How-To Library

Informatica

PowerCenter Repository Maintenance

Abstract

The PowerCenter repository grows over time as you run new jobs on the development and production environments. The PowerCenter administrator must perform certain maintenance tasks on the database that hosts the repository to maintain the size of the database for optimal performance. This article provides information for the PowerCenter administrator about the activities like truncating the workflow and session log files and purging unnecessary versions from the repository to optimize the size of repository and increase its performance.

Supported Versions

- PowerCenter 9.x

Table of Contents

Overview

The PowerCenter repository can grow to a size that may start slowing the performance of the repository and make it increasingly difficult for the administrator to make backups.

The PowerCenter administrator must perform the following maintenance activities on the repository to maintain the size of the database for optimal performance:

- Regularly back up the repository
- Delete old versions or purged objects, folders, and logs from the repository
- Perform repository tuning activities

The administrator can use the Repository Manager client tool, the Informatica Administrator, and the pmrep and infacmd command line programs to administer the repository.

Back Up the Repository Database

You must back up the repositories regularly to prevent data loss due to hardware or software problems. When you back up a repository, the PowerCenter Repository Service saves the repository in a binary file, including the repository objects, connection information, and code page information.

After you back up a repository, you can view the backup files.

Creating a PowerCenter Repository Backup File

When you back up a repository, the PowerCenter Repository Service stores the file in the backup location you specify for the node. You specify the backup location when you set up the node. View the general properties of the node to determine the path of the backup directory.

1. In the Administrator tool, click the Domain tab.
2. In the Navigator, select the PowerCenter Repository Service for the repository you want to back up.
3. On the Domain tab Actions menu, select **Repository Contents** > **Back Up**.
4. Enter your user name, password, and security domain.

   The Security Domain field appears when the Informatica domain contains an LDAP security domain.
5. Enter a file name and description for the repository backup file.

   Use an easily distinguishable name for the file. For example, if the name of the repository is DEVELOPMENT, and the back up occurs on May 7, you might name the file DEVELOPMENTMay07.rep. If you do not include the .rep extension, the PowerCenter Repository Service appends that extension to the file name.
6. If you use the same file name that you used for a previous backup file, select whether or not to replace the existing file with the new backup file.

   To overwrite an existing repository backup file, select Replace Existing File. If you specify a file name that already exists in the repository backup directory and you do not choose to replace the existing file, the PowerCenter Repository Service does not back up the repository.
7. Choose to skip or back up workflow and session logs, deployment group history, and MX data. You might want to skip these operations to increase performance when you restore the repository.
8. Click OK.

   The results of the back up operation appear in the activity log.

Viewing a List of Backup Files

You can view the backup files you create for a repository in the backup directory where they are saved. You can also view a list of existing backup files in the Administrator tool. If you back up a repository through *pmrep*, you must provide a file extension of .rep to view it in the Administrator tool.

1. In the Administrator tool, click the Domain tab.
2. In the Navigator, select the PowerCenter Repository Service for a repository that has been backed up.
3. On the Domain tab Actions menu, select Repository Contents > View Backup Files.

   The list of the backup files shows the repository version and the options skipped during the backup.

Restoring a PowerCenter Repository

You can restore metadata from a repository binary backup file. When you restore a repository, you must have a database available for the repository. You can restore the repository in a database that has a compatible code page with the original database.

If a repository exists at the target database location, you must delete it before you restore a repository backup file.

Informatica restores repositories from the current product version. If you have a backup file from an earlier product version, you must use the earlier product version to restore the repository.

Verify that the repository license includes the license keys necessary to restore the repository backup file. For example, you must have the team-based development option to restore a versioned repository.

1. In the Navigator, select the PowerCenter Repository Service that manages the repository content you want to restore.
2. On the Domain tab Actions menu, click Repository Contents > Restore.

   The Restore Repository Contents options appear.
3. Select a backup file to restore.
4. Select whether or not to restore the repository as new.

   When you restore a repository as new, the PowerCenter Repository Service restores the repository with a new repository ID and deletes the log event files.

   Note: When you copy repository content, you create the repository as new.
5. Optionally, choose to skip restoring the workflow and session logs, deployment group history, and Metadata Exchange (MX) data to improve performance.
6. Click OK.

   The activity log indicates whether the restore operation succeeded or failed.

   Note: When you restore a global repository, the repository becomes a standalone repository. After restoring the repository, you need to promote it to a global repository.

Managing a Versioned Repository

If you have the team-based development option, you can enable version control for a repository. A versioned repository can store multiple versions of an object. After you enable version control for a repository, you cannot disable it.

When you enable version control, you can maintain multiple versions of an object in the repository. This increases the size of the repository and may contain versions of the objects that my no longer be relevant. You must delete old versions or purged objects from the repository to reduce the size of the repository. You can use the repository queries in the client tools to create queries that can determine outdated versions and objects for deletion. Use the Query Browser to run object queries on both versioned and non-versioned repositories.

Old versions and objects increases the size of the repository and make it difficult to manage the repository further into the development cycle. Cleaning up the folders makes it easier to determine what is valid and what is not required.

Tip: Use shortcuts by creating shared folders if you are using the same source or target definition, reusable transformations in multiple folders.

Purging Versions of Objects

You can purge specific versions of objects, or you can purge all versions of objects.

To permanently remove an object version from the repository, you must purge it. You need to check in object versions to purge them. You might want to purge a version if you no longer need it and you want to reduce the size of the repository database.

You can purge multiple versions of an object from the repository at the same time. To completely purge an object from the repository, you must purge all versions. If you purge a version that is not the latest version, the repository keeps a record of the purge in the object history. If you purge the latest version, the repository does not keep a record of the purge.

You use the Repository Manager to purge versions. When you purge versions of objects, you can perform the following tasks:

- **Purge individual object versions.** You can select object versions in the View History window or Query Results window to purge the individual object versions.
- **Purge versions based on criteria.** You can purge versions at the repository, folder, or object level based on purge criteria. This type of purge is called an advanced purge. Use advanced purges to purge deleted or active objects. For deleted objects, you can specify the objects to purge based on the date of deletion. For active objects, you specify the versions to purge based on the version number, the check-in date, or both.
- **Preview purge results.** Preview an advanced purge to view the purge results before you purge objects from the repository. You can view summary or detailed information about the purge.
- **Purge composite objects.** You can purge versions of composite objects, and you can purge versions of dependent objects that make up composite objects. View object dependencies before you purge composite objects. You might get unexpected results if you do not determine the dependent object versions that a purge affects.

The following table shows the Repository Manager commands that you can use to purge versions at the object, folder, or repository level:

Purge Type	Single Object Version	Multiple Object Versions	Versions at Folder Level	Versions at Repository Level
By Object Version (View History Window)	yes	yes	no	no
By Object Version (Query Results Window)	yes	yes	no	no
Based on Criteria (Navigator)	yes	yes	yes	yes
Based on Criteria (View History Window)	yes	yes	no	no
Based on Criteria (Query Results window)	yes	yes	no	no

Purging Individual Object Versions

You can select individual versions of objects in the View History window or the Query Results window to purge the versions.

1. In the Navigator, Select an object and click Versioning > View History.

   Or, click Tools > Query, and run a query from the Query Browser.
2. In the results window, select the object versions to purge.
3. Click Tools > Purge Object Version.
4. In the confirmation message, click Yes.
5. Click OK.

   Warning: When you purge an object version, you might invalidate dependent objects.

Purging Versions Based on Criteria

In the Repository Manager, you can purge object versions based on criteria. This type of purge is called an advanced purge. You can purge object versions at the repository, folder, or object level.

When you purge versions based on criteria, you can perform the following tasks:

- **Purge versions of deleted objects.** Purge versions of checked-in deleted objects to permanently remove the versions from the repository. You can purge all checked-in deleted objects, or you can purge objects that were deleted before a specified date. When you purge deleted objects, you purge all versions of the objects.
- **Purge versions of active objects.** Purge specified checked-in versions of active objects. Active objects are undeleted objects and deleted objects that are not checked in. When you purge versions of active objects, you specify the number of versions to keep, a purge cutoff time, or both. If you specify a number of versions to keep and a purge cutoff time, you purge versions that meet both conditions.
- **Preview versions before purging.** Before you purge versions based on criteria, you can preview the purge results to verify that the purge criteria produces the expected results.

Note: When you purge versions based on criteria, you cannot purge a dependent object version if it is used in an unpurged composite object.

The following table describes the options in the Advanced Purge window:

Option	Description
Purge Deleted Objects	Purges versions of checked-in deleted objects. Select All to purge versions of all deleted objects in a repository or folder, or select Older Than to purge versions of objects deleted before an end time. You can specify the end time either as the number of days before the current date or in MM/DD/YYYY HH24:MI:SS format.
Purge Active Objects	Purges specified versions of active objects. Select Older Than the Last *n* Versions to specify the number of latest checked-in versions to keep. For example, select 6 to purge all versions except the last six checked-in versions. If the object is checked out, you also retain the checked-out version. Select Older Than and specify a number of days or a date and time to purge versions that were checked in before a specified time.
Save Purge List	Output file to save information about purged object versions. Default is disabled.
Summary Only	Saves summary information in the purge output file and displays summary information in purge previews. Disable to view detailed information about each object version. Default is enabled.

The amount of time that the Repository Service takes to purge versions depends on the size of the repository, the number of deleted and old objects, and the composite objects that are affected. For optimal performance, purge at the folder level or use purge criteria to reduce the number of purged object versions. Avoid purging all deleted objects or all older versions at the repository level.

1. In the Navigator, select a repository to purge versions at the repository level.

   Or, select a folder to purge versions from the folder.

   You can also select one or more objects to purge objects based on criteria.

   Note: You can also use the View History window or the Query Results window to purge based on criteria. Select one or more objects in the window, and click Tools > Advanced Purge.
2. Click Versioning > Advanced Purge.

   Alternatively, right-click the repository or folder and select Advanced Purge, or right-click the selected objects and click Versioning > Advanced Purge.
3. To purge deleted objects, select Deleted Objects, and then specify whether to purge all deleted objects or objects deleted before an end date.

   Or, to purge active objects, select Active Objects, and then specify the versions to keep, the purge cutoff, or both. After you purge an object version, you cannot retrieve it. To ensure that you can revert to past versions, avoid purging all versions of an object.
4. Optionally, click Save Purge List to create an output file for the purge information.
5. Optionally, choose to view and save summary information instead of detailed purge information.
6. Optionally, click Preview to preview the purge.
7. Click Purge to purge the deleted objects.

   Tip: When you use an advanced purge to purge deleted objects, you purge all versions of the objects. To keep recent versions of deleted objects and purge older versions, define a query that returns the deleted objects. Then, use the *pmrep* PurgeVersion command with the -q option to retrieve the deleted objects and specify the versions to purge.

Previewing Purge Results

Before you purge versions based on criteria, you might want to preview the purge results. When you preview purge results, check the purge criteria before you purge versions from the repository. Also, review the affected object versions to verify that the Repository Service removes the obsolete versions and retains the versions that you want to keep.

When you preview a purge, you can view summary or detailed information about the purge.

To preview a purge, configure the purge criteria for an advanced purge. Choose to view and save summary or detailed information. Then, click Preview.

In the Preview window, you can click Purge to proceed with the purge, or you can click Cancel to close the Preview window without purging. Click Save To File to save the purge preview results to an output file.

Purging Composite Objects

When you purge versions based on criteria, the purged objects might include composite objects such as mappings or workflows. Before you purge a composite object, you need to consider object dependencies. Object dependencies can affect the way that dependent reusable objects are purged.

If you purge a composite object that consists of non-reusable dependent objects, you also purge the non-reusable dependent objects. If you purge a composite object that contains reusable dependent objects, you purge the dependent object versions if they are not used in another composite object.

You cannot purge a version of a dependent object if it is used in a version of a composite object that you do not purge. Also, if you cannot purge a particular version of an object, you cannot purge more recent versions of that object, even if the more recent versions are not used in composite objects.

This section provides two examples that show how dependencies can affect purges of active objects. The first example describes a frequently modified composite object with rarely updated dependent objects. The second example describes a composite object with few versions but frequently modified dependent objects.

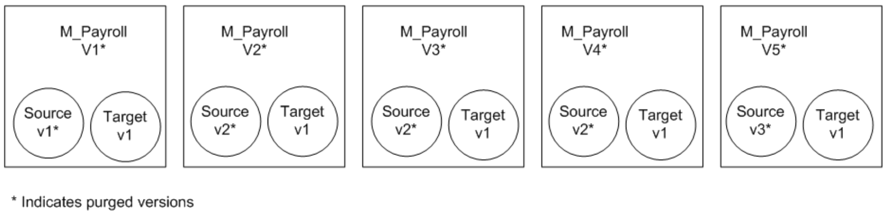
Tip: View dependencies before you purge an object to determine if a dependency might affect the versions that you purge.

Example of a Frequently Checked-Out Composite Object

You update the mapping m_Payroll often, and you frequently check it in and out. Five checked-in versions of the mapping exist. You rarely modify the source and target objects in the mapping. There are three checked-in versions of the source and one checked-in version of the target.

At the repository level, you purge versions based on criteria, and you indicate that you want to keep the last two checked-in versions of objects.

The following figure shows the history of versions 1 through 5 of the mapping:



The advanced purge produces the following results:

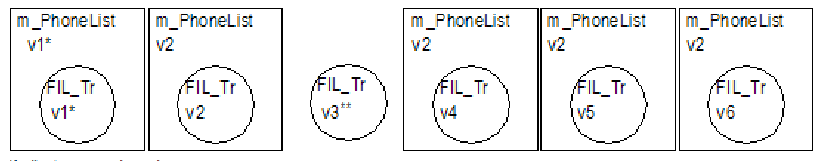
Object	Purged Versions
Mapping m_Payroll	Versions 1 through 3, assuming that no Session task or other composite object uses m_Payroll.
Source	Version 1. Because you purge the version of m_Payroll that uses source version 1, you also purge that version of the source. The purge keeps the last two checked-in versions of objects, so you do not purge versions 2 and 3 of the source.
Target	None. The purge keeps the last two checked-in versions of objects. Only one checked-in version of the target exists.

Example of a Rarely Checked-Out Composite Object

You rarely check in and check out the mapping m_PhoneList. Two checked-in versions of the mapping exist. However, you frequently check in and out the reusable transformation in the mapping. The transformation is a Filter transformation named FIL_Tr. It has six versions.

At the repository level, you purge versions based on criteria, and you specify that you want to keep only the latest checked-in version of objects.

The following figure shows the history of the mapping and transformation versions:



*Indicates purged versions.

**The mapping does not use version 3 of the transformation.

The advanced purge produces the following results:

Object	Purged Versions
Mapping m_PhoneList	Version 1, assuming that no Session task or other composite object uses m_PhoneList.
Transformation FIL_Tr	Version 1. You do not purge versions 2, 4, 5, and 6 of the transformation, because version 2 of m_PhoneList uses those transformation object versions. You do not purge version 3 of the transformation, because you retain version 2, which is an older version.

Note: If you cannot purge an older version of an object, the Repository Service retains all newer versions of the object during an advanced purge.

Rules and Guidelines for Purging Versions of Objects

Use the following rules and guidelines when you purge versions of objects:

- If you purge the latest version of an object and the preceding version has a different name, the preceding version takes the name of purged version. For example, you have the source src_Records. The latest version is named src_Records, but the name of the preceding version in the history is src_RecordsWeekly. If you purge the latest version, the name of the preceding version becomes src_Records.
- When you purge an individual version of a dependent object, you render composite objects invalid if they use the dependent object version. Verify object dependencies before purging individual object versions.
- In an advanced purge of an active object, you cannot purge a version of a dependent object if it is used in an unpurged version of a composite object.
- In an advanced purge of an active object, if you specify a number of versions to keep, you keep the latest checked-in version, even if it was checked in after the purge cutoff time. If the number of versions to keep is greater than the number of object versions, you keep all object versions.

Managing Folders

Folders provide a way to organize and store metadata in the repository, including mappings, schemas, and sessions. Folders help you logically organize the repository.

Deleting Obsolete Folders

If a folder becomes obsolete, you can delete that folder from the repository.

1. In the Repository Manager, connect to a repository and select a folder.
2. Click **Folder** > **Delete**.

3. In the confirmation message that appears, click OK.

Managing Logs

The Service Manager and the application services continually send log events to the Log Manager. As a result, the directory location for the logs can grow to contain a large number of log events.

You can purge logs events periodically to manage the amount of log events stored by the Log Manager. You can export logs before you purge them to keep a backup of the log events.

Purging Log Events

You can automatically or manually purge log events. The Service Manager purges log events from the log directory according to the purge properties you configure in the Log Management dialog box. You can manually purge log events to override the automatic purge properties.

Purging Log Events Automatically

The Service Manager purges log events from the log directory according to the purge properties. The default value for preserving logs is 30 days and the default maximum size for log event files is 200 MB.

When the number of days or the size of the log directory exceeds the limit, the Log Manager deletes the log event files, starting with the oldest log events. The Log Manager periodically verifies the purge options and purges log events. The Log Manager does not purge the current day log event files and folder.

Purging Log Events Manually

You can purge log events for the domain, application services, or user activity. When you purge log events, the Log Manager removes the log event files from the log directory. The Log Manager does not remove log event files currently being written to the logs.

Optionally, you can use the *infacmd* PurgeLog command to purge log events.

The following table lists the purge log options:

Option	**Description**
Log Type	Type of log events to purge. You can purge domain, service, user activity or all log events.
Service Type	When you purge application service log events, you can purge log events for a particular application service type or all application service types.
Purge Entries	Date range of log events you want to purge. You can select the following options: - All Entries. Purges all log events. - Before Date. Purges log events that occurred before this date. Use the yyyy-mm-dd format when you enter a date. Optionally, you can use the calendar to choose the date. To use the calendar, click the date field.

Time Zone

When the Log Manager creates log event files, it generates a time stamp based on the time zone for each log event. When the Log Manager creates log folders, it labels folders according to a time stamp. When you export or purge log event files, the Log Manager uses this property to calculate which log event files to purge or export. Set the time zone to the location of the machine that stores the log event files.

Verify that you do not lose log event files when you configure the time zone for the Log Manager. If the application service that sends log events to the Log Manager is in a different time zone than the master gateway node, you may lose log event files you did not intend to delete. Configure the same time zone for each gateway node.

Note: When you change the time zone, you must restart Informatica Services on the node that you changed.

Configuring Log Management Properties

Configure the Log Management properties in the Log Management dialog box.

1. In the Administrator tool, click the Logs tab.
2. On the Log Actions menu, click Log Management.
3. Enter the number of days for the Log Manager to preserve log events.
4. Enter the maximum disk size for the directory that contains the log event files.
5. Enter the time zone in the following format:

   ```
   GMT(+|-)<hours>:<minutes>
   ```

   For example: GMT+08:00
6. Click OK.

Repository Performance Tuning

You can use the Informatica features to improve the performance of the repository. You can update statistics and skip information when you copy, back up, or restore the repository.

Repository Statistics

Almost all PowerCenter repository tables use at least one index to speed up queries. Most databases keep and use column distribution statistics to determine which index to use to execute SQL queries optimally. Database servers do not update these statistics continuously.

In frequently used repositories, these statistics can quickly become outdated, and SQL query optimizers might not choose the best query plan. In large repositories, choosing a sub-optimal query plan can have a negative impact on performance. Over time, repository operations gradually become slower.

Informatica identifies and updates the statistics of all repository tables and indexes when you copy, upgrade, and restore repositories. You can also update statistics using the pmrep UpdateStatistics command.

Repository Copy, Back Up, and Restore Processes

Large repositories can contain a large volume of log and historical information that slows down repository service performance. This information is not essential to repository service operation. When you back up, restore, or copy a repository, you can choose to skip the following types of information:

- Workflow and session logs
- Deployment group history
- Metadata Exchange (MX) data

By skipping this information, you reduce the time it takes to copy, back up, or restore a repository.

You can also skip this information when you use the *pmrep* commands.

Truncating Workflow and Session Log Entries

When you configure a session or workflow to archive session logs or workflow logs, the Integration Service saves those logs in local directories. The repository also creates an entry for each saved workflow log and session log. If you move or delete a session log or workflow log from the workflow log directory or session log directory, you can remove the entries from the repository.

Use the Repository Manager to truncate the list of workflow logs for workflows that have completed. You can also use the *pmrep* TruncateLog command to truncate workflow logs.

You can truncate all log entries for a workflow or log entries that were created before a specified date. You cannot truncate a workflow log for a workflow that is still running. The Repository Service truncates the workflow log list and the session log list at the same time.

When the Repository Service truncates log entries for sessions and workflows, it also deletes the following run-time information for the sessions and workflows:

- Workflow details
- Session statistics
- Task details
- Source and target statistics
- Partition details
- Performance details

  Tip: You can truncate log entries to reduce the size of the repository in the database. The log entries can use a significant amount of space in the repository.

Note: When you truncate log entries from a Microsoft SQL Server repository, verify that no workflow is running. If you truncate log entries when a workflow is running, the workflow fails.

To truncate workflow and session log entries:

1. In the Repository Manager, select the workflow in the Navigator window or in the Main window.
2. Choose Edit > Truncate Log.

   The Truncate Workflow Log dialog box appears.
3. Choose to delete all workflow and session log entries or to delete all workflow and session log entries with an end time before a particular date.
4. If you want to delete all entries older than a certain date, enter the date and time.
5. Click OK.

   The Repository Service deletes the workflow and session log entries from the repository.

Author

Pradeep Venkateshlu