Using Git Version Control System for Model Repository
Abstract
You can integrate the Model repository with the Perforce, Subversion, or Git version control systems. This article discusses how to integrate Git as a version control system for Model repository in version 10.2 HotFix 1.

Supported Versions

- Big Data Management 10.2 HotFix 1

Table of Contents

Version Control for the Model Repository Service ........................................ 2
Working of Git Version Control System .......................................................... 3
Configuring Git as Version Control System for Model Repository Service .... 3
Configuring the Local Repository Path for Git Version Control System ........ 5
How to Migrate to Git Version Control System From Perforce or SVN Version Control System 5
How to Configure and Synchronize a Model Repository with a Version Control System ............. 6

Version Control for the Model Repository Service
You can integrate a Model repository with a version control system. A version control system protects Model repository objects from overwriting objects on a team where multiple developers work on the same projects. A Model repository can use only one version control system instance at a time.

You can integrate the Model repository with the Perforce, Subversion, or Git version control system. Perforce and Subversion are centralized version control systems. You might lose data if the Perforce or Subversion version control system server is not accessible or the server unexpectedly shuts down. Git is a distributed version control system. When you check in an object, a copy of the version is saved to the local repository and the Git server. If the Git server unexpectedly shuts down, the local repository retains all the versions of the object.

To integrate the Model repository with the Git version control system, you need the following repository details:

Global repository.
The global repository is the Git server. You need the URL, user name, and password for the Git server. Use HTTPS protocol to create the global repository.

Local repository.
The local repository is a directory on the node where you installed the Model Repository Service. The directory must be accessible by the client machines. If you have enabled high availability for Model repository, the directory must be accessible to the backup nodes in the domain. The local repository can support NFS, FAT32, and NTFS file systems. You can use only one local repository for a global repository.

To use Git version control system in AIX operating system, you need to configure the following parameters:

- In Informatica Administrator, configure the ACJVMCommandLineOptions=-Dhttps.protocols=TLSv1.2 environment variable before you start the domain.

- In Informatica Administrator, navigate to the Model Repository Service > Advanced Properties section and append -Dhttps.protocols=TLSv1.2 to the existing JVM Command Line Options field. You can also use the infacmd mrs UpdateServiceProcessOptions command to add the -Dhttps.protocols=TLSv1.2 option.

When the Model repository is integrated with a version control system, you can check in revised objects, undo the checkout of objects, and reassign the checked-out state of objects to another user.
Working of Git Version Control System

The following steps explain how the Git version control system works:

1. You check out an object in the Model repository.
2. The Model repository locks the object.
3. After you make your changes to the object, you check in the object.
4. The object is unlocked and a version is created.
5. The version is saved to the local repository and global repository.

Configuring Git as Version Control System for Model Repository Service

To connect to Git version control system, you must configure the versioning properties in the Model Repository Service.

1. In Informatica Administrator, navigate to Manage > Services and Nodes > Domain Navigator.
2. In the Domain Navigator, click the Model Repository Service you want to integrate with Git version control system.
3. In the Properties > Versioning Properties section, click the Edit icon. The Edit Versioning Properties dialog box appears.
4. In the Edit Versioning Properties dialog box, you can enter the following details:
   • Select Enable Version Control System option.
   • In the Version control system type field, choose GIT.
   • In the URL field, enter the URL for the global repository.
   • In the Username field, enter the user name for the global repository.
   • In the Password field, enter the password for the global repository.

The following image shows a sample Edit Versioning Properties dialog box:
5. In the **Edit Versioning Properties** dialog box, click **OK**.

The configured properties appears in the **Versioning Properties** section.

6. Configure the **git_local_repository_path** custom property to add the local repository path for the Model Repository Service.

7. Restart the Model repository.

   Before you restart the service, ask Model repository users to close all editable objects. When the Model repository restarts, it checks whether the version control system is in use by another Model repository. If the version control system is connected to a different Model repository, the Model Repository Service cannot restart. You must configure the versioning properties to connect to a version control system that has not been integrated with another Model repository.

8. Recycle the Model Repository Service.

9. To synchronize the Model repository contents to the version control system, perform one of the following tasks:

   - In Informatica Administrator, select the Model repository, and click **Actions > Synchronize With Version Control System**, and click **OK**.
   - Run the `infacmd mrs PopulateVCS` command.

When you synchronize the content, the Model repository populates the local directory and global repository with Model repository objects. After the synchronization is complete, versioning is active for Model repository objects. All Model repository objects are checked in to the version control system. Users can check out, check in, view version history, and retrieve historical versions of objects.
Configuring the Local Repository Path for Git Version Control System

To configure a local repository for the Git version control system, you need to configure the `git_local_repository_path` custom property.

1. In Informatica Administrator, navigate to the Manage > Services and Nodes > Model Repository Service > Properties section.
2. In the Custom Properties section, click the Edit icon.
   The Edit Custom Properties dialog box appears.
3. In the Edit Custom Properties dialog box, click New.
4. In the New Custom Property dialog box, enter the following details:
   - Name. Enter `git_local_repository_path` as the custom property.
   - Value. Enter the local repository path for the Git version control system.
5. Click OK.
6. In the Edit Custom Properties dialog box, click OK.
7. Recycle the Model Repository Service.

How to Migrate to Git Version Control System From Perforce or SVN Version Control System

To enable Git version control, configure the versioning properties, and synchronize the Model repository with the version control system.

The following image shows the process to migrate to Git version control system from Perforce or SVN version control system:
1. Configure versioning properties and recycle the Model Repository Service.
2. Synchronize the Model repository content with the version control system.
3. You can change the versioning properties, such as the global repository URL, username, or password.
4. After you change the versioning properties, configure the `git_local_repository_path` custom property to enter the local repository path for the Model Repository Service.
5. Recycle the Model Repository Service.
6. Re-synchronize the Model repository content with the version control system.

You can perform these tasks from the command line or from the Administrator tool.

**Note:** When you change Model repository properties, you must recycle the Model Repository Service for your changes to take effect. Ask users to save changes and close Model repository objects that they have open for editing. While synchronization is in progress, the Model repository is unavailable.

**How to Configure and Synchronize a Model Repository with a Version Control System**

You can enable version control, configure versioning properties and then synchronize the Model repository with the version control system. After you configure versioning and synchronize the Model repository with the version control system, the version control system begins to save version history.

The following image shows the process of configuring, synchronizing, and re-synchronizing the Model repository with a version control system:
1. Configure the versioning properties and recycle the Model Repository Service.
2. Synchronize the Model repository content with the version control system.
3. Optionally, change the version control system type.
   a. For Perforce, you can change the host, port number, username, or password.
   b. For SVN, you can change the URL, port number, username, or password.
c. For Git, you can change the global repository URL, username, or password. Configure the `git_local_repository_path` custom property to add the local repository path for the Model Repository Service.
   After you change the versioning properties, you can choose to retain or discard the version control history:
   a. Retain version control history. Copy content from local repository to new local repository.
   b. Discard version control history.

4. Recycle the Model Repository Service
5. Re-synchronize the Model repository content with the version control system.

You can perform these tasks from the command line or from the Administrator tool.

**Note:** When you change Model repository properties, you must recycle the Model Repository Service for your changes to take effect. When you enable version control system or change a versioning property, the Model repository remains unavailable until you synchronise the Model repository.

---

**Author**

Lavanya S
Senior Technical Writer

**Acknowledgements**

The author would like to acknowledge Saman Naheed for her technical assistance.