Quest® NetVault® Plug-in *for MySQL* 14.0

User's Guide

© 2026 Quest Software 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 Quest Software Inc.

The information in this document is provided in connection with Quest Software 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 Quest Software products. EXCEPT AS SET FORTH IN THE TERMS AND CONDITIONS AS SPECIFIED IN THE LICENSE AGREEMENT FOR THIS PRODUCT, QUEST SOFTWARE 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 QUEST SOFTWARE 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 QUEST SOFTWARE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Quest Software 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. Quest Software 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:

Quest Software Inc.
Attn: LEGAL Dept.
4 Polaris Way
Aliso Viejo, CA 92656

Refer to our website (https://www.quest.com) for regional and international office information.

Patents

Quest Software is proud of our advanced technology. Patents and pending patents may apply to this product. For the most current information about applicable patents for this product, please visit our website at https://www.quest.com/legal.

Trademarks

Quest, the Quest logo, and NetVault are trademarks and registered trademarks of Quest Software Inc. For a complete list of Quest marks, visit https://www.quest.com/legal/trademark-information.aspx. All other trademarks and registered trademarks are property of their respective owners.

Legend

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

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

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

NetVault Plug-in *for MySQL* User's Guide
Updated - January 2026
Software Version - 14.0
MYG-101-13.2-EN-01

Contents

Introducing NetVault Plug-in *for MySQL* **5**
NetVault Plug-in *for MySQL*: at a Glance 5
Key benefits 5
Feature summary 6
Target audience 7
Recommended additional reading 7

Installing and removing the plug-in **8**
Installation prerequisites 8
Enabling the Binary Log on the MySQL Server (Standard/Community option only) 9
Reviewing the recommended configuration 10
Installing or upgrading the plug-in 10
Removing the plug-in 11
Removing a specific MySQL Instance 11

Configuring the plug-in **12**
Configuring default settings 12
To save the settings, click OK. 16
Updating the configuration of an existing instance 16
Setting default actions for error conditions (optional) 16

Backing up data **18**
Backing up data: an overview 18
Defining a backup strategy 21
Performing a backup 25
Selecting data for a backup 25
Setting backup options 26
Finalizing and submitting the job 30

Restoring data **31**
Restoring data: an overview 31
Reviewing the available restore methods for MySQL Standard/Community 31
Reviewing the available restore options for MySQL Enterprise Backup 32
Reviewing the available restore options for Mariadb-backup 32
Restoring data 33
Selecting data for a restore 33
Setting restore options 34
Finalizing and submitting the job 39
Examples of restore scenarios for MySQL Standard/Community 40
Examples of restore scenarios for MySQL Enterprise Backup 61
Examples of restore scenarios for Mariadb-backup 63
Using advanced restore procedures for MySQL Standard/Community 65
Renaming a database during a restore 65
Restoring to a different MySQL Instance on the same server 65

Recovering to an alternate MySQL Server 68

Working with native MySQL replication 71
Using the plug-in in a native environment: an overview 71
Enabling replication support 71
Backing up replication servers 72
Replication configuration backups 72
Restoring replication servers 72

Using the plug-in in a Failover Cluster environment 73
MySQL Server Failover Clustering: an overview 73
Installing or upgrading the plug-in 74
Installation prerequisites 74
Installing the software 74
Configuring the plug-in 74
Backing up data 75
Restoring data 75

Troubleshooting 76
Technical support resources 77

1

Introducing NetVault Plug-in *for MySQL*

- NetVault Plug-in for MySQL: at a Glance
- Key benefits
- Feature summary
- Target audience
- Recommended additional reading

NetVault Plug-in *for MySQL*: at a Glance

Quest® NetVault® 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. Also, using the **MySQL Standard/Community** option, the plug-in 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 many recovery scenarios. You do not have to worry about understanding MySQL internals before implementing a backup policy that prevents the loss of committed transactions during backup and understanding when 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 and view level
  - Consolidation of multiple storage engines into a single job

  By relying on the plug-in to implement your backup policies, you can focus on more critical tasks without risking your ability to recover what is needed if a failure occurs. In addition, the IT manager's confidence is increased by knowing that MySQL data is protected.

- **Speeds up restores to reduce downtime**: You select what to restore, 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 and views
  - Rename of databases during restores
  - Restores to alternate MySQL Instances
- **Ensure business continuity**: With offsite backups being an important part of the data-protection for business-critical applications, the plug-in takes advantage of NetVault Backup's integration with a range of backup devices. NetVault lets you select which backup device to store the backup on. You can store the backup online in a virtual tape library (VTL). You can also duplicate the job to physical tape libraries shared by multiple MySQL Instances, other proprietary databases, or even general backup files.
- **Supports advanced MySQL replication techniques** –As detailed in the *MySQL Reference Guide*, MySQL supports 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 track 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 MySQL environments are protected and stored offsite for disaster-recovery purposes. At the same time, it frees administrators from having to be available 24x7. Less-experienced personnel can initiate restores, thus reducing downtime and improving business continuity.

Feature summary

- Support for MySQL Cluster Network Database (NDB) 7.x, which is based on a cluster-enabled MySQL Server 5.6. This feature uses the mysqldump utility as the backup method.
- 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 before 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
    - Transportable Tablespace (TTS) Backups
    - Common user interface across Storage Engines
    - Restoration of individual tables or databases, or entire instances
    - Renaming of a single table during restore of a TTS Backup
- Point-and-click WebUI

Target audience

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

Recommended additional reading

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

- **MariaDB documentation:** https://mariadb.com/kb/en/library/documentation/
- ***MySQL <X> Reference Guide*** (where <*X*> refers to the version of MySQL installed on the MySQL Server): https://dev.mysql.com/doc/refman/8.0/en/
- **NetVault documentation:**
    - *Quest NetVault Installation Guide*: This guide provides details on installing the NetVault Server and Client software.
    - *Quest NetVault Administrator's Guide*: This guide explains how to use NetVault and describes the functionality common to all plug-ins.
    - *Quest NetVault CLI Reference Guide*: This guide provides a description of the command-line utilities.

  You can download these guides from https://support.quest.com/technical-documents.

2

Installing and removing the plug-in

- Installation prerequisites
- Reviewing the recommended configuration
- Installing or upgrading the plug-in
- Removing the plug-in
- Removing a specific MySQL Instance

Installation prerequisites

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

CAUTION: This plug-in release supports only version 8.0 of MySQL Enterprise Backup and MySQL Enterprise Server. If you upgrade to this plug-in release, it will not be compatible with previous versions of MySQL Enterprise Backup and MySQL Enterprise Server.

- **NetVault Server and Client software**: At a minimum, the NetVault 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 setting 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 (MEB-based method) in a standalone (noncluster) environment, your environment must meet the following requirements:
  - For Windows, Linux, and UNIX environments, your MySQL Server must use version 8.0.
    - Version 8.0 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, see the applicable documentation for your MySQL Enterprise Edition product.
    - You can use version 8.0 of the MEB option only with MySQL 8.0 on RHEL 7.x and 8.x.

    IMPORTANT: If you use RHEL 6.x, verify that the libraries are up-to-date before continuing.

- **MySQL Enterprise Server:** If you want to use the **MySQL Enterprise Server** option in a standalone (noncluster) environment, your environment must meet the following requirements:
  - For Windows, Linux, and UNIX environments, your MySQL Server must use version 8.0.
  - Version 8.0 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, see the applicable documentation for your MySQL Enterprise Edition product.
  - You can use version 8.0 of the MEB option only with MySQL 8.0 on Windows Server 2019, RHEL 7.x, and 8.x.

    IMPORTANT: If you use RHEL 6.x, verify that the libraries are up-to-date before continuing.

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

Before you configure 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, see 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 wanted, 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 name of the file itself; do not include full path information or the filename extension. For more information about enabling the Binary Log, see 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. Start the **MySQL Administrator** application; for more information, see the relevant MySQL documentation.

   IMPORTANT: If you do not have the MySQL Administrator installed, update the configuration file on a Linux or UNIX system and then stop and restart the MySQL service to enable the Binary Log.
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 name of the file itself; do *not* include full path information or the filename extension. For more information about enabling the Binary Log, see the MySQL Reference Guide before continuing with installation of the plug-in.

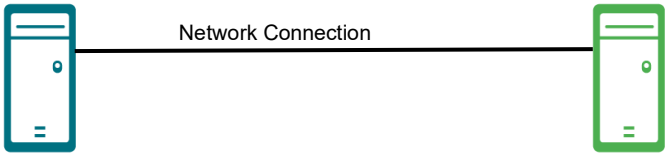
5 Exit the **MySQL Administrator** application.

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 Server and the MySQL Server, that is, all software installation and configuration requirements are performed on a single machine, Quest recommends that these two entities exist on ***separate*** machines.

Table 1. Recommended configuration



MySQL Server machine	NetVault Backup Server machine
Software installed and configuration	**Software installed and configuration**
• **MySQL Database software** (5.5 or later) • **NetVault Server and Client software** • **Plug-in *for MySQL*** • **Binary Log enabled** (MySQL Standard/Community option only) • **mysqldump/mysqlbinlog utilities**—compatible with the version of MySQL installed • **mysqlbackup utility**—MySQL Enterprise Backup option only	• **NetVault Backup Server software** • **MySQL Server added as a NetVault Client**—for details on adding a Client machine to the NetVault Backup Server, see the *Quest NetVault Backup Administrator's Guide*.

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 or upgrading the plug-in

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

NOTE: If the selected clients are all the same type, you can use the configuration wizard to install the plug-in on multiple clients at the same time. When you select multiple clients, verify 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 Plugin** button ( ).

2 Click **Choose Plug-in File**, navigate to the location of the **".npk"** installation file for the plug-in, for example, on the installation CD or the directory to which the file was downloaded from the website.

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-x.npk,"** where **xxxxx** represents the version number and platform, and click **Open**.

4 To begin installation, click **Install Plugin**.

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

Removing 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 box, 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.

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*.**

4 Click the applicable instance, and select **Remove Server** from the context menu.

No confirmation dialog box is displayed after this command is used.

3

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 configured for use. The configuration options available 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, **MariaDB Server** option, or **MariaDB Server MariaDB-Backup** option.

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. Click **Plug-in *for MySQL***, and select **Add New Server** from the context menu.
4. On the **Configure** dialog box, 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 is used in the NetVault WebUI. Quest strongly recommends that you use a generic name for the MySQL Instance instead of using a name associated with the machine on which the instance resides. This generic name improves portability and policy management across all affected clients.

     For example, if the local host name is **test_mysql_01_machine**, and the suggested name for the MySQL Instance is the same, change the name for the instance to something such as **local_mysql_server**. You can then use the instance name, **local_mysql_server**, when you configure the instance for each client. Therefore, the instance name on all clients is **local_mysql_server**.
   - **MySQL Edition** (required): Select the applicable option: **MySQL Standard/Community**, **MySQL Enterprise Backup**, **MariaDB Server, or MariaDB Server MariaDB-Backup option**. The version that you are using determines which options are modifiable on this dialog box.
   - **Username** and **Password** (required): Enter the user name and password. Use an account 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.

     The user identified by Username and Password, is required to have at least the following MySQL or MariaDB privileges:

     For ***Mysqldump*** and ***Mariadb-dump*** utilities:

     - **SELECT, SHOW VIEW, TRIGGER, LOCK TABLES, PROCESS, RELOAD, FLUSH_TABLES, CREATE,** and **ALTER**

     For ***Mysqlbackup*** utility:

     - **SELECT, BACKUP_ADMIN, RELOAD, SUPER, REPLICATION CLIENT, PROCESS, LOCK TABLES, DROP,** and **FILE**
     - **CREATE, INSERT, DROP,** and **UPDATE** on the tables mysql.backup_progress and mysql.backup_history

- **SELECT** and **ALTER** on mysql.backup_history

For ***Mariadb-backup*** utility:

- **SELECT, RELOAD, PROCESS, LOCK TABLES, and BINLOG MONITOR**

CREATE, INSERT ON **mysql.***

> **IMPORTANT:** Additional privileges could be required depending on backup contents, and backup method. Refer to MySQL Server Documentation and MariaDB Server Documentation for a complete list of requirements and privileges.

> **IMPORTANT:** NetVault 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 cannot access the instance and job requests 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-or 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- or UNIX-based Server**:

    `"<MySQLbaseDirectory>/bin"`
  - **Windows-based Server**:

    `"<MySQLbaseDirectory>\bin"`

> **NOTE:** The default path for the **mysqlbackup** utility varies depending on your environment and on whether you have customized the location. For example, with the Ubuntu 14 deb package, the utility is installed in /usr/bin/mysqlbackup by default.

- **Mysqldump Path or Mariadb-dump Path**: Enter the complete path and filename for the **mysqldump** or **mariadb-dump** 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- or 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 is displayed 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- and UNIX-based servers only): Enter the path and filename of the MySQL socket file. If a default installation of MySQL was performed, a default value is displayed 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, 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 does 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): To use the standard method of locking, flushing, and copying the table files, select this option.

  > **TIP:** The plug-in uses the MySQL **“--single-transaction”** option for InnoDB tables; however, MyISAM does not support this option. This option is useful when used with transactional tables, such as InnoDB; it is not useful with other types of tables, such as MyISAM. When the plug-in completes a Full Backup, it requires a consistent state for all table types in the targeted MySQL Instance. Also, the backup job runs **mysqldump** for *each* table that is backed up, which means that all tables are not backed up at the same time. This backup method ensures that you can select specific database objects to restore, if necessary, instead of restoring the full MySQL Instance.
  >
  > If you prefer to use an online approach, in which tables are not locked to prevent updates, consider using a MySQL replication environment. In that environment, database users interact with the master MySQL Server, on which no tables are ever locked. The backups are taken from the slave MySQL Server. During the backup process, replication updates from the master server are paused until the backup is finished. For more information about using the plug-in with a replication setup, see Working with native MySQL replication.

  - **Mysqldump**: If the tables are subject to heavy use or loading, select this option to use the **mysqldump** utility instead of copying the tables. This setting might affect 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 fail.

- **Enable Point-In-Time Recovery**: If you want to enable PIT backups and restores, select this check box. This setting lets you perform a recovery up to a point before, or after, data corruption occurs. To use this feature, 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- or UNIX-based server**:

    `<MySQLbaseDirectory>/data/<instanceName>-bin.index`
  - **Windows-based 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- or UNIX-based server**:

    `<MySQLbaseDirectory>/data/<instanceName>-relay-bin.index`
  - **Windows-based server**:

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

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

For Linux environments, Quest recommends that you use the NetVault script, .sh files, instead of running the **mysqlbackup** utility.

For example, if you are using Linux, the default paths to the .sh files include:

- For MySQL Enterprise Backup 8.0: /usr/netvault/plugins/mysql/mysqlbackup-8.0.sh

  NOTE: If you choose to use the utility instead of the .sh files, be aware that the default path for the **mysqlbackup** utility varies depending on your environment and on whether you have customized the location. For example, with the Ubuntu 14 deb package, the utility is installed in /usr/bin/mysqlbackup by default.

For Windows environments, you can use the **mysqlbackup** utility.

7 If you are using MariaDB Server MariaDB-Backup, complete the required Mysqlbackup or Mariadb-backup Path field by entering the complete path to the directory where the Mariadb-backup utility resides.

8 If you are running **MySQL Enterprise Backup (MEB)**, and **Transparent Data Encryption (TDE)** has been enabled in the MySQL instance, select the option **Server uses Transparent Data Encryption**. A user provided key can be entered in the **Transparent Data Encryption Password** field. If entered, the user provided password would be send to the MySQL Server to be used to encrypt the keyring data file that is included in the meta folder of the backup. If user provided password is entered, the same password would have to be used in the backup and in the restore. If the MySQL Server is using the keyring_encrypted_file plugin or the component_keyring_encrypted_file component the user provided password is used to provide the keyring file encryption password that has been set on the MySQL Server.

NOTE: Transparent Data Encryption (TDE) only applies to MySQL Enterprise Backup (MEB). For more details, refer to the MySQL Documentation.

To save the settings, click OK.

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.
4. Click the applicable instance, and select **Configure** from the context menu.

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

   IMPORTANT: In the **Edit** version, the **MySQL Instance Name** field is disabled. 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 tables. Occasionally during the execution of a backup job, an unsupported storage engine is encountered or a database or table is inaccessible. If this situation is encountered, it prevents one or more items from being backed up successfully, but the remaining items selected in the backup job *are* backed up. A MySQL DBA must 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.

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 issue 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 issue 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 issue 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 issue occurs when a table is encountered during the backup that has a type of storage engine that the plug-in does not support.

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."** The errors are logged in the NetVault binary logs and ignored on the **Job Status** page. A backup saveset is created that includes the items that were 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 To save the settings, click **Apply**.

4

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 topics:

- Important notes for MySQL Standard/Community
- Important notes for MySQL Enterprise Backup
- Defining a backup strategy

IMPORTANT: Quest strongly recommends that you omit special characters from database names. If a database name contains any of the following characters, the plug-in cannot restore it: $ ^ = @ # % +

Because backup data is streamed from the MySQL Server directly to the NetVault Backup Media Manager, use of the preceding characters might be interpreted as a sequence command, which affects the integrity of the backup data.

Important notes for MySQL Standard/Community

If you intend to use the **MySQL Standard/Community** option, review the following guidelines and information:

- All characters except alphanumeric characters and underscores are considered special characters
- Using the `MIXED` Binary Logging Format

All characters except alphanumeric characters and underscores are considered special characters

If your environment uses databases whose names contain special characters, such as hyphens, be aware of the following limitations:

- All table types except MyISAM are backed up, even if the database name contains hyphens. These backups occur 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. 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. Backups are not generated because the plug-in bypasses MySQL commands and tries to copy the table files directly. The plug-in logs an error message indicating that the table file cannot be located, and then fails the backup job without creating a saveset.

  With previous versions, the plug-in attempted to verify the existence of the database directory, logged a warning message when it failed to do so, and continued with backup of the next database. The backup completed with warnings and created 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. To do so, manually set the ValidateDatabaseDirectory parameter to TRUE in the plug-in configuration file, **"nvmysql.cfg,"** as follows:

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

If you then decide to use the new behavior instead, you can change the parameter to FALSE or remove the parameter from the "**nvmysql.cfg**" file.

- Restoring an Incremental or Differential Backup that includes SQL statements to create databases or database objects—for example, table, view, and so on—fails if any of the databases or database objects exist. To avoid this issue, run a Full Backup after you create or drop one or more databases or database objects. This step ensures that subsequent Incremental or Differential Backups do not include any CREATE or DROP SQL statements.

Using the `MIXED` Binary Logging Format

MySQL does not enforce the use of the `USE` statement when the `MIXED` Binary Logging Format is used. Therefore, Quest recommends that all database users and programs ensure that tables that are modified are in the database selected by `USE`, and that no cross-database updates are issued. If this guideline is not suitable for your environment, Quest recommends that you do not use the `MIXED` Binary Logging Format.

IMPORTANT: Incremental and Differential Backup jobs complete with a warning if the `MIXED` Binary Logging Format is `used`.

If your environment uses the `MIXED` Binary Logging Format, it might prevent Binary Log entries from being replayed during a PIT Recovery. During recovery, the plug-in uses **mysqlbinlog** with the **"--database"** option to replay only the entries related to the databases that you selected for the restore job. If **"--database"** is not used, all entries are replayed, which affects all databases. When the `MIXED` Binary Logging Format is used, entries are written in a way that might prevent **mysqlbinlog** with the **"--database"** option from replaying some, or all, entries. For more information, see https://dev.mysql.com/doc/refman/5.7/en/mysqlbinlog.html#option_mysqlbinlog_database.

To ensure that the `MIXED` Binary Logging Format works correctly with the **"--database"** option, all transactions for specific updates to a database have to be issued under a `USE` statement that selects the database.

This same situation occurs if the Incremental or Differential Backups are not restored and **mysqlbinlog** applies the current Binary Log from the MySQL Server. This situation occurs because of how the Binary Log is written, not because of how the Binary Log is stored in the backup.

IMPORTANT: Ensuring that tables that you modify belong to the database specified in the `USE` statement applies to transactions that are generated through the MySQL command prompt. It also applies to transactions that are generated by scripts, programs, and other applications that interact with the MySQL Server databases.

The following examples demonstrate different ways in which `MIXED` affects recovery behavior.

- **Example 1:** In this example, a row of data is inserted into **my_table** of **my_database**. There is no `USE` statement, so the database in use is the default database, for example, **mysql** database. If `binlog_format` is set to `MIXED`, the following transaction is not replayed when **mysqlbinlog** applies the **"--database my_database"** option to the Binary Log.

  ```
  -bash-$ mysql
  mysql> insert into my_database.my_table (C1,C2) values(1,now());
  Query OK, 1 row affected (0.01 sec)
  ```

- **Example 2:** In this example, a row of data is inserted into **my_table** of **my_database**. There is a `USE` statement, but a different database is specified; that is, **my_database** is not selected in the `USE` statement. If `binlog_format` is set to `MIXED`, the following transaction is not replayed when **mysqlbinlog** applies the **"--database my_database"** option to the Binary Log.

  ```
  -bash-$ mysql
  mysql> use mysql
  Database changed
  ```

```
mysql> insert into my_database.my_table (C1,C2) values(2,now());
Query OK, 1 row affected (0.04 sec)
```

- **Example 3:** In this example, a row of data is inserted into **my_table** of **my_database**, and **my_database** is selected in a `USE` statement. If `binlog_format` is set to `MIXED`, the following transaction is replayed when **mysqlbinlog** applies the **"--database my_database"** option to the Binary Log.

```
-bash-$ mysql
mysql> use my_database
Database changed
mysql> insert into my_database.my_table (C1,C2) values(3,now());
Query OK, 1 row affected (0.04 sec)
```

- **Example 4:** In this example, there are two insert queries. The first insert is done for **my_database**, which is different than the database selected in the `USE` statement. The second insert is done under the scope of a `USE` statement that selects **my_database**. If binlog_format is set to `MIXED`, the first insert is not replayed because **my_database** is not specified in the `USE` statement, but the second insert is replayed because **my_database** is specified in the `USE` statement.

```
-bash-$ mysql
mysql> use mysql
Database changed
mysql> insert into my_database.my_table (C1,C2) values(4,now());
Query OK, 1 row affected (0.01 sec)
mysql> use my_database
Database changed
mysql> insert into my_database.my_table (C1,C2) values(5,now());
Query OK, 1 row affected (0.04 sec)
```

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. Also, any database changes made during the backup process are included. This behavior 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 description reflects the current behavior of MySQL Enterprise Backup, **mysqlbackup** utility; this behavior might change in a future release, post-3.12, of MySQL.

- In MySQL 5.6 and later, the **innodb_file_per_table** configuration option 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 must place an InnoDB table outside the tablespace, create it while the **innodb_file_per_table** option is enabled in MySQL. Each .ibd file contains the data and indexes of only one table.

Defining a backup strategy

When defining a MySQL backup strategy, 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. 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** utility or applicable NetVault script is run 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 supporting hot backup of InnoDB tables, the MEB-based option supports improved backup performance.

  If you use the mysqldump-based option, the command is run 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 helps 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**
- **Entire Databases Copy 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 ensures that Binary Logs are purged after a Full or Incremental Backup. This option is enabled by default when the plug-in is used with a standard MySQL Server configuration, **Enable MySQL Replication** is disabled, and **Enable Point-In-Time Recovery** is enabled. It is disabled when the plug-in is connected to a cluster; you must manage the purging of the Binary Logs outside of the plug-in.

IMPORTANT: In a mixed environment, where a NetVault Backup Server manages both clustered and standard MySQL Servers, do *not* reuse a Backup Options Set—created for a standard MySQL Server—for a MySQL-based cluster.

If you do not select the **Purge Binary Logs…** option, the plug-in tracks the **Last Log Backed Up** in its configuration file; 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 instanced-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. You can use Incremental Backups to restore to a time before 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 that you start the MySQL Instance with the **"-log-bin"** option, which enables the Binary Log. This procedure is outlined in Enabling the Binary Log on the MySQL Server (Standard/Community option only). For more information, see 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 tracks 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. The size and duration increase because each backup of this type includes the Binary Logs that were also included in the previous Differential Backup *and* 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 includes the Binary Logs generated since the Full Backup on Sunday, while Tuesday's Differential includes the Binary Logs generated on Monday and those logs generated on Tuesday. Wednesday's Differential includes 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 before and after the failure or corruption. Unlike a Full Backup, a Differential does ***not*** require read-only access during the backup.

Differential Backups require that you start the MySQL Instance with the **"-log-bin"** option, which enables the Binary Log. This procedure is outlined in Enabling the Binary Log on the MySQL Server (Standard/Community option only). For more information, see 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 process can lead to longer restores due to the increased DBA intervention required 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 process 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.

Entire Databases Copy Backups

As described for Individual Database/Table Copy Only Backups, the Entire Databases Copy Backup option is used only for special purposes as it creates a copy of the selected MySQL databases, including all the corresponding InnoDB tables of the selected databases. **"Copy"** 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.

IMPORTANT: You can use this option only if all the tables for the selected databases are InnoDB tables.

Individual Database/Table Copy Only Backups vs. Entire Databases Copy Backups

For each selected database, even if only one table of the database is selected, the Entire Databases Copy Backup option backs up the entire database. This option lets you select individual databases for backup, but it does not let you select individual tables. In addition, this option only supports the backing up of InnoDB tables.

The Individual Database/Table Copy Only Backup option lets you select individual databases and individual tables, and you can include InnoDB and MyISAM tables in the backup. However, backups are usually completed faster for the Entire Databases Copy Backup option than for the Individual Database/Table Copy Only Backup option.

Reviewing the backup types for MySQL Enterprise Backup

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

Full Backups

In a Full Backup for the **MySQL Enterprise Backup** option, the plug-in uses the **mysqlbackup** utility or applicable NetVault Backup script 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.

Transportable Tablespace (TTS) Backups

If you perform a TTS Backup, the plug-in issues a Full Backup and adds the **"--use-tts"** MySQL option.

> **IMPORTANT:** Quest strongly recommends that you only generate TTS Backups as standalone backups, separate from your backup plan. Because TTS Backups are partial backups, you cannot use them to replace or complement a Full or Incremental Backup strategy, nor can you use them for a disaster recovery operation.

If you intend to generate TTS Backups, be aware of the following limitations:

- TTS Backups are only supported in MySQL Server 5.6 and later.
- Only InnoDB tables are included in the backup.
- Only tables that were created with the **innodb_file_per_table** option enabled are included in the backup.
- Backup of a partitioned table fails if the partition was created in a shared tablespace.
- The backup excludes the binary or relay log.

For more limitations on the use of the **"--use-tts"** option, see https://dev.mysql.com/doc/mysql-enterprise-backup/4.0/en/backup-partial-options.html.

Reviewing the backup types for Mariadb-backup

For the **Mariadb-backup** method, the plug-in runs the **Mariadb-backup** 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 **Mariadb-backup** option, the plug-in uses the **Mariadb-backup** utility or applicable NetVault script 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 MariaDB instance.

Incremental Backups

For Mariadb-backup Incremental backups, only data that changed since the last Full Backup is backed up.

Examples of backup sequences for MySQL Standard/Community

The following provides examples of the various sequences.

- **Full Backups only:** When 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 daily 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 can perform a PIT Recovery using the Binary Logs that currently exist on the MySQL Server.
- **Full and Incremental Backups:** When 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 biweekly 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 includes 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 run multiple restore jobs.

- **Full and Differential Backups:** When 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 biweekly 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 includes the Binary Logs that were generated since the last Full Backup. As noted earlier, this process 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 increase not only in size but in duration, restores are quicker due to the fewer restore jobs that must be run.

Performing a backup

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

- 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. Create the Backup Selection Set during a Full Backup, and use it for Full, Incremental, and Differential Backups. The backup job reports an error if you do not use a Selection Set for the Incremental or Differential Backup. For more information, see the *Quest NetVault Backup Administrator's Guide*.

TIP: To use an existing set, click **Create Backup Job**, and select the set from the **Selections** list.

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 name that lets you easily identify the job when monitoring its progress or restoring data. The job name can contain alphanumeric and nonalphanumeric characters, but it cannot contain non-Latin characters. On Linux, the name 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. Next to the **Selections** list, click **Create New**.

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

5 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 detailed 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 applicable 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 detailed set of data for backup with the **MySQL Standard/Community** option, 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**, detailed selections are ignored, and the entire database is backed up. For MySQL 5.5 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 5.5 and later, the "information_schema" database is displayed in the selection tree but it is not available for selection. This issue occurs 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.

6 Click **Save**, enter a name in the **Create New Set** dialog box, and click **Save**.

The name can contain alphanumeric and nonalphanumeric characters, but it cannot contain non-Latin characters. On Linux, the name 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.

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 if there is a failure.

> **TIP:** To use an existing set, in the **Plugin Options** list, select the set that you want to use.

1 Next to the **Plugin Options** list, click **Create New**.

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 box for this instance, the Full, Incremental, and Differential forms of backup are *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 and all tables included in the current MySQL Instance, select this option. Selecting this option backs up everything, regardless of the data that you select for the Backup Selection Set.
- **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

contains all Binary Logs generated since the original Full was performed. After the backup is finished, the Binary Logs are ***kept*** on the MySQL Instance.

IMPORTANT: Incremental and Differential Backup jobs complete with a warning if the `MIXED` Binary Logging Format is used. For more information, see Using the `MIXED` Binary Logging Format.

CAUTION: Because Binary Log entries might be not replayed during a PIT recovery if the `MIXED` Binary Logging Format is used, databases selected for recovery might not be rolled back to the selected point. For more information, see Important notes for MySQL Standard/Community and Using the `MIXED` Binary Logging Format**.**

- **Individual Database/Table Copy Only**: To create a special-purpose copy of a MySQL environment that does 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 do not affect the sequence established by a Full and Incremental or Differential MySQL backup scenario, that is, these backups have no effect on Binary Logs. This form of backup is independent of what should be established as a normal sequence of backups by using a Full and Incremental or Differential MySQL backup scenario. In addition, a copy backup ***cannot*** be used as a replacement for a Full Backup.
- **Entire Databases Copy Backup**: To back up all the contents of the selected databases by creating a special-purpose copy of a MySQL environment, select this option. For example, to build a test environment, select this option. This option is similar to the Individual Database/Table Copy Only option, but this option backs up all selected databases and their corresponding tables. You can use this option only if all the tables for the selected databases are InnoDB tables. When used, these copy backups do not affect the sequence established by a Full and Incremental or Differential MySQL backup scenario, that is, these backups have no effect on Binary Logs. In addition, a copy backup ***cannot*** be used as a replacement for a Full Backup.

IMPORTANT: You can use this option only if all the tables for the selected databases are InnoDB tables.

Selecting this option backs up the entire selected database and all corresponding tables. This occurs even if you select a specific table to back up. If you want to back up a specific table, use the Individual Database/Table Copy Only option.

- **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 are unable to insert, update, or delete data across the entire instance while the Full Backup is in progress. When this option is cleared, the plug-in locks each table during the backup process *only* when the table is being backed up; therefore, if the instance contains tables that are related, Quest 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 when the plug-in is used with a standard MySQL Server configuration, **Enable MySQL Replication** is disabled, and **Enable Point-In-Time Recovery** is enabled. It is disabled when the plug-in is connected to a cluster; you must manage the purging of the Binary Logs outside of the plug-in. Quest recommends that you use this option, but you can decide how much control you want over the Binary Logs.

  IMPORTANT: In a mixed environment, where a NetVault Backup Server manages both clustered and standard MySQL Servers, do *not* reuse a Backup Options Set—created for a standard MySQL Server—for a MySQL-based cluster.

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."** The errors are logged in the NetVault Backup binary logs and ignored on the **Job Status** page. A backup saveset is created that includes the items that were 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.

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 that the plug-in generates internally. Because of this order, an option that you enter here that contradicts one generated internally is overridden by the one generated by the plug-in.

Errors detected by **mysqldump** that cause the job to fail are 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 about the options supported by your version of **mysqldump**, see the applicable MySQL documentation.

CAUTION: Do *not* use the `--routines (-R)` or `--triggers` option with this feature. Using these options interferes with backups of database tables, and while backups succeed, restores might fail. If there are stored procedures and triggers that must be backed up for a database, the plug-in generates mysqldump commands internally with the `--routines` and `--triggers` options.

5 Click **Save** to save the set.

6 In the **Create New Set** dialog box, specify a name for the set, and click **Save**.

The name can contain alphanumeric and nonalphanumeric characters, but it cannot contain non-Latin characters. On Linux, the name 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.

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 if there is a failure.

TIP: To use an existing set, in the **Plugin Options** list, select the set that you want to use.

1 Next to the **Plugin Options** list, click **Create New**.

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.
- **Transportable Tablespace (TTS) Backup:** To create a partial backup that takes advantage of the MySQL TTS feature, select this option.

3 Click **Save** to save the set.

4 In the **Create New Set** dialog box, specify a name for the set, and click **Save**.

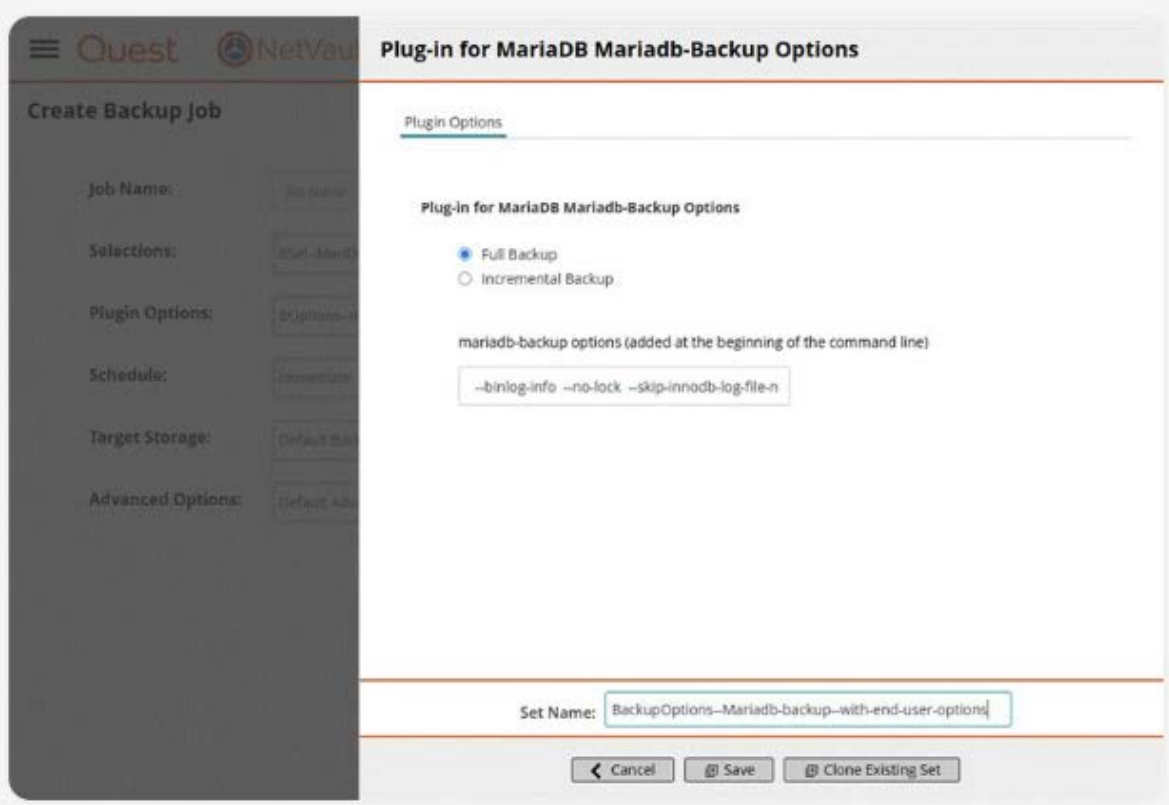
The name can contain alphanumeric and nonalphanumeric characters, but it cannot contain non-Latin characters. On Linux, the name can have a maximum of 200 characters. On Windows, there is no length restriction. However, a maximum of 40 characters is recommended on all platforms.

Setting backup options for Mariadb-backup

With the applicable items selected for backup, you can select the type of backup to perform.

TIP: To use an existing set, in the Plugin Options list, select the set that you want to use.

1. Next to the **Plugin Options** list, click **Create New**.
2. Select the applicable option:
   - **Full Backup** (default selection): To back up every database and table selected in the current MariaDB Instance, select this option.
   - **Incremental Backup**: To back up only the data changed since the last Full Backup, select this option.
3. Use the mariadb-backup options (added at the beginning of the command line) input box to specify command-line flags that you want the mariadb-backup utility to use for the job.



All the options must begin with a dash(-) or double dash(--) and must not contain any invalid charecters such as (; | <>).

The options are added to the **mariadb-backup** command first, followed by the options that the plug-in generates internally. Because of this order, the option you entered in the input box contradicts with the plug-in generated internally, hence the option is overridden by the one generated by the plug-in.

If you have existing **MariaDB** option files configured, the option which you input will be appended to the options in the Options file. To ignore the existing **MariaDB** option files, enter the flag ***--no-defaults*** in the text box.

When you input ***--no-defaults*** option, **mariadb-backup** does not read options from any of the standard MariaDB configuration files (like mariadb.cnf, my.cnf, or my.ini). This compels mariadb-backup to rely solely on the option that you specified on the command line and **mariadb-backup** own internal defaults.

If you input ***non mariadb-backup option*** or ***incorrect option***, the job fails. All the errors which mariadb-backup detects and causes the job to fail are found in the Job binary log.

For the complete list of supported options, refer to the official MariaDB documentation for your specific version of mariadb-backup utility.

4 Click **Save** to save the set.

5 In the **Create New Set** dialog box, specify a name for the set, and click **Save**.

The name can contain alphanumeric and non-alphanumeric characters, but it cannot contain non-Latin characters. On Linux, the name 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.

Finalizing and submitting the 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**.

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

5

Restoring data

- Restoring data: an overview
- Restoring data
- Use the Windows Services Console to start the MariaDB service, typically named MariaDB. Check MariaDBdocumentation for more details.

Restoring data: an overview

This topic 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. Quest recommends that you review these topics 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
- Reviewing the available restore options for MySQL Enterprise Backup
- Reviewing the available restore options for Mariadb-backup

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 the plug-in performs a Full or Individual Database/Table Copy Only backup, it uses MySQL's **mysqldump** utility 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 run automatically.

Incremental or Differential Restores

When the plug-in performs Incremental or Differential Backups, the MySQL Binary Log Index is used to determine which Binary Logs must 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 run automatically. This process is called "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 running 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, see 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.

Reviewing the available restore options for MySQL Enterprise Backup

With the MEB-based method, you can perform a Full or Incremental Restore. If you used the TTS Backup option, you also have the option of completing a TTS Restore. Within the TTS Restore process, you have the additional option of restoring specific tables, known as a partial restore, and renaming one of the specified tables.

IMPORTANT: Because of the limited capabilities of using a TTS Backup or Restore, Quest recommends that you use the TTS options with discretion.

Be aware of the following restrictions for restoring a TTS Backup:

- Make sure that the destination MySQL Server is running, because a connection with the server must be established.
- Make sure that the tables you are restoring do not exist on the destination server.
- Make sure that the destination server uses the same page size used on the original MySQL Server on which the backup was performed.
- Make sure that the innodb_file_per_table option is enabled on the destination server.
- Restore fails if the InnoDB file (.ibd file) you are restoring does not match the value of the innodb_file_format variable on the destination server.

For more information, see https://dev.mysql.com/doc/mysql-enterprise-backup/4.0/en/restore-use-tts.html.

Reviewing the available restore options for Mariadb-backup

With the Mariadb-backup based method, you can perform a Full or Incremental Restore.

Restoring data

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

- Selecting data for a restore
- Setting restore options
- Finalizing and submitting the job
- Examples of restore scenarios for MySQL Standard/Community
- Examples of restore scenarios for MySQL Enterprise Backup

Selecting data for a 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 by creation date.

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, server name, client name, plug-in name, saveset date and time, retirement setting, Incremental Backup or not, Archive or not, saveset size, and snapshot-based backup or not.

5 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 varies, 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 and table data was included in the backup.

  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. Selecting this option only restores all the data items that were selected for the backup job; that is, by selecting this node for a restore, you are not performing a restore of all the databases that currently exist in a MySQL Instance—only those databases 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 detailed restore is wanted, 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.

IMPORTANT: MySQL uses multiple file-formats to storage database information. Verify 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 depend 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, perform the following steps.

1 Use the following guidelines to select the applicable options on the **Point-in-Time Recovery** tab.

- **Perform PIT Recovery on Current Binary Logs**: 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, select this option. After this option is enabled, all remaining options on this tab are made available.
- **Point In Time Type:** In this section, select the applicable form of PIT Recovery:
  - **Time Based PIT** (default selection): To restore the selected data *to a specified time*, described in Time-based Point-in-Time (PIT) Recovery, select this option. With this option selected, the **Time Based PIT Details** section is made available.
  - **Position Based PIT**: 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, select this option. 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 ***before*** the unwanted transaction, select this option. If you select only this option, all transactions that occurred ***after*** the time specified here are 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 are 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.

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 automatically stops and starts at the specified time and ignores 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, select the **Enable Recovery Prior to …** option and enter **"11:00" - "29 Jan 2007"** 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, are 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 ***before*** the unwanted transaction, select this option. If you select only this option, all transactions that occurred ***after*** the position specified here are lost. This option offers the following associated options:
    - **Stop Position**: Enter the position in the Binary Log ***before*** 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 you want a different file or the applicable file is not listed, select **OTHER**, and enter the applicable filename 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 are 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 you want a different file or the applicable file is not listed, select **OTHER**, and enter the applicable filename 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 filename. Only transactions that occurred between the positions specified in the **Start Position** and the **Specific Position** fields are 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, select the **Enable Recovery Prior to …** option and enter **"805"** as the **Stop Position**, and then configure its associated options to call out the Binary Log. In addition, select the **Enable Recovery After …** option and enter **"810"** as the **Start Position**, and then configure its associated options to call out the Binary Log. As a result, all transactions that were logged in the specified Binary Log between 805 and 810 are 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** stab.

- **Restoring to the same MySQL Instance**: If the restore targets the same instance that was originally backed up, leave these fields blank. NetVault Backup uses the values set in the **Configure** dialog box. 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 allows access to the new instance. Also, enter the NetVault Backup name established for the new instance in the **Instance Name** field—this name is the name established as the **MySQL Instance Name** in the **Configure** dialog box; 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, perform the following steps.

1 Use the following guidelines to select the applicable options on the **Point-in-Time Recovery** tab.

- **Perform PIT Recovery**: To perform a **Point-in-Time** form of restore of the selected data items, select this option. After this option is enabled, all remaining options on this tab are made available.

  Incremental and Differential Restores use the Binary Logs to complete a restore. Therefore, when restoring this form of backup, 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 process is performed ***after*** any Binary Log transactions that were saved in the Incremental or 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 or Differential Backup to a ***temporary directory*** on the MySQL Server, that is, **"NETVAULT_HOME/tmp/MySQL/,"** select this option. This option lets you use the **mysqlbinlog** utility to review the recovered logs to identify the time and 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 process applies 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 process is 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, select the applicable form of PIT Recovery:
  - **Time Based PIT** (default selection): To restore the selected data *to a specified time,* described in Time-based Point-in-Time (PIT) Recovery select this option. With this option selected, the **Time Based PIT Details** section is made available.
  - **Position Based PIT**: 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 select this option. 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 ***before*** the unwanted transaction, select this option. If you select only this

option, all transactions that occurred ***after*** the time specified here are 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 are 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. Enter the applicable stop time in the associated fields, using the 24-hour time format.

> **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, select the **Enable Recovery Prior to …** option and enter **"11:00" - "29 Jan 2007"** as the **Stop Date/Time**. In addition, select the **Enable Recovery After …** option and enter **"11:15" - "29 Jan 2007"** as the **Start Date/Time**. As a result, all transactions that occurred between 11:00 and 11:15 on January 29, 2007, are 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 ***before*** the unwanted transaction, select this option. If you select only this option, all transactions that occurred ***after*** the position specified here are lost. This option offers the following associated options:
    - **Stop Position**: Enter the position in the Binary Log ***before*** 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 you want a different file or the applicable file is not listed, select **OTHER**, and enter the applicable filename 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 are 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 you want a different file or the applicable file is not listed, select **OTHER**, and enter the applicable filename 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 filename. Only transactions that occurred between the positions specified in the **Start Position** and the **Specific Position** fields are 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, select the **Enable Recovery Prior to …** option and enter **"805"** as the **Stop Position**, and then configure its associated options to call out the Binary Log. In addition, select the **Enable Recovery After …** option and enter **"810"** as the **Start Position**, and then configure its associated options to call out the Binary Log. As a result, all transactions that were logged in the specified Binary Log between 805 and 810 are 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. Use the fields in this section to input account information to allow restore access to the target instance of MySQL. Based on the wanted 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 uses the values set in the **Configure** dialog box. 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 allows access to the new instance. Also, enter the NetVault Backup name established for the new instance in the **Instance Name** field—this name is the name established as the **MySQL Instance Name** in the **Configure** dialog box; 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.

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, verify 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 **Client Settings**, and then click **General** in the **System and Security** section.

IMPORTANT: On Windows Operating System, by default, NetVault runs under *LOCAL_SERVICE* account. When you restore a MySQL Enterprise Backup selecting the **Copy Back** option, data will be restored back to MySQL by **LOCAL_SERVICE**. During restores, ensure that **LOCAL_SERVICE** account has read, write, and create permission to the MySQL Data folder, subfolders, and files. In a default installation of MySQL 8.0, on Windows Operating System, the MySQL Data folder is usually located under **C:\ProgramData\MySQL\MySQL Server 8.0\Data**

- **Full Restore**: Select the applicable options.
  - **Restore, Extract Raw Full Backup…** (default selection)**:** To restore a Full Backup to a temporary location that mirrors the MySQL Server data repository directory hierarchy, select this option. 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:** If you must list the contents of the backup to determine which backup you require to run the next option, select this option.
  - **Extract Raw Full Backup from Temp File…:** After you have used the results the preceding option to determine which backup you must restore, select this option. 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…** (option available for standard Full Restores)**:** When you are ready to shut down the MySQL Server and copy the restored contents from the temporary location back to the original location, select this option.
- **Copy Back Prepared Full Backup to MySQL Server Repository** (option available for TTS Full Restores)**:** If you want to copy the restored contents from the temporary location back to the original location, select this option. Two more options are also available:
  - **Include Tables:** If you want to perform a partial restore, enter a regular expression in this field to describe the naming pattern of the tables that you want to include in the restore. If you complete this field, the plug-in issues the **"--include-tables"** MySQL command.
  - **Rename Table:** If you completed the **Include Tables** field to specify which tables to restore, you can use this field to change the name of one of the specified tables. To rename a table, use the `original_name to new_name` expression. If you complete this field, the plug-in issues the **"--rename"** MySQL command.

  IMPORTANT: When you select a TTS Backup to restore using the **Shutdown MySQL Server and Copy Back…** option, the plug-in ignores items selected on the **Restore Selections** dialog box. The plug-in only restores the tables specified in the **Include Tables** field and the corresponding database.
- **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)**:** To restore an Incremental Backup, select this option. 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:** If you must list the contents of the backup to determine which backup you require to run the next option, select this option.
  - **Extract Incremental Backup from Temp File…:** After you have used the results the preceding option to determine which backup you must restore, select this option.
  - **Shutdown MySQL Server and Copy Back…:** When you are ready to shut down the MySQL Server and copy the restored contents from the temporary location back to the original location, select this option.
  - **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 the job

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

1. To save the settings, click **Ok**, 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 name that lets you easily identify the job when monitoring its progress. The job name can contain alphanumeric and nonalphanumeric characters, but it cannot contain non-Latin characters. On Linux, the name 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 filename on the target OS. For example, the characters /,\,*, and @ should not be used on Windows. This requirement 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 box.

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

5 Click **Save** or **Save & Submit**, whichever is applicable.

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

> **IMPORTANT:** If you are using MySQL Enterprise Backup in a Linux or UNIX environment, verify that the file-ownership and permissions information for the restored data matches what it was before the data was backed up. Because the **mysqlbackup** utility does not record this information during the backup process, the information might be different after the restore is completed. For more information, see https://docs.oracle.com/cd/E17952_01/mysql-enterprise-backup-3.11-en/bugs.backup.html.

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 regarding data selected for restore and options available on the **Options** tab. The following topics 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
- PIT restore when the `MIXED` Binary Logging Format is used and cross-database updates are issued

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 and 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 then learns that the table no longer exists because a developer unknowingly dropped it at ***6:00 A.M. on Monday*** before the DBA's arrival at work.

Method 1: Recovery *before* erroneous statement

The DBA decides to recover up to the time ***right before*** the Drop Table command was issued. This decision 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 before 6:00 A.M. on Monday.

3 **Submit the job.**

Method 2: Recovery *before* 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 decision ensures 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 before 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 then learns that the table no longer exists because a developer unknowingly dropped it at ***6:00 A.M. on Monday*** before the DBA's arrival at work.

Method 1: Recovery *before* 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 process, 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 step 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 about this utility and process, see 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 *before* 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 decision ensures 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 process, 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 step 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 about this utility and process, see 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 process 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 then learns that the table no longer exists because a developer unknowingly dropped the table sometime early Thursday, before 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 options 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 options 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 options 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 options are used.
3. **Submit the job.**

Full restore and time-based Point-in-Time Recovery

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

Method 1: Recovery *before* 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 then learns that the table no longer exists because a developer unknowingly dropped it at ***8:00 P.M. on Wednesday***.

The DBA must 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 options 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 options 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 options 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 before 8:00 P.M. on Wednesday.
3. **Submit the job.**

Method 2: Recovery *before* 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 then 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 decision ensures 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 options 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 options 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 options 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 before 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 *before* 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 then learns that the table no longer exists because a developer unknowingly dropped it at ***6:00 A.M. on Thursday***.

The DBA must 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 options 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 options 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 options 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 before 6:00 A.M. on Thursday.

3 **Submit the job.**

Method 4: Recovery *before* 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 then 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 decision ensures 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 options 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 options 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 options 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 before 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 and position-based Point-in-Time Recovery

In the following examples, a Full and 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 recovery is done using identified "position values" that exist in the MySQL Binary Logs.

Method 1: Recovery *before* 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 then 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 process, 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 process:

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 options 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 options 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 options 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 process 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 step 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 about this utility and process, see 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 *before* 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 then 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 process, 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 process:

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 options 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 options 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 options 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 step 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 step 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 about this utility and process, see 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 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, **"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 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 *before* 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 then learns that the table no longer exists because a developer unknowingly dropped it at ***6:00 A.M. on Thursday***.

The DBA must 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 process, 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 process:

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 options 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 options 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 options 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 step is performed outside of NetVault to identify the ***position*** of the Drop Table command that the DBA does not want to restore. (For information about this utility and process, see 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 to use the current Binary Logs to apply all database transactions that occurred ***after*** Wednesday night's Incremental Backup. This step recovers 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 *before* 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 then learns that the table no longer exists because a developer unknowingly dropped it at ***6:00 A.M. on Thursday***.

The DBA must 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 process, 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 process:

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 options 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 options 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 options 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 step 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 about this utility and process, see 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 step recovers 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."**
   - **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.

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 and 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 then learns that the table no longer exists because a developer unknowingly dropped it sometime early Thursday before 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 options 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* have to restore **Monday** and **Tuesday** night's Differential Backups. By choosing 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 and time-based Point-in-Time Recovery

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

Method 1: Recovery before 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 then learns that the table no longer exists because a developer unknowingly dropped it at ***8:00 P.M. on Wednesday***.

The DBA must 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 options 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* have to restore **Monday** and **Tuesday** night's Differential Backups. By choosing 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 before 8:00 P.M. on Wednesday.

3 **Submit the job.**

Method 2: Recovery *before* 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 then 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 decision ensures 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 options 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* have to restore **Monday** and **Tuesday** night's Differential Backups. By choosing 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 before 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 *before* 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 then learns that the table no longer exists because a developer unknowingly dropped it at ***6:00 A.M. on Thursday***.

The DBA must 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 options 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* have to restore **Monday** and **Tuesday** night's Differential Backups. By choosing 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 before 6:00 A.M. on Thursday.

3 **Submit the job.**

Method 4: Recovery *before* 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 then 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 decision ensures 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 options 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* have to restore **Monday** and **Tuesday** night's Differential Backups. By choosing 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 before 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 and position-based Point-in-Time Recovery

In the following examples, a Full and 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 process is done using identified "position values" that exist in the MySQL Binary Logs.

Method 1: Recovery *before* 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 then 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 process:

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 options 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 process 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 step 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 about this utility and process, see 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* have to restore **Monday** and **Tuesday** night's Differential Backups. By choosing 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 *before* 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 then 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 process:

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 options 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 process 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 step 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 about this utility and process, see 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.

> **IMPORTANT:** The DBA does *not* have to restore **Monday** and **Tuesday** night's Differential Backups. By choosing 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**.

  > **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 *before* 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 then learns that the table no longer exists because a developer unknowingly dropped it at ***6:00 A.M. on Thursday***.

The DBA must 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 process:

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 options 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 step 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 about this utility and process, see 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* have to restore **Monday** and **Tuesday** night's Differential Backups. By choosing 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 step recovers 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 *before* 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 then 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. Furthermore, the DBA wants a more precise recovery, so he decides to use a position-based recovery. The following phases illustrate this process:

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 options 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 step 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 about this utility and process, see 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* have to restore **Monday** and **Tuesday** night's Differential Backups. 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 step recovers 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.

PIT restore when the `MIXED` Binary Logging Format is used and cross-database updates are issued

> **IMPORTANT:** If your site uses the `MIXED` Binary Logging Format, and all database users and programs follow the best practice of ensuring that tables that are modified are in the database selected by `USE`, and no cross-database updates are issued, this topic does not apply to your site. (For more information, see Using the `MIXED` Binary Logging Format.) You can run PIT restore jobs and the Binary Log is replayed to the specified point for the databases selected in the job.

As stated previously, if users and programs in your environment modify tables in databases that are not selected by `USE` and they issue cross-database updates, transactions might not be replayed to the specified time when you run a PIT restore job. Quest recommends that all database users and programs ensure that modified tables are in the database selected by `USE`, and that no cross-database updates are issued. If this guideline is not suitable for your environment, Quest recommends that you do not use the `MIXED` Binary Logging Format.

IMPORTANT: The following procedure uses **mysqlbinlog** without the **"--database"** option. Therefore, all the contents of the Binary Log are applied, and all databases might be modified. Consider applying this procedure to an alternate MySQL Server and extract the applicable data from the alternate MySQL Server. If you apply the following procedure to your production MySQL Server, all databases are rolled back to the specified point. Do not apply the procedure in your production environment unless you plan to roll back all your MySQL Server Databases to the specified point.

1. In the Navigation pane, click **Create Restore Job**.
2. On the **Create Restore Job—Choose Saveset** page, click **Table Filtering**, and select **Edit Filters**.
3. From the **Plugin Type** list, select **Plug-in *for MySQL***.
4. In the saveset table, select the saveset that contains the Incremental or Differential Backup with the Binary Logs, and click **Next**.
5. On the **Create Selection Set** page, select **Binary Logs**.

   The Binary Logs are common to all the MySQL Server Databases.
6. On the **Create Selection Set** page, click **Edit Plugin Options**.
7. On the **Point-in-Time Recovery** tab, select the **Perform PIT Recovery** and **Restore Logs to Temporary Directory to Identify Time or Position** options.

   The Binary Logs are restored to a temporary directory located in: ***<NetVaultBackupInstallationDirectory>*/tmp/mysql/*<savesetName>***
8. To apply the Binary Logs manually from a **mysqlbinlog** command prompt, type:

   ```
   mysqlbinlog --stop-datetime="yyyy/mm/dd hh:mm:ss"
   "<NetVaultBackupInstallationDirectory>/tmp/mysql/<savesetName>" |
   mysql -u<user> -p<password>
   ```

   Example:

   ```
   mysqlbinlog --stop-datetime="2018/06/06 15:09:00"
   "/usr/netvault/tmp/mysql/MySQL 59 - DIFF - DIFFERENTIAL (Saveset 86) 15.17 06
   Jun 2017/mysql-bin.000038" | mysql -uroot -p<password>
   ```
9. If your restore sequence includes more than one Incremental Backup that needs to be restored, repeat this procedure for each Incremental Backup.

   The different Incremental Backup Savesets are restored to different subdirectories in the ***<NetVaultBackupInstallationDirectory>*/tmp/mysql** directory. You can then apply the **mysqlbinlog** command in each directory, or you can copy or move all the Binary Logs to a common directory and run **mysqlbinlog**.

Examples of restore scenarios for MySQL Enterprise Backup

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

- Additional step on Linux and UNIX environments for TTS restore
- Full and Incremental Backup restore scenarios
- Additional step for Linux and UNIX environments
- Start services in Windows environments
- TTS only 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. Before restarting the MySQL Server, note that if you are running MySQL in a Linux or Unix system, data files might be written to disk as root user. After restoring a backup you may have to change the ownership of the restored files. Ensure to read the section Additional step for Linux and UNIX environments.

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. Before restarting the MySQL Server, note that if you are running MySQL in a Linux or Unix system, data files might be written to disk as root user. After restoring a backup you may have to change the ownership of the restored files. Ensure to read the section Additional step for Linux and UNIX environments.

Additional step for Linux and UNIX environments

If you are using MySQL Enterprise Backup in a Linux or UNIX environment, verify that the file-ownership and permissions information for the restored data matches what it was before the data was backed up. Because the mysqlbackup script does not record this information during the backup process, the file-ownership might be different after the restore is completed. For more information, see

https://dev.mysql.com/doc/mysql-enterprise-backup/8.0/en/mysqlbackup.restore.html

When MySQL datafiles are restored, file and directory privileges are preserved. However, files might be written to disk as root user. After restoring a backup you may have to change the ownership of the restored files with the user and group of the MySQL Server, usually mysql. For example, to recursively change ownership in a typical installation of MySQL Server, where the data directory is located in **`/var/lib/mysql`**, the following command could be applied:

```
sudo chown -R mysql:mysql  /var/lib/mysql
```

After changing permissions, the MariaDB Server can be restarted:

```
sudo systemctl start mysqld.service
sudo systemctl status mysqld.service
```

Check MySQL documentation for more details.

Start services in Windows environments

Use the Windows Services Console to start the MySQL service, typically named MySQL80. Check MySQL documentation for more details.

TTS 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 copy the prepared Full Backup to the MySQL Server repository and rename one table, submit a job in which you have:
   - Selected the **Copy Back Prepared Full Backup…** option on the **Options** tab.
   - Entered a regular expression pattern in the **Include Tables** field to indicate which tables to include in the restore.

     Example of **Include Tables** field: `database_name\.`
   - Entered a rename request in the **Rename Table** field.

     Example of **Rename Table** field: `original_name to new_name`

Additional step on Linux and UNIX environments for TTS restore

During a TTS Copy Back restore when MySQL table files are restored back to the MySQL data directory, the table files privileges are preserved. However, the table files might be written to disk as **root** user. After performing a TTS Copy Back restore you may have to change the ownership of the restored tables files with the user and group of the MySQL Server user, usually **mysql**. In addition, in the NetVault binary logs, MySQL Server might log an entry stating that ***alter table … import tablespace*** query did not complete.

Ensure, after performing the TTS Copy Back restore, to change ownership of the restored tables files, and to issue an ***alter table … import tablespace*** for each of the restored tables.

For example, if restoring to a table named *new_table* to database *database_name*, and the MySQL data directory is located in the default location, `/var/lib/mysql`, the following commands and query can be applied:

```
sudo chown mysql:mysql  /var/lib/mysql/database_name/new_table*
mysql -u root -p
mysql> alter table database_name.new_table import tablespace;
```

Examples of restore scenarios for Mariadb-backup

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

- Full Backup only restore scenarios of MariaDB
- Full and Incremental Backup restore scenarios of MariaDB
- Additional step for Linux and UNIX environments in MariaDB
- Start services in Windows environments

Full Backup only restore scenarios of MariaDB

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 the MariaDB service and copy the prepared Full Backup to the MariaDB Server repository, submit a job in which you have selected the **Shutdown MariaDB Server and Copy Back…** option on the **Options** tab.
3. Restart the MariaDB Server by entering the applicable command at a command prompt. Before restarting the MariaDB Server, note that if you are running MariaDB in a Linux or Unix system, data files might be written to disk as root user. After restoring a backup you may have to change the ownership of the restored files. Ensure to read the section A**dditional step for Linux and UNIX environments**.

Full and Incremental Backup restore scenarios of MariaDB

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 the MariaDB service and copy the prepared Full Backup to the MariaDB 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 MariaDB Server by entering the applicable command at a command prompt. Before restarting the MariaDB Server, note that if you are running MariaDB in a Linux or Unix system, data files might be written to disk as root user. After restoring a backup you may have to change the ownership of the restored files. Ensure to read the section **Additional step for Linux and UNIX environments**.

Additional step for Linux and UNIX environments in MariaDB

If you are using Mariadb-backup in a Linux or UNIX environment, verify that the file-ownership and permissions information for the restored data matches what it was before the data was backed up. Because the Mariadb-backup utility does not record this information during the backup process, the file-ownership might be different after the restore is completed. For more information, see :

https://mariadb.com/kb/en/full-backup-and-restore-with-mariabackup/

When MariaDB datafiles are restored, file and directory privileges are preserved. However, files might be written to disk as root user. After restoring a backup you may have to change the ownership of the restored files with the user and group of the MariaDB Server, usually **mysql**. For example, to recursively change ownership in a typical installation of MariaDB Server, where the data directory is located in **`/var/lib/mysql`**, the following command could be applied:

```
sudo chown -R mysql:mysql  /var/lib/mysql
```

After changing permissions, the MariaDB Server can be restarted:

```
sudo systemctl start mariadb.service
sudo systemctl status mariadb.service
```

Check MariaDB documentation for more details.

Start services in Windows environments

Use the Windows Services Console to start the MariaDB service, typically named MariaDB. Check MariaDBdocumentation for more details.

Using advanced restore procedures for MySQL Standard/Community

This topic 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 does not overwrite the existing version of that database. This process can be useful when creating a copy of an existing database. To accomplish this process, perform the steps outlined in the following topics.

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

Before 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 with a restore to a different MySQL Instance or MySQL Server.

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**.

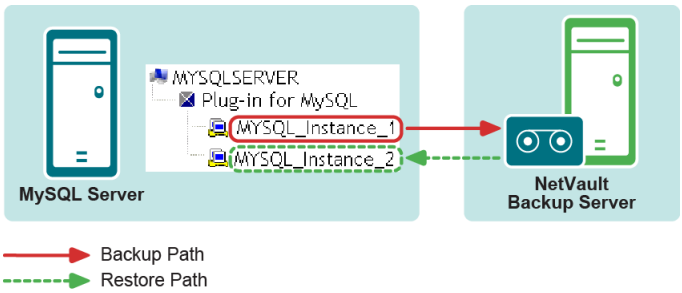
   For more information, see Selecting data for a restore.
2. On the **Create Selection Set** page, click the database that you want to rename, and select **Rename** from the context menu.
3. In the **Rename/Relocate** dialog box, enter the new name in the **Rename to** box, and click **Ok**.

   The database item is accompanied by renaming information in parentheses.
4. Continue with the restore as explained in Restoring data.

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 process, perform the steps outlined in the following topics.

Known limitations and intended uses

Before 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, regarding MySQL:
  - **Same OS installed**
  - **Same version of MySQL installed**
- **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 starting the restore

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

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 by creation date.
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, server name, client name, plug-in name, saveset date and time, retirement setting, Incremental Backup or not, Archive or not, saveset size, and snapshot-based backup or not.
5. 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 login account name used to access the target MySQL Instance.
   - **Password**: Enter the password associated with the login 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 name is the name established as the **MySQL Instance Name** in the **Configure** dialog box; for more information, see Configuring the plug-in.
9. If applicable, 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. To save the settings, click **Ok**, 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 name that lets you easily identify the job when monitoring its progress. The job name can contain alphanumeric and nonalphanumeric characters, but it cannot contain non-Latin characters. On Linux, the name 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 filename on the target OS. For example, the characters /,\,*, and @ should not be used on Windows. This requirement 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.

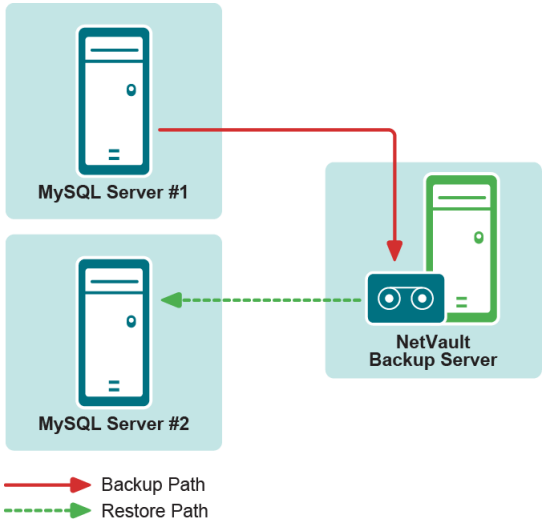
    TIP: You can also click **Choose**, and then locate and select the applicable client in the **Choose the Target Client** dialog box.
13. Use the **Schedule**, **Source Options**, and **Advanced Options** lists to configure any additional required options.
14. Click **Save** or **Save & Submit**, whichever is applicable.

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

Recovering to an alternate MySQL Server

Similar to the plug-in's ability to restore databases or individual tables to a different MySQL Instance on the same MySQL Server, you can target a ***different MySQL Server*** during the restore process. This option is used during disaster recovery operations.

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



To accomplish this process, perform the steps outlined in the following topics.

Known limitations and intended uses

Before conducting a relocation restore to a different 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 and 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, regarding MySQL:
  - **Same OS installed**
  - **Same version of MySQL installed**
  - **Same installation and base directory**
  - **Same MySQL "Data" directory**

- **NetVault Backup software and Plug-in *for MySQL* installed on all clients**: NetVault Backup, Client or 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 through 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 exists on the ***new restore target***. This instance serves 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 box on the new restore target: **Username**, **Password**, and **Instance Name**.

  During setup 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.

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 by creation date.
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, server name, client name, plug-in name, saveset date and time, retirement setting, Incremental Backup or not, Archive or not, saveset size, and snapshot-based backup or not.
5. 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 box.
   - **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 applicable, 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. To save the settings, click **Ok**, 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 name that lets you easily identify the job when monitoring its progress. The job name can contain alphanumeric and nonalphanumeric characters, but it cannot contain non-Latin characters. On

Linux, the name 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 filename on the target OS. For example, the characters /,\,*, and @ should not be used on Windows. This requirement 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 box.

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

14 Click **Save** or **Save & Submit**, whichever is applicable.

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

6

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 and slave setup. If problems occur 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 nonupdating queries dominate, which is the normal case.
- An extra 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 box. For information about accessing this dialog box, see Configuring the plug-in.

- **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 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 the Binary Log. 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 filenames and locations are retained, the plug-in automatically backs up and restores all these files 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, you 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 you can reinitialize the Slave 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 and tables to the master. If you want to resynchronize an individual table or database on a slave, Quest recommends that you use MySQL's Replication process for resynchronization instead of restoring to the slave and then trying to get the slave synchronized with the master.

7

Using the plug-in in a Failover Cluster environment

- MySQL Server Failover Clustering: an overview
- Installing or upgrading 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 to create a **Clustered Service**. A Virtual Service is displayed on the network as if it were a single computer running an application, but provides failover from one node to a different node if the current node becomes unavailable.

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

Important considerations

- Unless outlined in the topics that follow, backups and restores performed with the plug-in of clustered data are the same as backups and restores performed with traditional MySQL Server data.
- The following topics only offer information about 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 *Quest NetVault Backup Administrator's Guide*.
- Before you continue, review all cluster-related information provided in the *Quest NetVault Administrator's Guide* to understand how the following information works with MySQL Server Failover Cluster functionality.

Installing or upgrading the plug-in

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

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.

  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, 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. This machine ***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 the same as the traditional installation of this plug-in. For more information, see Installing and removing the plug-in.

Configuring the plug-in

Perform the following steps on the primary node.

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 primary node.
3. Open **Plug-in *for MySQL*.**
4. Click the **All Instances** node, and select **Configure** from the context menu.
5. On the **Configure** dialog box, set the applicable configuration options.

   The options available are the same as those options covered in Configuring the plug-in.

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

6. If you anticipate having to create more 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. To save the settings, click **OK**.

Backing up data

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.

The instance name displayed in this page is actually the MySQL Clustered Service. If other MySQL Server Clustered Services are running on this node, those instances are also 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, run the process using the primary node. If you open or expand one of the nodes and drill down through the hierarchy, you see the MySQL Clustered Service; 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, do not attempt to run any processes at this level.

Restoring data

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. The only difference is that restorable backups are displayed on the **Create Restore Job—Choose Saveset** page under the name of the primary node that was active during each backup. When a restore job is initiated, NetVault Backup communicates with all member Clients to determine which machine is in control of the failover cluster, and then targets this machine for the restore.

All the instructions offered for performing a restore can be used in the recovery of a failover cluster. For more information, see the various topics in Restoring Data. To restore a failover cluster to a standalone NetVault Client, use the instructions provided in Recovering to an Alternate MySQL Server.

A

Troubleshooting

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

Table 2. Troubleshooting

Error message	Explanation
• Failed to add backup record • Failed to write index of backup to the database	These messages indicate that the selected data was backed up, but the job's index information was not properly added by NetVault to its database. Without this index information, the data cannot be properly restored. **Method 1:** Access the **Manage Devices** page of the NetVault 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 that the information was added, open the **Create Restore Job—Choose Saveset** page and locate the specific job. If you can browse it and set up a restore job, the scan process has corrected the problem. **Method 2:** If the previous method failed, run the backup job again.
Backup fails with a replication error.	If a backup fails with a message similar to "Failed to start Replication slave server," it might indicate that you selected the **Enable MySQL Replication** check box but did not configure replication. To correct this issue, either clear the **Enable MySQL Replication** check box on the **Configure** dialog box or set up replication, and then run the backup job again. For more information about updating the configuration, see Configuring the plug-in; for more information about replication, see Working with native MySQL replication.
In a Linux or UNIX environment, a backup or restore job fails with the following error: Cannot establish connection to mysql server. Connection open fails with error "Can't connect to local MySQL server through socket '/tmp/mysql.sock' (2)"	The job is trying to access the default location, "/tmp/mysql.sock," for the MySQL Server socket file, but the file is located elsewhere. The file might be located in "/var/lib/mysql/mysql.sock" or "/opt/mysql/mysql.sock," or any other location. To address this issue, use the following command to create a symbolic link so that the job can access the socket file. `ln -s <existingFile> <symbolicLinkFile>` For more information about updating the path and filename, see Configuring the plug-in.

About us

Quest provides software solutions for the rapidly-changing world of enterprise IT. We help simplify the challenges caused by data explosion, cloud expansion, hybrid datacenters, security threats, and regulatory requirements. We are a global provider to 130,000 companies across 100 countries, including 95% of the Fortune 500 and 90% of the Global 1000. Since 1987, we have built a portfolio of solutions that now includes database management, data protection, identity and access management, Microsoft platform management, and unified endpoint management. With Quest, organizations spend less time on IT administration and more time on business innovation. For more information, visit www.quest.com.

Technical support resources

Technical support is available to Quest customers with a valid maintenance contract and customers who have trial versions. You can access the Quest Support Portal at https://support.quest.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. The Support Portal enables you to:

- Submit and manage a Service Request.
- View Knowledge Base articles.
- Sign up for product notifications.
- Download software and technical documentation.
- View how-to-videos.
- Engage in community discussions.
- Chat with support engineers online.
- View services to assist you with your product.