted).

### Phase 4: Perform the position-based PIT restore

With the position identified from the restored Binary Log, a PIT restore is then performed using Wednesday's Differential Backup.

1 **Select the Differential Backup performed Wednesday night** - The DBA again selects the backup saveset on the **Create Restore Job - Choose Saveset** page that corresponds to Wednesday's Differential Backup.

**IMPORTANT:** The DBA does *not* need to restore **Monday** and **Tuesday** night's Differential Backups because, by selecting to perform Differential Backups, each night's backup is cumulative, back to Sunday night's Full Backup (that is, Wednesday night's backup includes all the Binary Logs that were generated on Monday, Tuesday, *and* Wednesday—back to Sunday's Full Backup).

2 **Set specific options on the restore-related Options tab** - The DBA sets the following options:

- **Perform PIT Recovery** - Selected to enable this form of restore and all associated options.
- **Apply Binary Logs from Temporary Directory** - Selected to target the Binary Logs that were restored to the temporary location in the last phase of this procedure. Because the restored Binary Log was used to identify the specific position that the Drop Table command occupied, this option is selected to tell the plug-in to use this same Binary Log.
- **Enable Recovery Prior to Erroneous/Bad SQL Statement(s)** - Selected this option, and set the **Stop Position** to "804" (the position in the Binary Logs that exists *before* the Drop Table command position identified using `mysqlbinlog`). The **Binary Log Containing Stop Position** option was used to select the Binary Log ("MYSQLSVR-bin.000009") that was restored to the temporary directory.

3 Submit the job.

### Method 2: Recovery *prior* to and *after* erroneous statement using restored Binary Logs only

On Thursday at 9:00 A.M., the DBA learns that users are encountering "table not found" errors on the **Orders** table. The DBA subsequently learns that the table no longer exists because a developer unknowingly dropped it at **8:00 P.M. on Wednesday**.

The DBA decides to recover up to the time *right before* the Drop Table command was issued. The DBA also wants to recover the transactions that occurred to the remaining tables from the time *after* the Orders table was dropped, and up until the end of the backed-up Binary Logs. Furthermore, the DBA wants a more precise recovery, so he decides to use a position-based recovery. The following phases illustrate this:

#### Phase 1: Full restore from Sunday

- 1 **Select the Full Backup performed Sunday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Sunday's Full Backup.
- 2 **Leave all restore-related Options at their default** - None of these are used.
- 3 **Submit the job, and wait for it to complete.**

## Phase 2: Restore backed-up Binary Logs to determine the position of the erroneous statement

In this phase, only the Binary Logs recorded in Wednesday night's Incremental Backup are restored to a temporary location. This allows the DBA to locate the specific position in the log that marks when the Orders table was dropped.

- 1 **Select the Differential Backup performed Wednesday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Wednesday's Differential Backup.
- 2 **Set specific options on the restore-related Options tab** - The DBA sets the following options:
  - **Perform PIT Recovery** - Selected to enable this form of restore and all associated options.
  - **Restore Logs to Temporary Directory to Identify Time or Position** - Selected to restore only the Binary Logs included in Wednesday night's Differential Backup.
  - **Time Based PIT** - Selected as the type, but left all options in the **Time Based PIT Details** section *cleared*.
- 3 **Submit the job, and wait for it to complete.**

## Phase 3: Identify the position of the Drop Table command in the restored Binary Logs

**Use the mysqlbinlog utility against the restored Binary Logs** - This is performed outside of NetVault Backup to identify the *position* of the Drop Table command that the DBA does not want to restore. (For information on this utility and process, refer to the *MySQL Reference Guide*.) In this process, the DBA identified the Drop Table command as log position "805" in the "MYSQLSVR-bin.000009" Binary Log that was restored to the temporary location on the MySQL Server (and both values were noted).

## Phase 4: Perform the position-based PIT restore

With the position identified from the restored Binary Logs, the PIT restore is then performed using Wednesday's Incremental Backup.

- 1 **Select the Differential Backup performed Wednesday night** - The DBA again selects the backup saveset on the **Create Restore Job - Choose Saveset** page that corresponds to Wednesday's Differential Backup.
  - ① **IMPORTANT:** The DBA does *not* need to restore **Monday** and **Tuesday** night's Differential Backups because, by selecting to perform Differential Backups, each night's backup is cumulative, back to Sunday night's Full Backup (that is, Wednesday night's backup includes all the Binary Logs that were generated on Monday, Tuesday, *and* Wednesday – back to Sunday's Full Backup).
- 2 **Set specific options on the restore-related Options tab** - The DBA sets the following options:
  - **Perform PIT Recovery** - Selected to enable this form of restore and all associated options.
  - **Apply Binary Logs from Temporary Directory** - Selected to target the Binary Logs that were restored to the temporary location in the last phase of this procedure. Because the restored Binary Log was used to identify the specific position that the Drop Table command occupied, this option is selected to tell the plug-in to use this same Binary Log.
  - **Enable Recovery Prior to Erroneous/Bad SQL Statement(s)** - Selected this option, and set the **Stop Position** to "804" (the position in the Binary Logs that exists *before* the Drop Table command position identified using `mysqlbinlog`). The **Binary Log Containing Stop Position** option was used to select the Binary Log ("MYSQLSVR-bin.000009") that was restored to the temporary directory.
  - **Enable Recovery After to Erroneous/Bad SQL Statement(s)** - Selected this option, and set the **Start Position** to "806" (the position in the Binary Logs that exists *after* the Drop Table command position identified using `mysqlbinlog`). The **Binary Log Containing Stop Position** option was used to select the Binary Log ("MYSQLSVR-bin.000009") that was restored to the temporary directory. Finally, because the recovery is to be performed to the end of the named Binary Log, the **None** option was selected for the **Stop Position**.
    - ① **IMPORTANT:** Stop and Start positions must be *actual positions* listed in a Binary Log, not arbitrary numbers that are greater than the position of the unwanted transaction.

- 3 Submit the job.

### Method 3: Recovery *prior* to erroneous statement using restored and current Binary Logs

On Thursday at 9:00 A.M., the DBA learns that users are encountering “**table not found**” errors for the **Orders** table. The DBA subsequently learns that the table no longer exists because a developer unknowingly dropped it at **6:00 A.M. on Thursday**.

The DBA needs to perform a recovery that restores the database up to the time **right before** the developer dropped the table at **6:00 A.M. on Thursday**. Furthermore, the DBA wants a more precise recovery, so he decides to use a position-based recovery. The following phases illustrate this:

#### Phase 1: Full restore from Sunday

- 1 **Select the Full Backup performed Sunday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Sunday’s Full Backup.
- 2 **Leave all restore-related Options at their default** - None of these are used.
- 3 **Submit the job, and wait for it to complete.**

#### Phase 2: Identify the position of the Drop Table command in the current Binary Logs

**Use the mysqlbinlog utility against the current Binary Logs** - This is performed outside of NetVault Backup to identify the **position** of the Drop Table command that the DBA does not want to restore. (For information on this utility and process, refer to the *MySQL Reference Guide*.) In this process, the DBA identified the Drop Table command as log position “**805**” in the current Binary Log (“**MYSQLSVR-bin.000009**”).

#### Phase 3: Perform the position-based PIT restore

With the position identified from the restored Binary Logs, the PIT restore is then performed using Wednesday’s Differential Backup.

- 1 **Select the Differential Backup performed Wednesday night** - The DBA again selects the backup saveset on the **Create Restore Job - Choose Saveset** page that corresponds to Wednesday’s Differential Backup.
  - ① **IMPORTANT:** The DBA does *not* need to restore **Monday** and **Tuesday** night’s Differential Backups because, by selecting to perform Differential Backups, each night’s backup is cumulative, back to Sunday night’s Full Backup (that is, Wednesday night’s backup includes all the Binary Logs that were generated on Monday, Tuesday, *and* Wednesday—back to Sunday’s Full Backup).
- 2 **Set specific options on the restore-related Options tab** - The DBA sets the following options:
  - **Perform PIT Recovery** - Selected to enable this form of restore and all associated options.
  - **Restore and Apply Binary Logs (Used when Time or Position is already known)** - Selected to tell the plug-in to use the Binary Log that was included in the backup.
  - **Include Current Binary Logs** - Selected to tell NetVault Backup to use the current Binary Logs to apply all database transactions that occurred **after** Wednesday night’s Differential Backup. This will recover all transactions that occurred between the completion of the Differential Backup on Wednesday night, and the time the Drop Table command was issued.
  - **Enable Recovery Prior to Erroneous/Bad SQL Statement(s)** - Selected this option, and set the **Stop Position** to “**804**” (the position in the current Binary Log that exists **before** the Drop Table command position identified using `mysqlbinlog`). Set **Binary Log Containing Stop Position** to **OTHER FILE**, and entered the name of the current binary file in the text box (for example, “**MYSQLSVR-bin.000009**”).

### Method 4: Recovery *prior* to and *after* erroneous statement using restored and current Binary Logs

On Thursday at 9:00 A.M., the DBA learns that users are encountering “**table not found**” errors for the **Orders** table. The DBA subsequently learns that the table no longer exists because a developer unknowingly dropped it at **6:00 A.M. on Thursday**.

The DBA decides to recover up to the time *right before* the Drop Table command was issued. The DBA also wants to recover the transactions that occurred to the remaining tables from the time *after* the Orders table was dropped, and up until the end of the current Binary Log, as well. Furthermore, the DBA wants a more precise recovery, so he decides to use a position-based recovery. The following phases illustrate this:

### Phase 1: Full restore from Sunday

- 1 **Select the Full Backup performed Sunday night** - On the **Create Restore Job - Choose Saveset** page, the DBA selects the backup saveset that corresponds to Sunday's Full Backup.
- 2 **Leave all restore-related Options at their default** - None of these are used.
- 3 **Submit the job, and wait for it to complete.**

### Phase 2: Identify the position of the Drop Table command in the current Binary Logs

**Use the mysqlbinlog utility against the current Binary Logs** - This is performed outside of NetVault Backup to identify the *position* of the Drop Table command that the DBA does not want to restore. (For information on this utility and process, refer to the *MySQL Reference Guide*.) In this process, the DBA identified the Drop Table command as log position "805" in the current Binary Log ("MYSQLSVR-bin.000009").

### Phase 3: Perform the position-based PIT restore

With the position identified from the restored Binary Logs, the PIT restore is then performed using Wednesday's Differential Backup.

- 1 **Select the Differential Backup performed Wednesday night** - The DBA again selects the backup saveset on the **Create Restore Job - Choose Saveset** page that corresponds to Wednesday's Differential Backup.

**IMPORTANT:** The DBA does *not* need to restore **Monday** and **Tuesday** night's Differential Backups because, by choosing to perform Differential Backups, each night's backup is cumulative, back to Sunday's Full Backup (that is, Wednesday night's backup includes all the Binary Logs that were generated on Monday, Tuesday, *and* Wednesday—back to Sunday's Full Backup).

- 2 **Set specific options on the restore-related Options tab** - The DBA sets the following options:
  - **Perform PIT Recovery** - Selected to enable this form of restore and all associated options.
  - **Restore and Apply Binary Logs (Used when Time or Position is already known)** - Selected to tell the plug-in to use the Binary Log that was included in the backup.
  - **Include Current Binary Logs** - Selected to tell NetVault Backup to use the current Binary Logs to apply all database transactions that occurred *after* Wednesday night's Differential Backup. This will recover all transactions that occurred between the completion of the Differential Backup on Wednesday night, and the time the Drop Table command was issued.
  - **Enable Recovery Prior to Erroneous/Bad SQL Statement(s)** - Selected this option, and set the **Stop Position** to "804" (the position in the current Binary Logs that exists *before* the Drop Table command position identified using `mysqlbinlog`). Set **Binary Log Containing Stop Position** to **OTHER FILE**, and entered the name of the current binary file in the text box (for example, "MYSQLSVR-bin.000009").
  - **Enable Recovery After to Erroneous/Bad SQL Statement(s)** - Selected this option, and set the **Start Position** to "806" (the position in the current Binary Log that exists *after* the Drop Table command position that was identified using `mysqlbinlog`). Set **Binary Log Containing Stop Position** to **OTHER FILE**, and entered the name of the current binary file in the text box (for example, "MYSQLSVR-bin.000009"). Finally, because the recovery is to be performed to the end of the current Binary Log, the **None** option was selected for the **Stop Position**.

**IMPORTANT:** Stop and Start positions must be *actual positions* listed in a Binary Log, not arbitrary numbers that are greater than the position of the unwanted transaction.

# Examples of restore scenarios for MySQL Enterprise Backup

To recover successfully from a failure or data corruption, various settings must be made when setting up the job in regards to data selected for restore and options available on the **Options** tab.

- [Full Backup only restore scenarios](#)
- [Full and Incremental Backup restore scenarios](#)

## Full Backup only restore scenarios

- 1 To generate a prepared Full Backup to be restored, submit a job in which you have selected the **Restore, Extract Raw Full Backup...** option on the **Options** tab.
- 2 To shut down MySQL and copy the prepared Full Backup to the MySQL Server repository, submit a job in which you have selected the **Shutdown MySQL Server and Copy Back...** option on the **Options** tab.
- 3 Restart the MySQL Server by entering the applicable command at a command prompt.

## Full and Incremental Backup restore scenarios

- 1 To generate a prepared Full Backup to be restored, submit a job in which you have selected the **Restore, Extract Raw Full Backup...** option on the **Options** tab.
- 2 To apply the required Incremental Backups to the prepared Full Backup in the order in which they were backed up, submit the applicable number of jobs in which you have selected the **Restore, Extract Incremental Backup...** option on the **Options** tab.
- 3 To shut down MySQL and copy the prepared Full Backup to the MySQL Server repository, submit a job in which you have selected the **Shutdown MySQL Server and Copy Back...** option on the **Options** tab.
- 4 Restart the MySQL Server by entering the applicable command at a command prompt.


# Using advanced restore procedures for MySQL Standard/Community

This section describes other restore operations that you can perform with the plug-in for the **MySQL Standard/Community** option.

- [Renaming a database during a restore](#)
- [Restoring to a different MySQL Instance on the same server](#)
- [Recovering to an alternate MySQL Server](#)

## Renaming a database during a restore

NetVault Backup lets you select a backed up MySQL database and rename it for a restore so that it will not overwrite the existing version of that database. This can be useful when creating a copy of an existing database. To accomplish this, perform the steps outlined in the following sections.

-  **IMPORTANT:** Only complete databases can be renamed for a restore. Attempts to rename an individual table will be met with an error message.

Prior to conducting a restore rename, review this list of known limitations and intended uses for this operation:

- Valid restore sequences are limited to Full or Individual Database/Table Copy Only backups.

- Not allowed during Incremental and Differential Restores.
- Can be used in conjunction with a restore to a different MySQL Instance or MySQL Server.

### To rename the database during restore

- 1 In the Navigation pane, click **Create Restore Job**, select **Plug-in for MySQL** from the **Plugin Type** list, select the applicable saveset, and click **Next**.

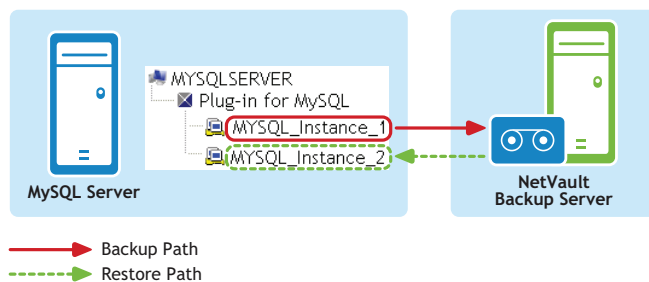
For more information, see [Selecting data for restore](#).

- 2 On the **Create Selection Set** page, select the database that you want to rename.
- 3 From the **Actions** list, select **Rename**.
- 4 In the **Rename/Relocate** dialog, enter the new name in the **Rename to** box, and click **Ok**.  
The database item is accompanied by renaming information in parenthesis.
- 5 Continue with the restore as explained in [Restoring data in MySQL](#).

## Restoring to a different MySQL Instance on the same server

In this form of relocation restore, a Plug-in for MySQL backup is to be restored to the *same* MySQL Server machine, but to a *different instance* of MySQL that has been configured there.

Figure 1. Data backed up on one MySQL Instance and recovered to different instance



To accomplish this, perform the steps outlined in the following sections.

## Known limitations/intended uses

Prior to conducting a relocation restore, review this list of known limitations and intended uses for this operation:

- Valid restore sequences can include Full, Incremental, Differential, and Individual Database/Table Copy Only backups.
- Only restored Binary Logs from an Incremental or Differential Backup can be applied to the destination instance (that is, current Binary Logs from the source instance cannot be applied to the destination instance).

## Prerequisites

The following prerequisites must be met before a restore of this type can be set up and run.

- **Existing and target machines must have the same Installation configurations** - Both machines must have the following established, in regards to MySQL:
  - Same OS installed
  - Same version of MySQL installed
  - Same installation/base directory
  - Same MySQL “Data” directory
- **New target instance must be successfully configured in Plug-in for MySQL** - The process outlined in [Configuring the plug-in](#) must have been successfully performed to add the new MySQL Instance (that is, the target instance must be revealed and accessible within the **Plug-in for MySQL** node on the **NetVault Backup Selections** page).

## Setting up and launching the restore

With the prerequisites met, perform the following steps to set up this form of relocation restore job.

### *To run the restore*

- 1 In the Navigation pane, click **Create Restore Job**.
- 2 On the **Create Restore Job - Choose Saveset** page, select **Plug-in for MySQL** from the **Plugin Type** list.
- 3 To filter the items displayed in the saveset table further, use the **Client**, **Date**, and **Job ID** lists.  
The table displays the saveset name (Job Title and Saveset ID), creation date and time, and size. By default, the list is sorted alphabetically by saveset name.
- 4 In the saveset table, select the applicable item.  
When you select a saveset, the following details are displayed in the **Saveset Information** area: Job ID, Job Title, name of the NetVault Backup Server, name of the client from which the data was backed up, plug-in used to create the saveset, saveset creation date and time, saveset retirement setting, whether it is an Incremental Backup, whether it is an Archive, and saveset size.
- 5 To continue, click **Next**.
- 6 On the **Create Selection Set** page, select the data that you want to restore.  
Display the individual MySQL Instance that was the target of the backup, navigate the selection tree until the applicable data items are located, and select them for inclusion.
- 7 With the applicable databases selected, click **Edit Plugin Options**, and then click the **Restore Destination** tab.
- 8 In the **Restore Destination Details** section, enter the following:
  - **Username** - Enter the logon account name used to access the target MySQL Instance.
  - **Password** - Enter the password associated with the logon account.
  - **Instance Name** - Enter the NetVault Backup name established for the new instance of MySQL, based on what was established during its configuration in NetVault Backup (this is the name established as the **MySQL Instance Name** in the **Configure** dialog; for more information, see [Configuring the plug-in](#)).
- 9 If desired, select the applicable options on the **Point-in-Time Recovery** tab.  
These options are not required to perform this form of restore. For more information, see [Setting restore options](#).
- 10 Click **Ok** to save the settings, and then click **Next**.



11 In **Job Name**, specify a name for the job if you do not want to use the default setting.

Assign a descriptive title that lets you easily identify the job for monitoring its progress. The job name can contain alphanumeric and non-alphanumeric characters, but it cannot include non-English characters. On Linux, the names can have a maximum of 200 characters. On Windows, there is no length restriction. However, a maximum of 40 characters is recommended on all platforms.

**IMPORTANT:** Do not use special characters that are not supported in a file name on the target OS. For example, the characters /, \, \*, @, and so on, should not be used on Windows. (This is because Plug-in for MySQL tries to create a folder with the same name as the Job Title for restoring data temporarily.)

12 In the **Target Client** list, select the machine on which you want to restore the data.

**TIP:** You can also click **Choose**, and then locate and select the applicable client in the **Choose the Target Client** dialog.

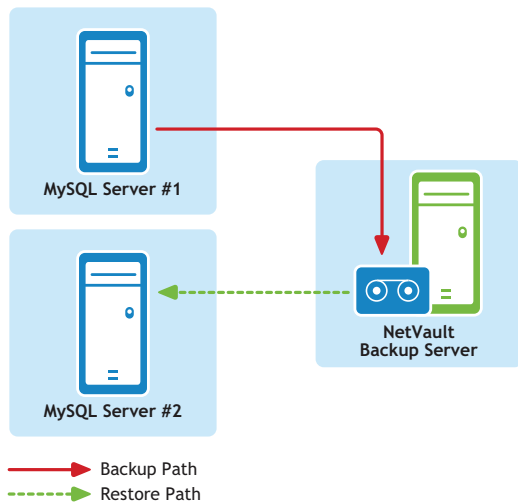
13 Use the **Schedule**, **Source Options**, and **Advanced Options** lists to configure the any additional required options.

14 Click **Submit** to submit the job for scheduling.

## Recovering to an alternate MySQL Server

Similar to Plug-in for MySQL's ability to restore databases or individual table(s) to a different MySQL Instance on the same MySQL Server, a *different MySQL server* can be targeted during the restore process. This is used during disaster recovery operations.

Figure 2. Example of data path for this form of relocation restore



To accomplish this, perform the steps outlined in the following sections.

## Known limitations/intended uses

Prior to conducting a relocation restore to another MySQL server, review this list of known limitations and intended uses for this operation.

- Valid restore sequences can include Full, Incremental, Differential, and Individual Database/Table Copy Only backups
- Only restored Binary Logs from an Incremental or Differential Backup can be applied to the destination instance, that is, current Binary Logs from the source instance cannot be applied to the destination MySQL Instance

## Software installation/configuration prerequisites

The following prerequisites must be met before a restore of this type can be set up and run.

- **Existing and target machines must have the same Installation configurations** - Both machines must have the following established, in regards to MySQL:
  - Same OS installed
  - Same version of MySQL installed
  - Same installation/base directory
  - Same MySQL “Data” directory
- **NetVault Backup software and Plug-in for MySQL installed on all clients** - NetVault Backup (Client/Server version) and the plug-in must be installed and configured on *both* machines in use for this process (that is, the *existing MySQL machine* and the *new restore target*).
- **All Client machines added to the NetVault Backup Server** - With all software installation requirements met, the target NetVault Backup Client machines must be added to the NetVault Backup Server via the NetVault Backup WebUI (that is, the *existing MySQL machine* and the *new restore target*).
- **Instance of MySQL must exist on the new restore target** - The relocation process requires that an instance of MySQL exist on the *new restore target*. This instance will serve as the target of the relocation restore. This instance must be properly set up and configured in MySQL, and you must add it to the plug-in on the new restore target (follow the steps outlined in [Configuring the plug-in](#)).

**IMPORTANT:** Note the following values in the instance’s **Configure** dialog on the new restore target: **Username, Password, and Instance Name.**

During set up of a relocation restore, the plug-in requires that you enter these values on the **Options** tab to gain proper access to the targeted MySQL Instance.

## Running the restore

With the prerequisites met, perform the following steps to restore a MySQL backup to a different machine.

### To run the restore

- 1 In the Navigation pane, click **Create Restore Job**.
- 2 On the **Create Restore Job - Choose Saveset** page, select **Plug-in for MySQL** from the **Plugin Type** list.
- 3 To filter the items displayed in the saveset table further, use the **Client, Date,** and **Job ID** lists.

The table displays the saveset name (Job Title and Saveset ID), creation date and time, and size. By default, the list is sorted alphabetically by saveset name.

- 4 In the saveset table, select the applicable item.

When you select a saveset, the following details are displayed in the **Saveset Information** area: Job ID, Job Title, name of the NetVault Backup Server, name of the client from which the data was backed up, plug-in used to create the saveset, saveset creation date and time, saveset retirement setting, whether it is an Incremental Backup, whether it is an Archive, and saveset size.

- 5 To continue, click **Next**.
- 6 On the **Create Selection Set** page, select the data that you want to restore.

Display the individual MySQL Instance that was the target of the backup, navigate the selection tree until the applicable data items are located, and select them for inclusion.

- 7 With the applicable databases selected, click **Edit Plugin Options**, and then click the **Restore Destination** tab.

- 8 In the **Restore Destination Details** section, enter the following:
  - **Username** - Enter the user name established for the target instance on the *new restore target* (that is, what was set up in the **Username** field of the **Configure** dialog).
  - **Password** - Enter the password established for the target instance on the *new restore target*.
  - **Instance Name** - Enter the NetVault Backup name established for the target instance of MySQL on the *new restore target*.
- 9 If desired, select the applicable options on the **Point-in-Time Recovery** tab.

These options are not required to perform this form of restore. For more information, see [Setting restore options](#).
- 10 Click **Ok** to save the settings, and then click **Next**.
- 11 In **Job Name**, specify a name for the job if you do not want to use the default setting.

Assign a descriptive title that lets you easily identify the job for monitoring its progress. The job name can contain alphanumeric and non-alphanumeric characters, but it cannot include non-English characters. On Linux, the names can have a maximum of 200 characters. On Windows, there is no length restriction. However, a maximum of 40 characters is recommended on all platforms.

**!** **IMPORTANT:** Do not use special characters that are not supported in a file name on the target OS. For example, the characters `/, \, *, @`, and so on, should not be used on Windows. (This is because Plug-in for MySQL tries to create a folder with the same name as the Job Title for restoring data temporarily.)
- 12 In the **Target Client** list, select the machine on which you want to restore the data.

**!** **TIP:** You can also click **Choose**, and then locate and select the applicable client in the **Choose the Target Client** dialog.
- 13 Use the **Schedule**, **Source Options**, and **Advanced Options** lists to configure the any additional required options.
- 14 Click **Submit** to submit the job for scheduling.

# Working with native MySQL replication

- [Using the plug-in in a native environment - an overview](#)
- [Enabling replication support](#)
- [Backing up replication servers](#)
- [Restoring replication servers](#)

## Using the plug-in in a native environment - an overview

When you are using replication, all updates to the tables that are replicated should be performed on the master server. Otherwise, you must avoid conflicts between updates that users make to tables on the master and updates that they make to tables on the slave.

Replication offers benefits for robustness, speed, and system administration:

- Robustness is increased with a master/slave setup. In the event of problems with the master, you can switch to the slave as a backup.
- You can improve response time for clients by splitting the load for processing client queries between the master and slave servers. SELECT queries may be sent to the slave to reduce the query-processing load of the master. Statements that modify data should still be sent to the master so that the master and slave do not get out of synchrony. This load-balancing strategy is effective if non-updating queries dominate, which is the normal case.
- Another benefit of using replication is that you can perform database backups using a slave server without disturbing the master. The master continues to process updates while the backup is being made.

Plug-in for MySQL supports backup and recovery of single-master replication environments.

## Enabling replication support

Replication support is enabled using the **Configure** dialog. For information on accessing this dialog, see [Configuring the plug-in](#).

### *To enable replication*

- **Enable MySQL Replication** - If native MySQL Replication is enabled for this instance, select this check box.
  - **Slave Instance** - If the instance is configured as a **Slave**, select this option.
  - **Master Instance** - If the instance is configured as a **Master**, select this option.
- **Enable Point-In-Time Recovery** - If you want to enable PIT backups and restores, select this check box.
- **Binary Log Index Path** - If you selected the **Enable Point in Time Recovery** check box, use this field to specify the complete path to the Binary Log Index file.

- **Relay Log Index Path** - If you are configuring a Slave Instance, enter the complete path to the Relay Log Index file to include it in backups.

## Backing up replication servers

Support for backing up native MySQL Replication environments has the following limitations:

- **Slave replication servers** - Backup types supported include:
  - Full
  - Incremental
  - Differential
  - Individual Database/Table Copy Only
- **Masters replication servers** - Backup types supported include:
  - Individual Database/Table Copy Only

Incremental and Differential Backups on the slave server will require that you enable the “**--log-slave-updates**” option in MySQL. This option tells the slave to log the updates performed by its SQL thread to its own Binary Log. For this option to work, the slave must also be started with the “**--log-bin**” option to enable Binary Logging. Normally, this option is used to chain replication servers; however, it can also be used for Binary Log backups enabling PIT Recovery of a replicated environment without the complications of purging Binary Logs on the master server before they have been applied to the slaves.

## Replication configuration backups

Using the **Relay Log Index Path** option, you can specify the full path name to the Relay Log Index file to include it in backups. By default, the status files, “**master.info**” and “**relay-log.info**,” reside in the same location. If you use the Relay Log Index Path option and the default file names and locations are retained, all these files are backed up and restored automatically by the plug-in for a slave replication server.

## Restoring replication servers

You can use Full, Incremental, and Differential Backups from the MySQL Replication Slave Instance to perform disaster recovery for the MySQL Replication Master Instance. After the Master Instance has been restored, the DBA can use the same set of backups to restore each Slave Instance to the same level as the Master Instance, and then restart Replication, or the DBA can reinitialize the Slaves Instances using other initialization methods provided in the *MySQL Reference Guide*.

You can use Individual Database/Table backups from both the master and the slave to restore individual databases/tables to the master. If a DBA wants to resynchronize an individual table or database on a slave, Dell recommends that the DBA use MySQL’s Replication process for resynchronization instead of restoring to the slave and then trying to get the slave back in sync with the master.

# Using the plug-in in a Failover Cluster environment

- [MySQL Server Failover Clustering - an overview](#)
- [Installing the plug-in](#)
- [Licensing the plug-in](#)
- [Configuring the plug-in](#)
- [Backing up data](#)
- [Restoring data](#)

## MySQL Server Failover Clustering - an overview

**MySQL Failover Clustering** (Active/Passive) is designed to provide high-availability for an entire MySQL Server instance. For example, you can configure a MySQL Server instance on one node of a failover cluster to fail over to any other node in the cluster during a hardware failure, OS failure, or a planned upgrade.

A failover cluster is a combination of one or more nodes (hosts) with one or more shared disks. Various resources hosted by the nodes, such as IP, shared storage, and an application (MySQL in this case) can be grouped together to create a **Clustered Service**. A Virtual Service appears on the network as if it were a single computer running an application, but provides failover from one node to another node if the current node becomes unavailable.

**IMPORTANT:** In NetVault Backup terminology, a clustered service is accessed by a **Virtual Client**. The references to **Virtual Client** in Plug-in for MySQL are basically references to the **Clustered Service** in the MySQL Server Failover Cluster environment.

Plug-in for MySQL provides support for MySQL Server Failover Clustering. Using the failover cluster network name, the plug-in is able to identify the current node that is in control of the MySQL Server Clustered Service and target it for backup.

This section points out differences between the setup and usage of the plug-in in a Failover Cluster environment vs. a traditional one. It mirrors the sections found in the instructions for the **MySQL Standard/Community** option.

## Important considerations

- Unless outlined in the sections that follow, backups and restores performed with the plug-in of clustered data are the same as those performed with traditional MySQL Server data.
- The following sections only offer information on MySQL-specific settings required for the use of this plug-in in a Failover Cluster environment. They do not offer instructions on how to set up NetVault Backup's **Application Cluster Support** to administer backups and restores of non-MySQL Server-related data and files. This process is not plug-in-specific, and you can find complete details in the *Dell NetVault Backup Administrator's Guide*.

- Before you continue, Dell strongly recommends that you review all cluster-related information provided in the *Dell NetVault Backup Administrator's Guide* to understand how the following information works in conjunction with MySQL Server Failover Cluster functionality.

## Installing the plug-in

To install the plug-in, complete the steps outlined in the following sections.

## Installation prerequisites

The following prerequisites must be met before you install Plug-in for MySQL in a clustered environment:

- **MySQL failover clustering environment in place** - You must have a properly configured MySQL Cluster environment.
  - ① **IMPORTANT:** Support for this feature was tested on Red Hat Enterprise Linux (RHEL) v5.x using the Red Hat Clustering and Clustered Storage Suite, and employing a two-node MySQL (v5.5) cluster configuration with shared storage containing the database data files and logs. If you intend to use clustering in a different configuration, make sure that you test backups and restores before deploying it in a production environment.
- **Separate NetVault Backup Server machine** - The machine that is to serve as the NetVault Backup Server must be properly configured and it *must exist outside* the MySQL Server cluster, but have network connectivity to the nodes (hosts) within the cluster.

## Installing the software

Installation of the plug-in for a clustered environment is different than the traditional installation of this plug-in. This process is completed through the creation of a **Virtual Client** on the NetVault Backup Server. A Virtual Client is a group of nodes within the cluster that are seen by the NetVault Backup Server as a *single* client that is created to back up a single clustered service (for example, a MySQL Clustered Service). During the Virtual Client creation process, the plug-in is transferred from the NetVault Backup Server to selected nodes within a cluster and installed there.

## Creating a Virtual Client

As noted earlier, the Virtual Client creation process is not plug-in-specific, and you can find complete details in the *Dell NetVault Backup Administrator's Guide*. However, consider the following points during the Virtual Client creation process:

- **Assign a name to the Virtual Client** - Dell strongly recommends that the network name assigned to the MySQL Clustered Service be used as the NetVault Backup Virtual Client name. When a Virtual Client is browsed, NetVault Backup will locate the node currently in control of the clustered service, and reveal the MySQL Server instance (for example, on the **NetVault Backup Selections** page). With a Virtual Client name set up as the MySQL Clustered Service network name, it is easier to recognize the MySQL Server instance for which the Virtual Client was created.
- **Only include relevant cluster nodes in the Virtual Client** - The hosts that are to be included in the creation of a Virtual Client should *only* be those nodes within the cluster that are relevant to the MySQL Clustered Service that is to be backed up and restored.

After the creation of the Virtual Client, the plug-in is transferred to all designated cluster nodes and installed locally. You can use the installed Plug-in for MySQL via the Virtual Client to back up and restore shared data (you can *only* perform backups and restores of data established as shared within the cluster).

# Licensing the plug-in

Another difference between using Plug-in for MySQL in a clustered environment is how it is licensed for use. The plug-in supports backup and restore of shared data *only*. Hence, for a MySQL Server Failover Cluster environment, only a single license would be needed—a clustered application license for the Virtual Client.

For information on the licensing process, including how to obtain the proper license keys, refer to the *Dell NetVault Backup Installation Guide*.

# Configuring the plug-in

This section assumes that a Virtual Client has already been created for use with Plug-in for MySQL Clustered Service. You must configure the actual Virtual Client from the NetVault Backup Server by performing the following steps on the primary node.

## To configure the plug-in

- 1 In the Navigation pane of the NetVault Backup WebUI on the NetVault Backup Server, click **Create Backup Job**, and click **Create New** next to the **Selections** list.
- 2 In the selection tree, open the newly created Virtual Client.
- 3 Open **Plug-in for MySQL**, and select the **All Instances** node.
- 4 From the **Actions** list, select **Configure**.
- 5 On the **Configure** dialog, set the applicable configuration options.

The options available are the same as those covered in [Configuring the plug-in](#).

**IMPORTANT:** You must add each cluster instance in the **Instances** field of the **Configure** dialog. To add an instance, specify the MySQL Clustered Service name as VIRTUAL SERVER NAME\INSTANCE NAME.

- 6 If you anticipate having to create additional backup jobs or modify existing backup jobs on the secondary node, perform the following steps:
  - a Fail over the primary node to the secondary node.
  - b Repeat [Step 1](#) through [Step 5](#).
  - c Fail back to the primary node.
- 7 Click **OK** to save the settings.

# Backing up data

Backing up data using Plug-in for MySQL that has been set up for use in a Virtual Client is relatively simple. Open the Plug-in for MySQL node on the **NetVault Backup Selections** page, and select the MySQL Server Virtual Server (or the items contained within) for inclusion in the backup.

Note that the instance name displayed in this page is actually the MySQL Clustered Service that is established as the Virtual Client during the installation process (as outlined in [Creating a Virtual Client](#)). If other MySQL Server Clustered Services are running on this node, those instances will also be displayed within the Plug-in for MySQL node. Data from these other instances *must not* be selected for inclusion in the backup.

**NOTE:** When you back up or restore data, make sure that you execute the process using the Virtual Client and *not* from one of the nodes. If you open or expand one of the nodes and drill down through the hierarchy, you will see the MySQL Clustered Service and, depending on which node is active, you might be able to drill down and select items. While the system might use this instance in maintaining log information, you should not attempt to execute any processes at this level.



## Restoring data

Restoring data to a Virtual Client is conducted in the same manner as a restore performed to a traditional NetVault Backup Client. All options available for a restore with Plug-in for MySQL are also available for Failover Clustering environments, and data selection is performed in the same way as well. The only difference is that restorable backups of a Virtual Client are displayed on the **Create Restore Job - Choose Saveset** page under the name of the Virtual Client, not the specific NetVault Backup Client or node that was active during each backup. When a restore job is initiated, NetVault Backup will communicate with all member Clients to determine which machine is currently in control of the failover cluster, and then target this machine for the restore.

Additionally, you can restore a NetVault Backup Virtual Client to a non-clustered (standalone) NetVault Backup Client.

All the instructions offered in reference to performing a restore can be used in the recovery of a Virtual Client. For more information on restoring a NetVault Backup Virtual Client, see the various sections in [Restoring Data](#). To restore a NetVault Backup Virtual Client to a standalone NetVault Backup Client, use the instructions provided in [Recovering to an Alternate MySQL Server](#).

# Troubleshooting

This section describes some common errors and their solutions. In those cases where an error occurs and is not described in this table, obtain the MySQL error number from the NetVault Backup logs, and then refer to the relevant MySQL documentation for the resolution.

**Table 2. Troubleshooting**

Error message	Explanation
NetVault Backup 10.x service (netvault-pgsql) will not start on Windows	<p>Check the Windows Event Viewer to see if it displays the following message: PDT FATAL: lock file “postmaster.pid” already exists</p> <p>NetVault Backup 10.x uses a PostgreSQL database. If the PostgreSQL database does not start, NetVault Backup cannot start. To correct this issue, delete the “postmaster.pid” file from the location referenced in the log and restart the NetVault Backup Server. For more information, refer to <a href="https://support.software.dell.com/netvault-backup/kb/122475">https://support.software.dell.com/netvault-backup/kb/122475</a>.</p>
<ul style="list-style-type: none"> <li>Failed to add backup record</li> <li>Failed to write index of backup to the database</li> </ul> <p>These messages indicate that the selected data was actually backed up, but the job’s index information was not properly added by NetVault Backup to its database. Without this index information, the data cannot be properly restored.</p>	<p><b>Method 1:</b></p> <p>Access the <b>Manage Devices</b> page of the NetVault Backup WebUI, and perform a scan of the media targeted by the job. NetVault Backup stores index information for backup jobs in two locations: in the NetVault Database and on the media targeted by the backup. Performing this scan adds the index information to the NetVault Database. To verify this, open the <b>Create Restore Job - Choose Saveset</b> page and locate the specific job. If you can browse it and set up a restore job, the scan process has corrected the problem.</p> <p><b>Method 2:</b></p> <p>If the previous method failed, you must run the backup job again.</p>
Backup fails with a replication error.	<p>If a backup fails with a message similar to “Failed to start Replication slave server,” it might indicate that you selected the <b>Enable MySQL Replication</b> check box but did not configure replication. To correct this issue, either clear the <b>Enable MySQL Replication</b> check box on the <b>Configure</b> dialog or set up replication, and then execute the backup job again. For more information on updating the configuration, see <a href="#">Configuring the plug-in</a>; for more information on replication, see <a href="#">Working with native MySQL replication</a>.</p>



Dell listens to customers and delivers worldwide innovative technology, business solutions and services they trust and value. For more information, visit [www.software.dell.com](http://www.software.dell.com).

## Contacting Dell

**Technical support:**

[Online support](#)

**Product questions and sales:**

(800) 306-9329

**Email:**

[info@software.dell.com](mailto:info@software.dell.com)

## Technical support resources

Technical support is available to customers who have purchased Dell software with a valid maintenance contract and to customers who have trial versions. To access the Support Portal, go to <https://support.software.dell.com/>.

The Support Portal provides self-help tools you can use to solve problems quickly and independently, 24 hours a day, 365 days a year. In addition, the portal provides direct access to product support engineers through an online Service Request system.

The site enables you to:

- Create, update, and manage Service Requests (cases)
- View Knowledge Base articles
- Obtain product notifications
- Download software. For trial software, go to [Trial Downloads](#).
- View how-to videos
- Engage in community discussions
- Chat with a support engineer