madcap software

USER GUIDE

MADCAP LINGO 12

Source Control: Git

Copyright © 2026 MadCap Software. All rights reserved.

Information in this document is subject to change without notice. The software described in this document is furnished under a license agreement or nondisclosure agreement. The software may be used or copied only in accordance with the terms of those agreements. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or any means electronic or mechanical, including photocopying and recording for any purpose other than the purchaser's personal use without the written permission of MadCap Software.

MadCap Software
1660 17th Street, Suite 201
Denver, Colorado 80202
858-320-0387
www.madcapsoftware.com

THIS PDF WAS CREATED USING MADCAP FLARE.

CONTENTS

CHAPTER 1

Introduction 5

CHAPTER 2

General Information for Git 6
- Common Source Control Terms 7
- Source Control Icons 8
- Bind Detection, Disabling Providers, and Unbinding Providers 9

CHAPTER 3

Process for Git 12
- Binding a Project to Git 13
- Importing From Git 21
- Pulling Files From a Remote Repository 24
- Committing Source Control Files 27
- Synchronizing Source Control Files 29
- Pushing Files to a Remote Repository 32
- Merging Source Control Files 35

CHAPTER 4

Branch Activities for Git 38

Creating Branches 39
Publishing Branches 42
Switching Branches 46
Getting Remote Branches 52
Merging Branches 54
Reverting Branches 57
Deleting Branches 60

CHAPTER 5

Other Activities for Git 64
Adding and Editing an Ignore File 65
Disabling the Get Latest Prompt for Source Control 69
Disabling a Git Provider 70
Enabling Source Control Status Checks 73
Modifying Network Settings 74
Reverting Modified Source Control Files 77
Rolling Back to an Earlier Version of a File 78
Setting Color Options for Project File Differences 84
Unbinding a Git Provider From a Project 86
Using Git for Windows 90
Viewing Differences in Source Control Files 92
Viewing Modified Files 95
Viewing the History of Source Control Files 97

APPENDIX

PDFs 99
Cheat Sheets 99
User Guides 99

CHAPTER 1

Introduction

Git is a source control system that uses distributed development, as opposed to a centralized workflow used by other systems. This means that each writer has a local Git repository, as well as being bound to a remote repository (created via GitHub, Bitbucket, etc.), which other authors may also be connected to. As each author works, changes are made and committed in the local repository, which can then be synchronized with the remote repository. You can use the Lingo interface to perform various source control tasks for a project that is bound to Git. Alternatively, you can use a third-party solution to perform tasks.

General Information

- "Common Source Control Terms" on page 7
- "Source Control Icons" on page 8
- "Bind Detection, Disabling Providers, and Unbinding Providers" on page 9

Process

1. Install and Set Up Git (done outside of Lingo)
2. "Binding a Project to Git" on page 13
3. (Other Team Members) "Importing From Git" on page 21
4. "Pulling Files From a Remote Repository" on page 24
5. "Committing Source Control Files" on page 27
6. "Synchronizing Source Control Files" on page 29 or "Pushing Files to a Remote Repository" on page 32
7. "Merging Source Control Files" on page 35

CHAPTER 2

General Information for Git

There are various pieces of general information you should know if you plan to use this feature.

This chapter discusses the following:

Common Source Control Terms 7

Source Control Icons 8

Bind Detection, Disabling Providers, and Unbinding Providers 9

Common Source Control Terms

Following are definitions for some of the common phrases used in Lingo's integrated source control with Git.

- **Bind** This means to connect your project to Git. After doing this, you can then take advantage of all the automated source control tasks (such as commit, revert, pull, push, and so on).
- **Commit** This means to record changes to your Lingo files to the local Git repository.
- **Revert** This means to undo changes you have made to a Lingo branch or file. Changes are reverted to the way they were at the last commit.
- **Synchronize** This pulls the files from the remote Git repository and merges them with your local database, then pushes your local changes back to the remote Git repository.
- **Pull** This retrieves commits from the remote Git repository and merges them with the files in your local database.
- **Push** This sends commits from your local database to the remote Git repository.
- **Branch** This creates a new path for commits. This lets you create new versions of a file while keeping the original file unchanged (e.g., for documenting a new feature, testing page layouts, or rewriting existing information). You can create as many branches as you want.

NOTE Lingo integrates with multiple source control providers to provide built-in source control support. Each of the source control providers built-in to Lingo uses different terms. As such, Lingo's source control interface is different depending on which source control provider you use. Please refer to the sections for each source control provider if you need to see information about the terms used by other built-in systems.

Source Control Icons

Following are descriptions for the primary icons that you may see next to files when using Git.

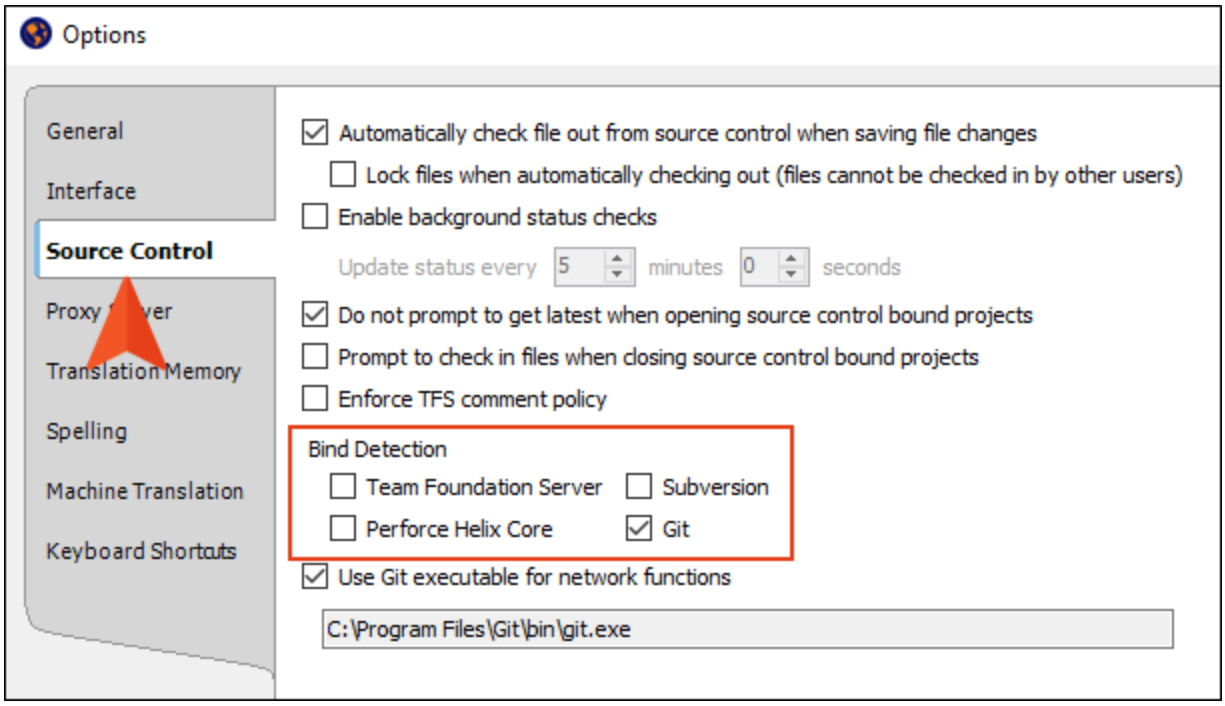
Icon	Description
✔	Modified This indicates that the file has been modified. You can commit the file to your local Git repository when you are ready.
✚	New File (Pending Add) This indicates that you have a file in your project but have not yet added it to Git. This might occur, for example, if you create a new topic and do not add the file to source control during the topic creation process. To resolve this, simply right-click on the file and select Source Control > Add.

Bind Detection, Disabling Providers, and Unbinding Providers

Lingo has options for bind detection, disabling providers, and unbinding providers. Although these are separate features, they are all somewhat related. This information is especially important if you are using an external tool to bind and manage your source control tasks.

Bind Detection

Lingo's bind detection settings are found on the Source Control tab of the Options dialog.



Bind detection scans your project when you load it to see if the project has been previously bound to source control. If a binding is detected, you then have the option of applying the binding and committing the project to source control. Depending on the provider you are using, Lingo may search the file system and its artifacts, as well as contact and query servers, to find potential source control bindings.

When you open a Lingo project that hasn't been bound to source control before, the bind detection option is disabled for Perforce Helix Core, Subversion, and Team Foundation Server. It is enabled by default for Git. If you bind a project to source control using the Lingo interface, the option is then automatically enabled.

NOTE If you bind a project to Git using the Lingo interface, the detection will automatically happen behind the scenes. If you use a tool outside of Lingo for the Git binding, you may or may not want to enable bind detection. For example, you might create a folder where you store all of your Lingo projects, and you use an external tool to create the bindings, with a .Git folder stored at the root level of that main folder. In that case, you would want the bind detection option on this tab disabled for Git.

NOTE You can use bind detection as an alternative to importing a Lingo project. If you have received a Lingo project file (e.g., by copying it from a server, by opening it from a network location), you can simply open the file and Lingo will search for and apply existing source control bindings.

NOTE Source control providers are scanned in the following order:

1. Git
2. Subversion
3. Perforce Helix Core

TIP Detecting source control bindings may take a considerable amount of time. It is recommended that you select only the source control providers that you use to speed up the detection process.

Disabling Providers

By default, when a project is bound to source control, the provider (Git, Perforce Helix Core, Subversion, or Team Foundation Server) is enabled. This means that the source control interface elements in Lingo are visible, and you can use them to perform various tasks (e.g., commits, synchronize changes).

Disabling a provider means that the source control interface elements are no longer shown. This does not mean you cannot use source control. As long as the provider is still *bound* to the project, you can perform source control tasks in a third-party tool outside of Lingo.

For more details and steps, see "Disabling a Git Provider" on page 70.

Unbinding Providers

When you unbind a provider, it means you are removing the connection altogether between the Lingo project and the local repository.

You can unbind a provider via the Project Properties dialog or the Settings view in the Source Control Explorer. Click the Unbind Provider button.

For more details and steps, see "Unbinding a Git Provider From a Project" on page 86.

CHAPTER 3

Process for Git

Certain tasks must be completed in order when using this feature.

This chapter discusses the following:

Binding a Project to Git 13
Importing From Git 21
Pulling Files From a Remote Repository 24
Committing Source Control Files 27
Synchronizing Source Control Files 29
Pushing Files to a Remote Repository 32
Merging Source Control Files 35

Binding a Project to Git

Use the following steps if you have already created a Lingo project and want to bind ("connect") it to Git. You can also automatically detect existing source control bindings if your project has been previously connected to Git.

When using Git for source control, every Lingo project will have a local repository. So after you bind a Lingo project to Git, you will see a folder named ".git" alongside your other project folders in Windows Explorer. You will see a .gitignore file, which prevents unnecessary folders and files from being included in the source control processes.

As you work in your project, changes can be committed to your local repository. Then, you would periodically synchronize (i.e., pull, push) files with your remote repository. This allows other people on your team to get your changes, and it lets you get their changes as well.

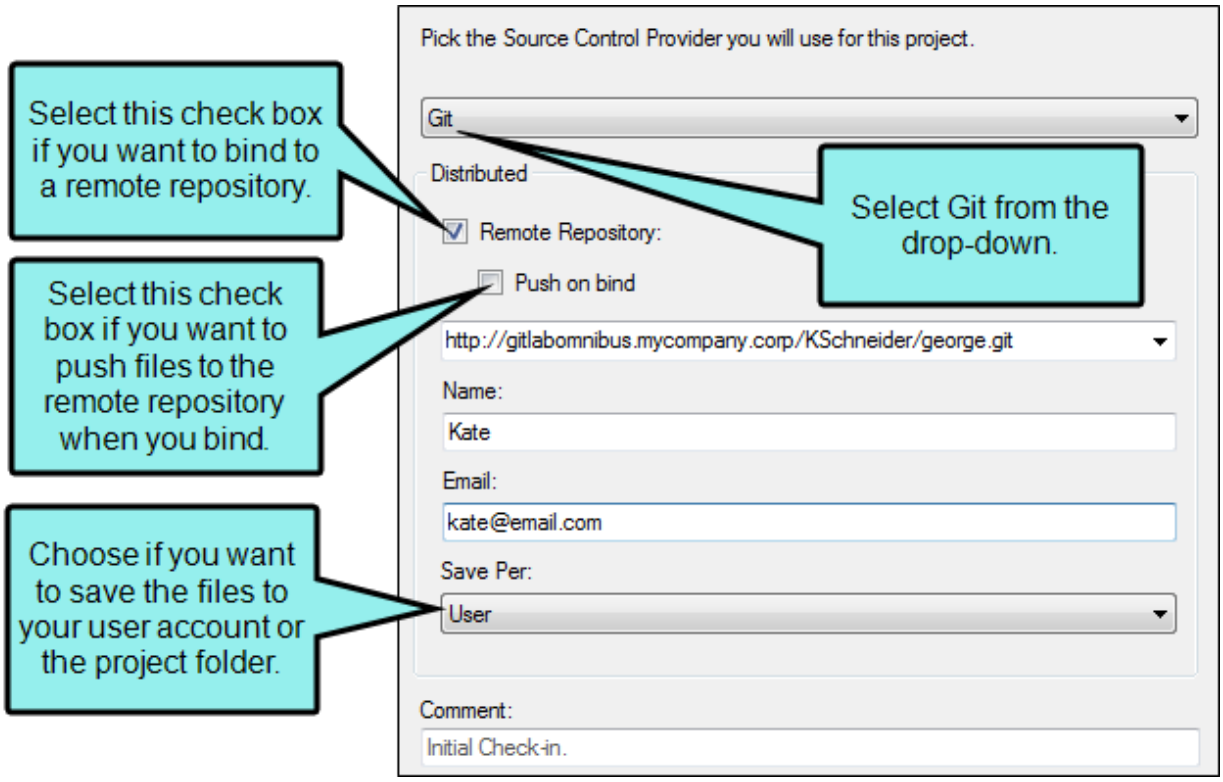
NOTE The following steps show how to bind a project using the Lingo interface. It is also possible to bind a project outside of Lingo (e.g., using Git Bash). If you decide to do this, you should be aware of some additional aspects of source control, such as bind detection and disabling providers.

NOTE Each project bound to Git must have its own source control repository. Also, when first binding a project to Git, the repository must be empty. For example, if you are using a third-party solution such as GitHub to create a remote repository, you might initially be presented with options (such as README or .gitignore) that add files to your new repository. If you select any of these options and then attempt to bind the project to Git, you will experience an error that looks something like this: "Destination already in use by another project. Please push to an empty remote."

Therefore, you should avoid selecting any such options when creating your remote repository. After you bind the project to the remote Git repository, you can then return to that repository and add the README or other files as necessary.

How to Bind a Project Using the Project Properties Dialog

1. Select **File > Project Properties**.
2. Select the **Source Control** tab.
3. Click **Bind Project**. The Bind Project dialog opens.
4. From the drop-down, select **Git**.



5. If you are going to bind to a remote repository, select **Remote Repository**, then enter the address of the repository in the field. If you want to work locally, you can leave this box unselected. You can enter an HTTP URL or an SSH URL.

> NOTE You may need to obtain this information from your system administrator.

6. If you want to push to the remote repository when you bind the project, select the **Push on bind** check box. This will push the initial project files to the repository when you bind the project.
7. In the **Name** field, enter your name.
8. In the **Email** field, enter your email address.
9. From the **Save Per** drop-down, select how you want to save your files.
   - **User** Saves the files in your local folder. Select this option if you are working with other tools (e.g., Tortoise) or if you want to use the same user identity across multiple projects.
   - **Project** Saves the files locally to your project. Select this option if you want to use different identities for each project.
10. (Optional) In the **Comment** field, you can enter any internal comments.
11. In the Bind Project dialog, click **OK**.
12. In the Project Properties dialog, click **OK**.
13. (Optional) If you entered an HTTP address, enter your user name and password in the Log In dialog. Click **OK** when you are finished.
14. (Optional) If you entered an SSH URL, the Certificate Specification dialog opens. In the dialog, do the following and click **OK** when you are finished:
    a. In the **Public key** field, enter your public SSH key, or use ··· to browse for the key on your network.
    b. In the **Private key** field, enter your private SSH key, or use ··· to browse for the key on your network.
    c. If you want Lingo to remember your key information so you do not need to enter it again later, select the check box next to **Save certificate information**.

NOTE SSH keys allow you to establish a secure connection between your computer and your Git source control provider (likewise, using an SSH URL is more secure than an HTTP URL; you need to use SSH keys if you want to use an SSH URL). If you do not have a public and private SSH key, your can generate these keys using your Git source control provider (e.g., Gitlab). Follow the directions provided by your source control provider to add these keys to your Git account. Once you generate these keys, they are typically found in the C:\Users\[username]\.ssh folder on your computer.

Public keys typically have a .pub extension. Private keys use the same file name as the public key, but without the file extension.

EXAMPLE Use the following as guides when setting up your URLs and keys.

SSH URL

```
git@gitlabomnibus.mycompany.corp:MyUserName/myproject.git
```

HTTP URL

```
http://gitlabomnibus.mycompany.corp/MyUserName/myproject.git
```

SSH Private Key

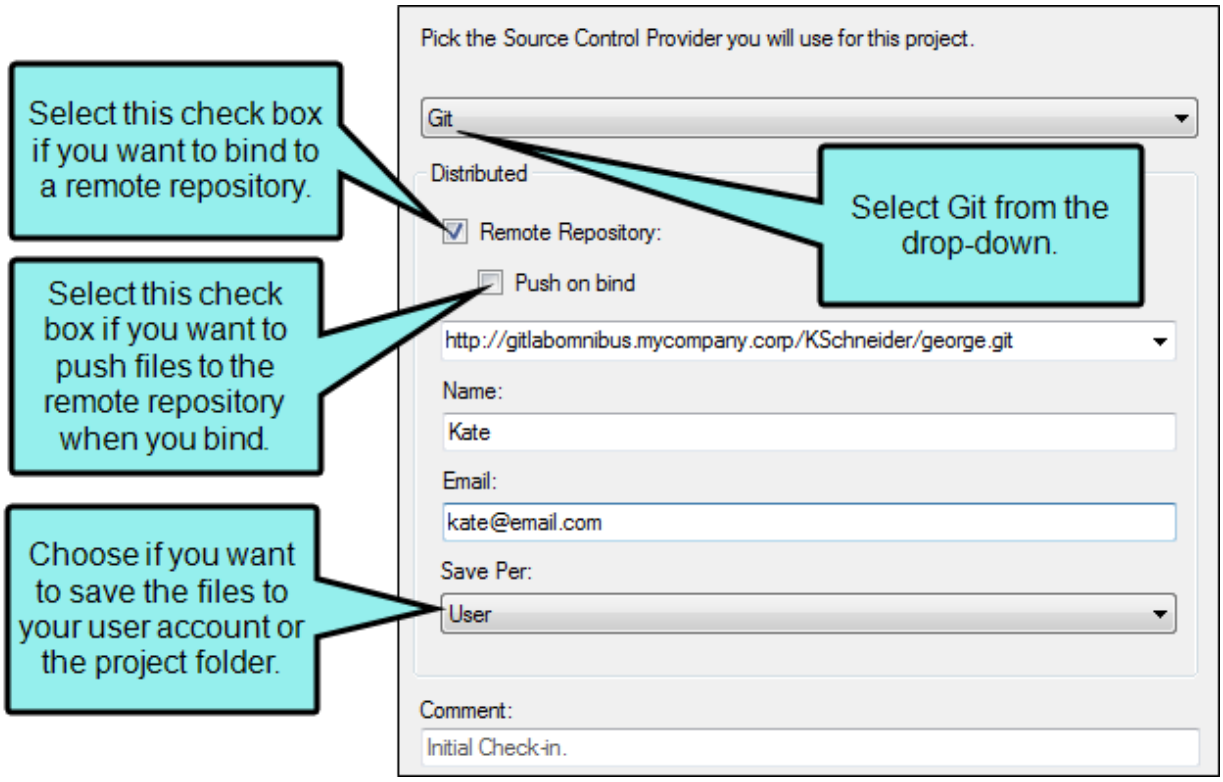
```
mycompanySSHkey_id_rsa
```

SSH Public Key

```
mycompanySSHkey_id_rsa.pub
```

How to Bind a Project Using the Explorer

1. Select **View > Source Control Explorer**. The Source Control Explorer opens.
2. From the drop-down or the Home pane, select **Settings**. The Settings pane opens.
3. Click **Bind**. The Bind Project dialog opens.
4. From the drop-down, select **Git**.



5. If you are going to bind to a remote repository, select **Remote Repository**, then enter the address of the repository in the field. If you want to work locally, you can leave this box unselected. You can enter an HTTP URL or an SSH URL.

> **NOTE** You may need to obtain this information from your system administrator.

6. If you want to push to the remote repository when you bind the project, select the **Push on bind** check box. This will push the initial project files to the repository when you bind the project.
7. In the **Name** field, enter your name.

8. In the **Email** field, enter your email address.
9. From the **Save Per** drop-down, select how you want to save your files.
    - **User** Saves the files in your local folder. Select this option if you are working with other tools (e.g., Tortoise) or if you want to use the same user identity across multiple projects.
    - **Project** Saves the files locally to your project. Select this option if you want to use different identities for each project.
10. (Optional) In the **Comment** field, you can enter any internal comments.
11. In the Bind Project dialog, click **OK**.
12. In the Project Properties dialog, click **OK**.
13. (Optional) If you entered an HTTP address, enter your user name and password in the Log In dialog. Click **OK** when you are finished.
14. (Optional) If you entered an SSH URL, the Certificate Specification dialog opens. In the dialog, do the following and click **OK** when you are finished:
    a. In the **Public key** field, enter your public SSH key, or use ··· to browse for the key on your network.
    b. In the **Private key** field, enter your private SSH key, or use ··· to browse for the key on your network.
    c. If you want Lingo to remember your key information so you do not need to enter it again later, select the check box next to **Save certificate information**.

NOTE SSH keys allow you to establish a secure connection between your computer and your Git source control provider (likewise, using an SSH URL is more secure than an HTTP URL; you need to use SSH keys if you want to use an SSH URL). If you do not have a public and private SSH key, your can generate these keys using your Git source control provider (e.g., Gitlab). Follow the directions provided by your source control provider to add these keys to your Git account. Once you generate these keys, they are typically found in the C:\Users\[username]\.ssh folder on your computer.

Public keys typically have a .pub extension. Private keys use the same file name as the public key, but without the file extension.

EXAMPLE Use the following as guides when setting up your URLs and keys.

SSH URL

```
git@gitlabomnibus.mycompany.corp:MyUserName/myproject.git
```

HTTP URL

```
http://gitlabomnibus.mycompany.corp/MyUserName/myproject.git
```

SSH Private Key

```
mycompanySSHkey_id_rsa
```

SSH Public Key

```
mycompanySSHkey_id_rsa.pub
```

What’s Noteworthy?

TIP If you are having difficulty binding your project, try binding to a brand new directory in your source control provider. You should also ensure that the directory on your local machine (and its parent directories) is not already mapped to source control, as this may cause issues with binding.

NOTE You can also bind a new Lingo project to source control while creating it.

NOTE When you bind a project to Lingo using Git, a .gitignore text file is created in your local project folder. Advanced users can edit the .gitignore file with a text editor to control which files or folders can be pushed to your Git repository. You can also specify which files and folders are ignored by Git, and are not pushed to the repository.

Importing From Git

This topic focuses on importing a Lingo project from source control. You might use this method, for example, if you are working on a project with several other translators and another member of the team has placed the Lingo project in Git.

How to Import a Project From Git

1. Do one of the following, depending on the part of the user interface you are using:
   - Ribbon Select File > New Project > Import Project.
   - Source Control Explorer From the View ribbon, open the Source Control Explorer. From the drop-down, select the Home pane. Click Import Project.

   The Import Project from Source Control Wizard dialog opens.
2. From the drop-down, select Git.
3. In the Remote drop-down, enter the remote repository where the project is located.

   NOTE You may need to obtain this information from your system administrator.
4. In the Name field, enter your name.
5. In the Email field, enter your email address.
6. From the Save Per drop-down, select how you want to save your files.
   - User Saves the files in your local folder. Select this option if you are working with other tools (e.g., Tortoise) or if you want to use the same user identity across multiple projects.
   - Project Saves the files locally to your project. Select this option if you want to use different identities for each project.
7. Click Next.

8. Next to the **Project** field, click **Browse**. The Browse Source Control Files dialog opens. (You may need to log in with your user name and password.)
9. (Optional) If the remote you selected in Step 3 requires an SSH certificate, the Certificate Specification dialog opens. In the dialog, do the following:
    a. In the **Public key** field, enter your public SSH key, or use ··· to browse for the key on your network.
    b. In the **Private key** field, enter your private SSH key, or use ··· to browse for the key on your network.
    c. If you want Lingo to remember your key information so you do not need to enter it again later, select the check box next to **Save certificate information**.

> NOTE SSH keys allow you to establish a secure connection between your computer and your Git source control provider (likewise, using an SSH URL is more secure than an HTTP URL; you need to use SSH keys if you want to use an SSH URL). If you do not have a public and private SSH key, your can generate these keys using your Git source control provider (e.g., Gitlab). Follow the directions provided by your source control provider to add these keys to your Git account. Once you generate these keys, they are typically found in the **C:\Users\[username]\.ssh** folder on your computer.
>
> Public keys typically have a .pub extension. Private keys use the same file name as the public key, but without the file extension.

EXAMPLE Use the following as guides when setting up your URLs and keys.

SSH URL

```
git@gitlabomnibus.mycompany.corp:MyUserName/myproject.git
```

HTTP URL

```
http://gitlabomnibus.mycompany.corp/MyUserName/myproject.git
```

SSH Private Key

```
mycompanySSHkey_id_rsa
```

SSH Public Key

```
mycompanySSHkey_id_rsa.pub
```

10. Find and click on the Lingo project file (LIPRJ) that you want to import. (You may need to log in with your user name and password.)
11. Click **OK**.
12. Click **Next**..
13. In the **Project name** field, the name of the project being imported is displayed. It is recommended that you leave the name as it is, especially if you are working with other users on the project. However, you can enter a different project name if you want.
14. In the **Project folder** field, either accept the default location for the new project or click ... to browse for and select a folder.
15. Click **Finish**. The project is imported and loaded into Lingo.

NOTE If you want to import a project from source control, you can alternatively open the project file from another location (e.g., a server location), and then use Lingo's bind detection functionality to automatically apply available source control bindings to the project.

Pulling Files From a Remote Repository

If you need to update the files in your local database, you can do a pull. When you pull files, Lingo retrieves files from the remote repository and downloads them to your local Git database. This updates your current branch with the most current version of the file. This is a good way to get changes that other users have made to a file and pushed to the remote.

When you pull, you will see the Resolve Version Conflict dialog if conflicts exist between the remote files and the files in your local database.

How to Pull Files From a Remote Repository

1. Do one of the following, depending on the part of the user interface you are using:
   - **Ribbon** Select **Source Control > Pull**.
   - **Right-Click** In the File List, right-click any file and select **Source Control > Project > Pull**.
2. (Optional) If you did not commit your files before starting the pull, a dialog asks if you want to commit your files. Click **Yes** to continue. See "Committing Source Control Files" on page 27.

   > NOTE You must commit *all* modified files to proceed with the pull.

3. The Select Remote for Pull dialog opens. From the **Remote** drop-down, select the repository you want to pull from.

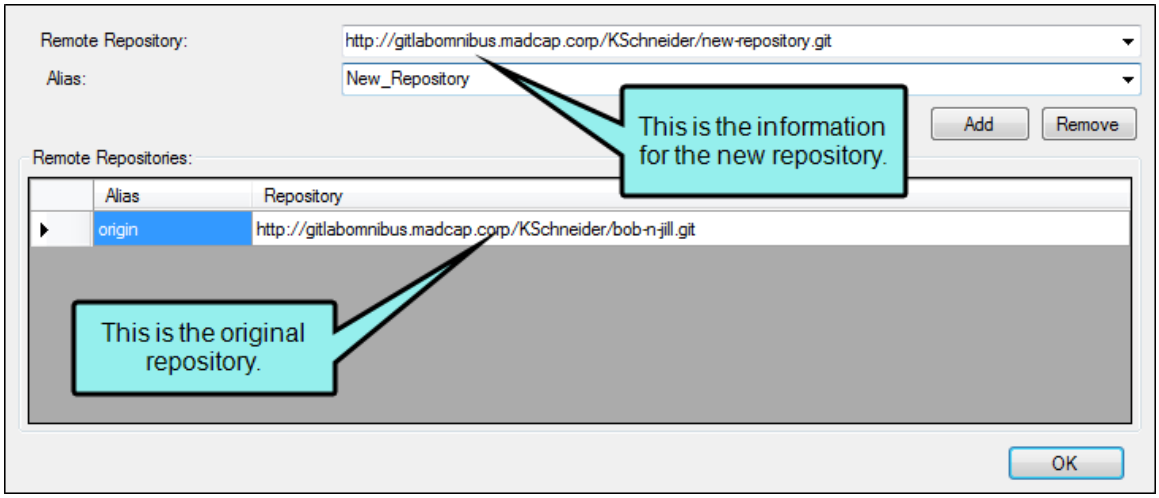
4. (Optional) To add or remove a remote repository, click ⋯. The Remote Repositories dialog opens.

TO ADD A REMOTE REPOSITORY

a. In the **Remote Repository** field, enter the address of the new repository.

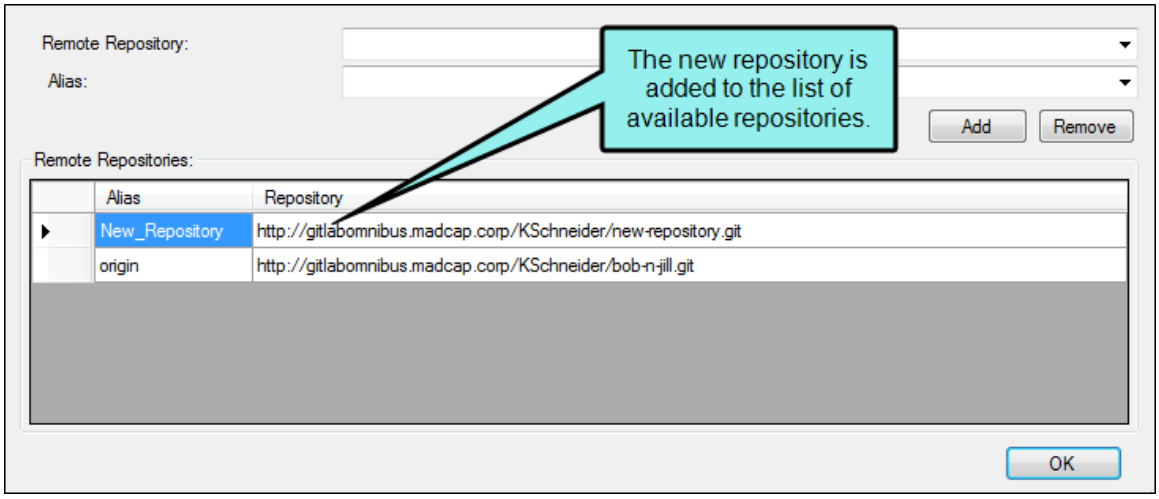
> **NOTE** You may need to obtain this information from your system administrator.

b. In the **Alias** field, enter a name for the new repository. This is the name that will appear in the **Remote** drop-down when you need to select a remote repository.



> **NOTE** Repository names cannot include spaces.

c. Click **Add** to add the new repository to the list of available repositories.



d. Click **OK**.

TO REMOVE A REMOTE REPOSITORY

a. From the **Remote Repository** list, select the repository you want to delete.

b. Click **Remove**. The repository is removed from the list of available repositories.

c. Click **OK**.

5. In the Select Remote for Pull dialog, click **OK**. If the Log In dialog opens, complete the **User name** and **Password** fields and click **OK**.

6. If the remote and local files are the same, a message tells you so. Click **OK**.

   If a remote file is different from the version in your local database, the Resolve Conflicts dialog opens. If you want to accept all of the differences between the remote and local files, thus merging them, click **Auto Merge All**. If you want to review the differences in the files side by side and resolve each conflict, click **Resolve**. For more information about merging files and resolving conflicts, see "Merging Source Control Files" on page 35.

Committing Source Control Files

When you are finished editing files, you can commit them to source control. Committing a file adds your changes to the local Git database. It is a good idea to periodically commit files so that they are organized in logical chunks with a comment that accurately summarizes the changes. When you are ready to add your local commits to the remote repository, you can push these files to the remote. Committing and pushing changes frequently can help to avoid conflicting changes from other users, although conflicts are bound to occur from time to time.

How to Commit Files to Source Control

1. Do one of the following, depending on the part of the user interface you are using:
   - **Status Bar** In the lower-right of Lingo, click (a number indicates how many files have changes that need to be committed).

     > NOTE If you do not see this option, make sure **View > Status Bar** is enabled.

   - **Ribbon** Select **Source Control > Commit** (for selected files) or **Source Control > Commit All** (for all files in the project).
   - **Right-Click** In the File List, right-click the file you want to commit and select **Source Control > Commit** (for selected files) or **Source Control > Project > Commit All** (for all files in the project).

   The Commit dialog opens. The selected files are listed with check boxes next to them.
2. Enter a comment tied to the commit. This enables you to keep an audit trail for a file. The comment can then be viewed from the History dialog, which can be accessed from the Source Control Explorer, the Source Control ribbon, or the Source Control button .
3. Click **Commit**.

How to Commit Files to Source Control Using the Explorer

1. Select **View > Source Control Explorer**. The Source Control Explorer opens.
2. From the drop-down or the Home pane, select **Pending Changes**.

   The Pending Changes pane opens. Files that will be committed are listed under **Included Changes**, and files that will not be committed are listed under **Excluded Changes**. You can identify edited files because [modified] is displayed next to the file name.

   > NOTE When you modify a file in source control, you may sometimes see a SKL file alongside the XLF and original files. This skeleton file is a placeholder file. Be sure to commit all three associated files together. Committing just the SKL file may result in errors.

3. In the **Comment** field, enter a comment tied to the commit. This enables you to keep an audit trail for a file. The comment can then be viewed from the History dialog, which can be accessed from the Source Control Explorer, the Source Control ribbon, or the Source Control button .
4. (Optional) If you want to select the files or folders that you include in the commit, right-click a file or folder and select one of the following options from the context menu.
   - **Exclude** Excludes the selected file from the commit
   - **Exclude Unselected** Excludes all unselected files from the commit
   - **Include** Includes the selected file in the commit
   - **Include Unselected** Includes all unselected files in the commit
5. Click **Commit Included** to commit all of the files in the Included Changes list.

Synchronizing Source Control Files

Synchronizing files performs a pull to update the local database with files from the remote repository, and then pushes any local commits to the remote repository. Changes made during a synchronize affect your current branch. You will see the Resolve Version Conflict dialog if conflicts exist when you synchronize your files.

If a remote file is different from the version in your local database, the Resolve Conflicts dialog opens. If you want to accept all of the differences between the remote and local files, thus merging them, click **Auto Merge All**. If you want to review the differences in the files side by side and resolve each conflict, click **Resolve**

How to Synchronize Files

1. Do one of the following, depending on the part of the user interface you are using:
   - **Status Bar** In the lower-right of Lingo, click . A number next to the up arrow indicates how many commits need to be pushed. If you see a number next to the down arrow on this button, it indicates that there are pending remote commits that you still need to pull to your local repository. A number displays next to the down arrow only after you do a fetch (i.e., "git fetch"). This is a command that you cannot perform from the Lingo interface, but you can do it using another tool such as Git Bash. Another way for a number to display next to the down arrow is if you first do a pull on one branch, then switch to another branch that has pending remote commits.

     > NOTE If you do not see this option, make sure **View > Status Bar** is enabled.

     > NOTE After committing changes, you might need to click somewhere in the interface before numbers populate next to this button.

   - **Ribbon** Select **Source Control > Synchronize**.
   - **Right-Click** In the File List, right-click any file and select **Source Control > Project > Synchronize**.

2. (Optional) If you did not commit your files before starting the synchronize, a dialog asks if you want to commit your files. Click **Yes** to continue. See "Committing Source Control Files" on page 27.

> **NOTE** You must commit *all* modified files to proceed with the synchronization.

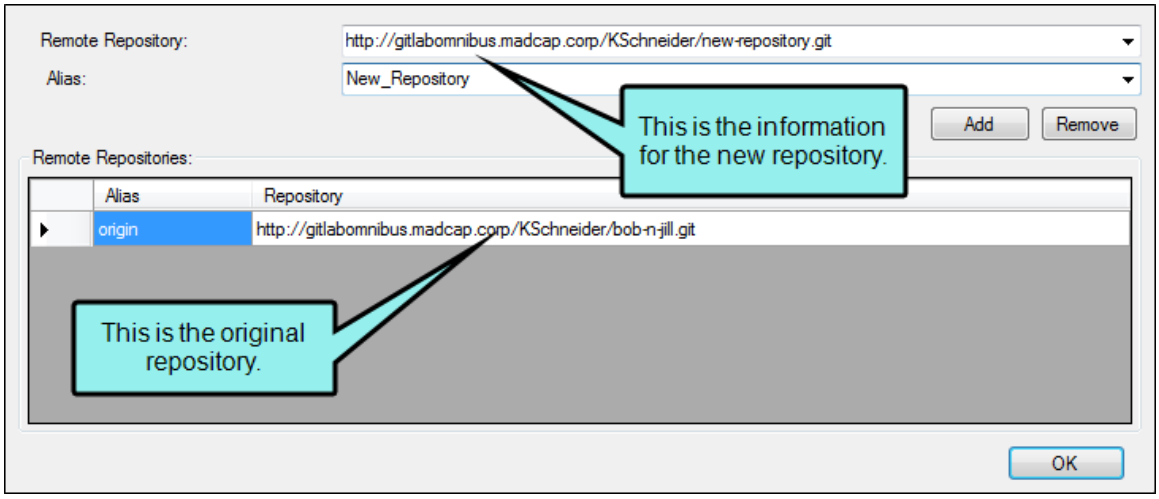
3. The Select Remote for Synchronize dialog opens. From the **Remote** drop-down, select the repository you want to synchronize with.
4. (Optional) To add or remove a remote repository, click ▪. The Remote Repositories dialog opens.

TO ADD A REMOTE REPOSITORY

a. In the **Remote Repository** field, enter the address of the new repository.

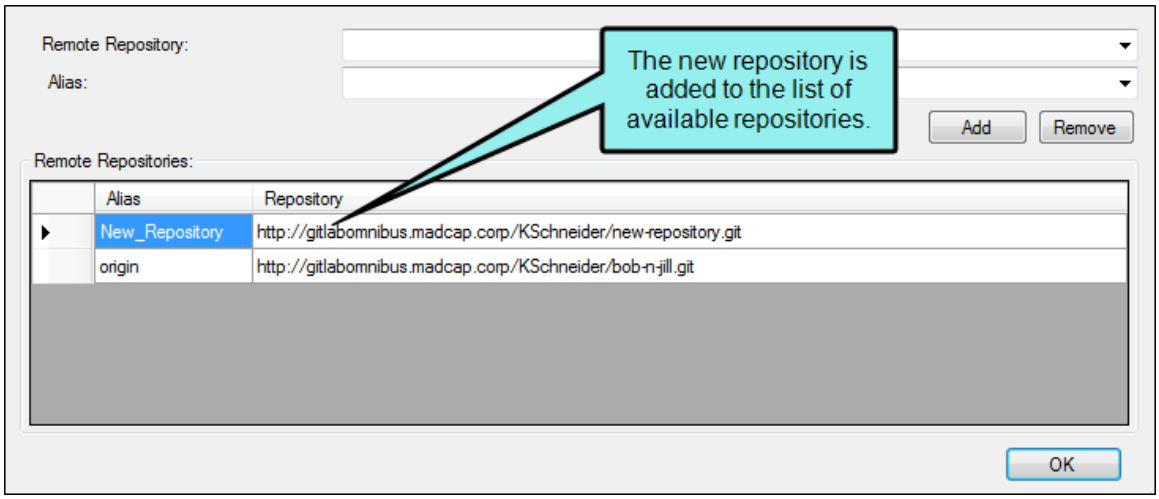
> **NOTE** You may need to obtain this information from your system administrator.

b. In the **Alias** field, enter a name for the new repository. This is the name that will appear in the **Remote** drop-down when you need to select a remote repository.



> **NOTE** Repository names cannot include spaces.

c. Click **Add** to add the new repository to the list of available repositories.



d. Click **OK**.

TO REMOVE A REMOTE REPOSITORY

a. From the **Remote Repository** list, select the repository you want to delete.

b. Click **Remove**. The repository is removed from the list of available repositories.

c. Click **OK**.

5. In the Select Remote for Synchronize dialog, click **OK**. If the Log In dialog opens, complete the **User name** and **Password** fields and click **OK**.

6. If a remote file is different from the version in your local database, the Resolve Conflicts dialog opens. If you want to accept all of the differences between the remote and local files, thus merging them, click **Auto Merge All**. If you want to review the differences in the files side by side and resolve each conflict, click **Resolve** For more information about merging files and resolving conflicts, see "Merging Source Control Files" on page 35.

Pushing Files to a Remote Repository

After you commit the changes from your project to the local Git repository, you can push them to the remote Git repository. Performing a push sends all of the files you have committed on your current branch to the remote repository. This is a way to back them up to source control and for other users to have access the changes you have made.

How to Push Files To a Remote Repository

1. Do one of the following, depending on the part of the user interface you are using:
   - **Status Bar** In the lower-right of Lingo, click 2↑ 0↓ (a number next to the up arrow indicates how many commits need to be pushed).

     > NOTE If you do not see this option, make sure **View > Status Bar** is enabled.

   - **Ribbon** Select **Source Control > Push**.
   - **Right-Click** In the File List, right-click a file and select **Source Control > Project > Push**.
2. (Optional) If you did not commit your files before starting the push, a dialog asks if you want to commit your files. Click **Yes** to continue. See "Committing Source Control Files" on page 27.

   > NOTE You must commit *all* modified files to proceed with the push.

3. The Select Remote for Push dialog opens. From the **Remote** drop-down, select the repository you want to push to.

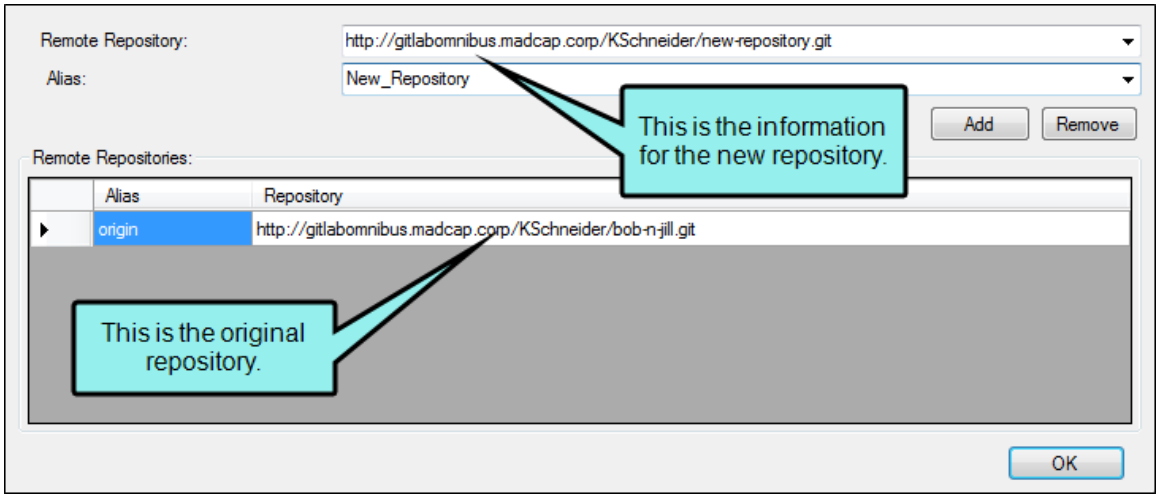
4. (Optional) To add or remove a remote repository, click ⋯. The Remote Repositories dialog opens.(Optional)

TO ADD A REMOTE REPOSITORY

a. In the **Remote Repository** field, enter the address of the new repository.

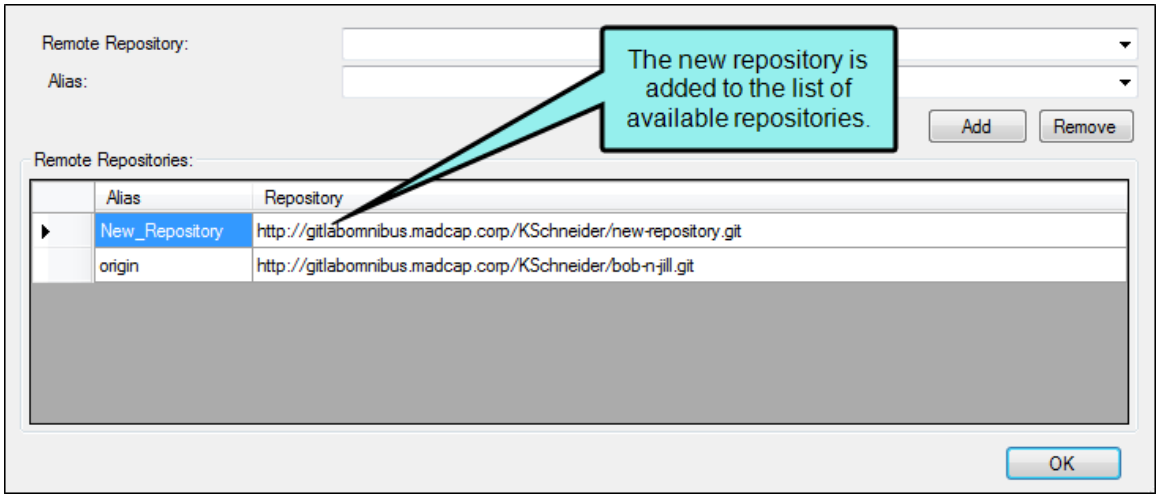
> **NOTE** You may need to obtain this information from your system administrator.

b. In the **Alias** field, enter a name for the new repository. This is the name that will appear in the **Remote** drop-down when you need to select a remote repository.



> **NOTE** Repository names cannot include spaces.

c. Click **Add** to add the new repository to the list of available repositories.



d. Click **OK**.

TO REMOVE A REMOTE REPOSITORY

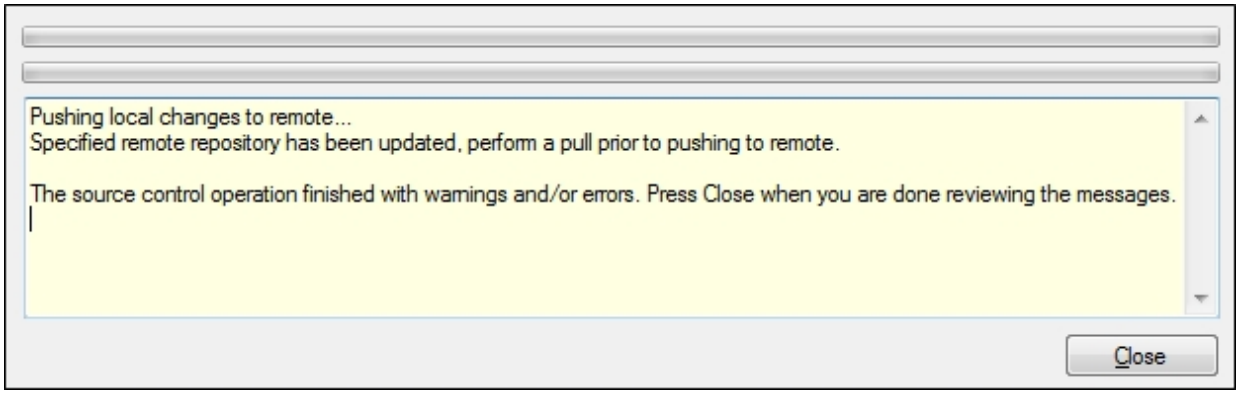
a. From the **Remote Repository** list, select the repository you want to delete.

b. Click **Remove**. The repository is removed from the list of available repositories.

c. Click **OK**.

5. In the Select Remote for Push dialog, click **OK**. If the Log In dialog opens, complete the **User name** and **Password** fields and click **OK**.

6. If the local changes are different from those in the remote, you will see an error in the Progress dialog during the push. Perform a pull, then attempt the push again. See "Pulling Files From a Remote Repository" on page 24.



Merging Source Control Files

There may be times when you need to merge changes from different authors when synchronizing or pulling files. The merge occurs automatically if there are no conflicting changes (i.e., changes do not occur in the same location in the file). If there are conflicting changes, a dialog opens, allowing you to determine how changes are merged.

How to Merge Source Control Files

1. Synchronize your files with the remote repository. If your local copy of the file is different from the remote copy (e.g., another user has already pushed the same file to the remote), the Resolve Conflicts dialog opens. See "Synchronizing Source Control Files" on page 29.
2. Click **Auto Merge All**. If changes from the other user do not conflict with your changes, this will merge all changes. A message lets you know that a backup of your local copy has been made. This lets you restore that file if you do not want to keep the merged version. You do not need to complete the rest of the steps below.

   However, if your changes conflict with those from another user, a message displays to tell you. In this case, continue with the next step.
3. Click **OK** on the conflict message.
4. In the Resolve Conflicts dialog click **Resolve**. The Resolve Version Conflict dialog opens. From this dialog, you can choose from the following options.
   - **Merge changes for me** Automatically merges changes within the same file that are not part of the same element. If changes have been made to the same element (e.g., the same segment), Lingo will display a prompt to merge the changes using the merge tool.
   - **Merge changes in merge tool** Opens a merging interface, which lets you see exactly what changes were made and choose which to keep.
   - **Undo my local changes** Automatically removes your changes and keeps changes from other users.
   - **Discard external changes** Automatically removes changes from other users and keeps your changes.

5. If you selected the option to use the merge tool, the Merge Changes dialog opens. Use this dialog to view and select changes. You can take actions in the following ways.
   - Click a change You can click a change on either the remote or local side. This lets you select a particular change. Use the key at the top of the merge changes dialog, as well as the color coding on the local and server sides, to determine if a change has been added (new), deleted, changed, or moved.

     When you select a change, the change you selected will display with a solid colored background, and the conflicting change will display with a striped background. If you select the other change, the background shading will switch.
   - Type content If you want to use your changes as well as those from another user, and even tweak the paragraph a bit more, you can click in the area at the bottom of the dialog and simply type content.
   - Previous/next conflict When you are finished resolving the first conflict, you can use the "Previous Conflict" and "Next Conflict" buttons at the bottom of the dialog to work on other conflicts in the file.
6. After all conflicts have been resolved, a message lets you know that a backup of your local copy has been made. This lets you restore that file if you do not want to keep the merged version. Click OK.

EXAMPLE

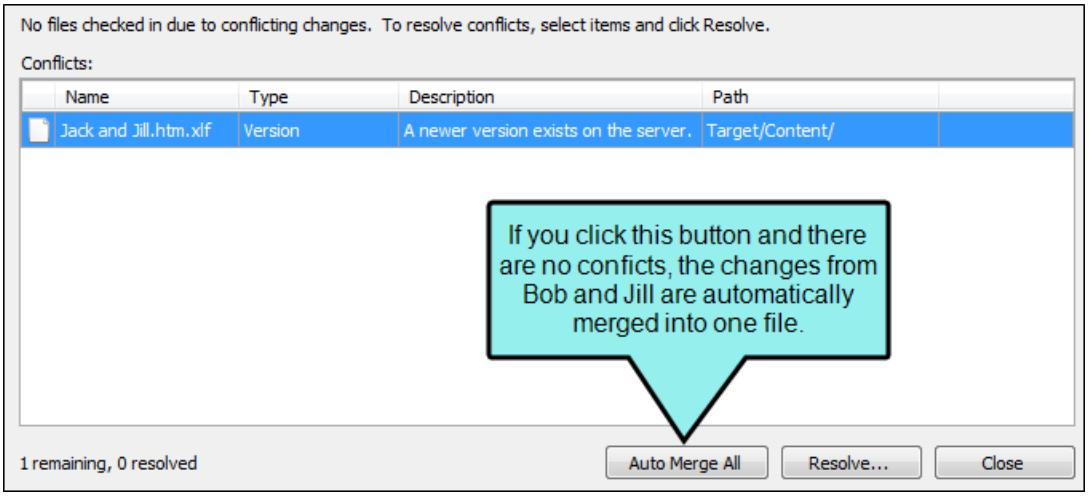
Two translators—Bob and Jill—are working on the same project, using source control to manage the files.

Bob starts making changes to his local version of the "Jack and Jill" topic.

Jill also starts making changes to her local version of the topic. Jill's changes are in a different segment in the topic than Bob's changes; there are no conflicts. She finishes before Bob and synchronizes her files.

Bob finishes his changes and tries to synchronize his files. During the pull, Bob is prompted with a dialog, indicating that changes from another user have already been made.

Bob selects **Auto Merge All**. The changes from Bob and Jill are now both shown in the merged topic. When either translator pulls the topic again or synchronizes their files, they will see the new version of the topic.



CHAPTER 4

Branch Activities for Git

A Git branch is a pointer to a snapshot of your changes, or you can think of it as a variation from the original or main state of your files. Adding a branch lets you create a new development area for your work. Then later, you can merge the branch into another one.

This chapter discusses the following:

Creating Branches 39
Publishing Branches 42
Switching Branches 46
Getting Remote Branches 52
Merging Branches 54
Reverting Branches 57
Deleting Branches 60

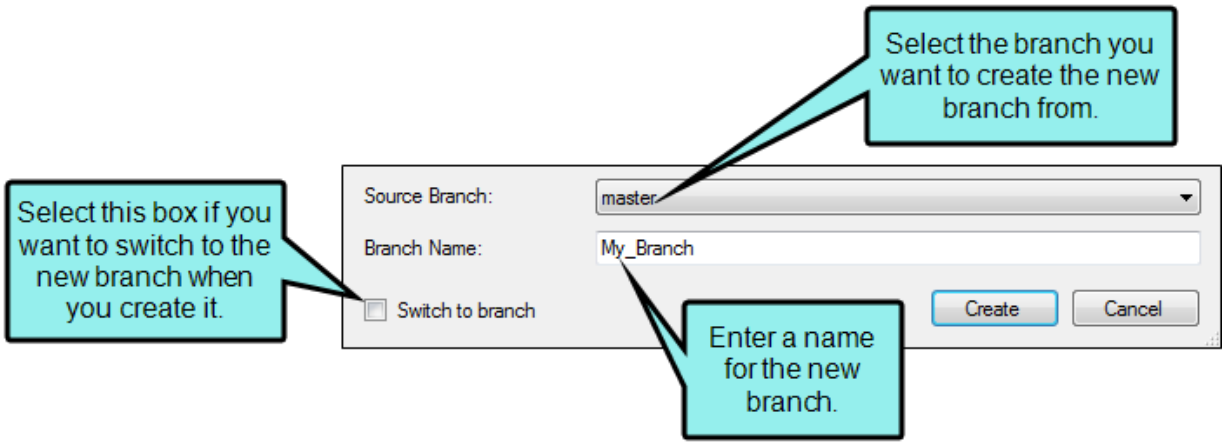
Creating Branches

When you are working in Git, you can create branches.

How to Create Branches—Branch Management Dialog

1. Do one of the following, depending on the part of the user interface you are using:
   - Ribbon Select Source Control > Branch.
   - Right-Click If you have the File List open, right-click on any file and select Source Control > Project > Branch.

   The Branch Management dialog opens.
2. Click Create. The Create Branch dialog opens.
3. From the Source Branch drop-down, select the existing branch you want to use to create the new branch. Your new branch will copy the existing files from the source branch, but commits you make on the new branch will not affect the source branch.
4. In the Branch Name field, enter a name for your branch. Names cannot include spaces.
5. (Optional) If you want to switch the branch when you create it, select the Switch to branch check box.

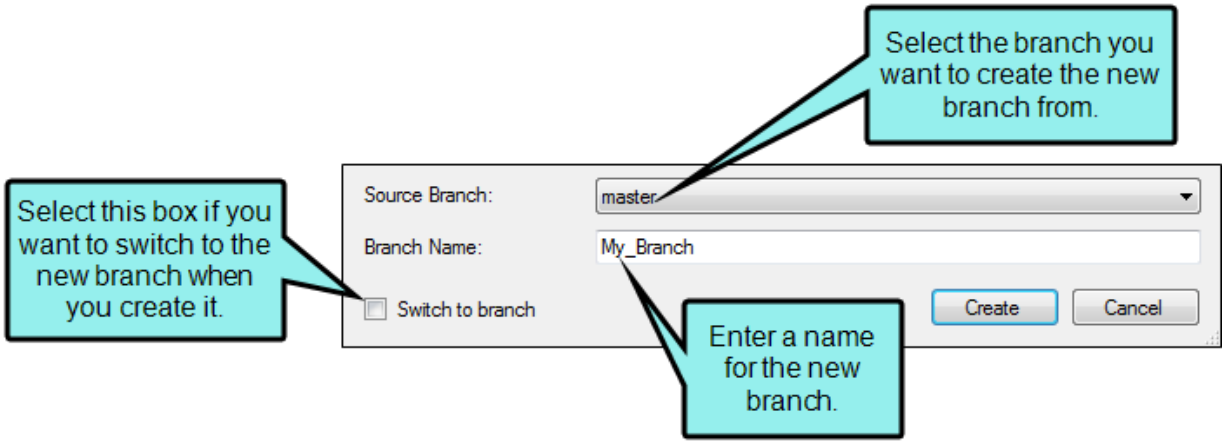


> **WARNING** You will not see a confirmation message when switching to a newly created branch. Be sure all of your current changes are committed before creating and switching to a new branch. If you have uncommitted changes, they will be lost. See "Committing Source Control Files" on page 27.

6. Click **Create**. Your new branch is added to the "All branches" list. If you chose to switch to the branch, you can now begin working on the branch.

How to Create Branches—Source Control Explorer

1. Select **View > Source Control Explorer**. The Source Control Explorer opens.
2. From the drop-down or the Home pane, select **Branches**. The Branches view opens. In the pane, you can see your current branch, as well as lists of your published and unpublished branches.
3. Click **Create Branch**. The Create Branch dialog opens.
4. From the **Source Branch** drop-down, select the existing branch you want to use to create the new branch. Your new branch will copy the existing files from the source branch, but commits you make on the new branch will not affect the source branch.
5. In the **Branch Name** field, enter a name for your branch. Names cannot include spaces.
6. (Optional) If you want to switch the branch when you create it, select the **Switch to branch** check box.



> **WARNING** You will not see a confirmation message when switching to a newly created branch. Be sure all of your current changes are committed before creating and switching to a new branch. If you have uncommitted changes, they will be lost. See "Committing Source Control Files" on page 27.

7. Click **Create**. Your new branch is added to the "All branches" list. If you chose to switch to the branch, you can now begin working on the branch.

Publishing Branches

If you want to push to, pull from, or synchronize using a branch you have created, you must publish the branch. You can still commit to an unpublished branch, but until you publish, you will not be able to send your commits to the remote repository. It is a good idea to publish branches unless you are only using a branch locally for a short period of time.

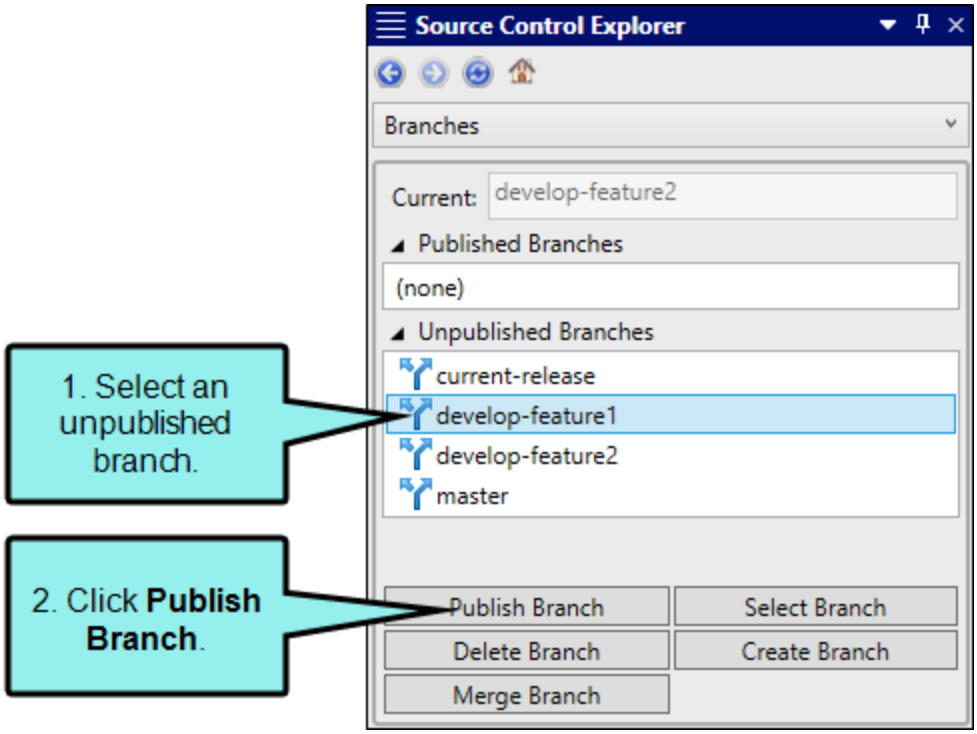
You can publish branches in the following ways:

- Automatic If your current branch is unpublished, pushing, pulling, or synchronizing automatically publishes the branch. See "Pushing Files to a Remote Repository" on page 32, "Pulling Files From a Remote Repository" on page 24, and "Synchronizing Source Control Files" on page 29.
- Manual Select an unpublished branch in the Source Control Explorer and manually publish it. You can select any branch, even if it is not your current branch.

How to Manually Publish Branches

1. Select **View > Source Control Explorer**. The Source Control Explorer opens.
2. From the drop-down or the Home pane, select **Branches**.

   The Branches pane opens. In the pane, you can see your current branch, as well as lists of your published and unpublished branches.
3. Select the unpublished branch you want to publish.
4. Click **Publish Branch**.



5. (Optional) If you did not commit your files before publishing the branch, a dialog asks if you want to commit your files. Click **Yes** to continue. See "Committing Source Control Files" on page 27.

> **NOTE** You must commit *all* modified files before you can publish the branch.

6. The Select Remote for Publish dialog opens. From the **Remote** drop-down, select the repository you want to push to.

> **NOTE** If you have pending changes that you are committing while publishing your branch, you will see the Select Remote for Push dialog. However, the steps for pushing while publishing a branch are the same. See "Pushing Files to a Remote Repository" on page 32 for additional information.

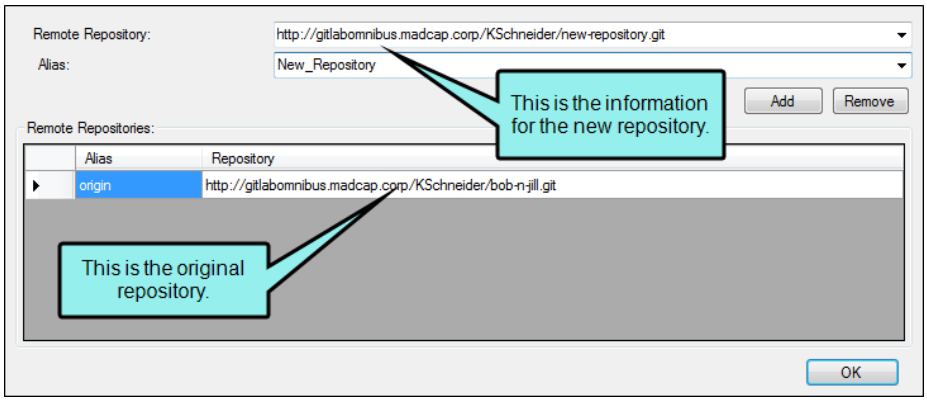
7. (Optional) To add or remove a remote repository, click ▪▪▪. The Remote Repositories dialog opens.

TO ADD A REMOTE REPOSITORY

a. In the **Remote Repository** field, enter the address of the new repository.

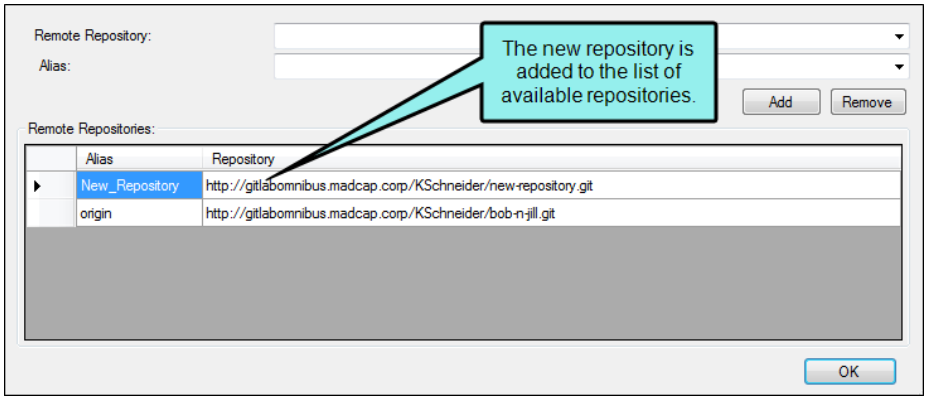
> **NOTE** You may need to obtain this information from your system administrator.

b. In the **Alias** field, enter a name for the new repository. This is the name that will appear in the **Remote** drop-down when you need to select a remote repository.



> **NOTE** Repository names cannot include spaces.

c. Click **Add** to add the new repository to the list of available repositories.



d. Click **OK**.

TO REMOVE A REMOTE REPOSITORY

a. From the **Remote Repository** list, select the repository you want to delete.

b. Click **Remove**. The repository is removed from the list of available repositories.

c. Click **OK**.

8. In the Select Remote for Publish dialog, click **OK**. If the Log In dialog opens, complete the **User name** and **Password** fields and click **OK**. Your branch is published and added to the **Published Branches** list.

Switching Branches

If you have created more than one branch, you can switch to, or select, the branch you want to work with. When you switch to a new branch, you will see the files associated with that branch, and any new commits will be associated with the selected branch. There are multiple ways to switch branches. One of the benefits of using the Branch Management dialog is that you can switch to either local or remote branches.

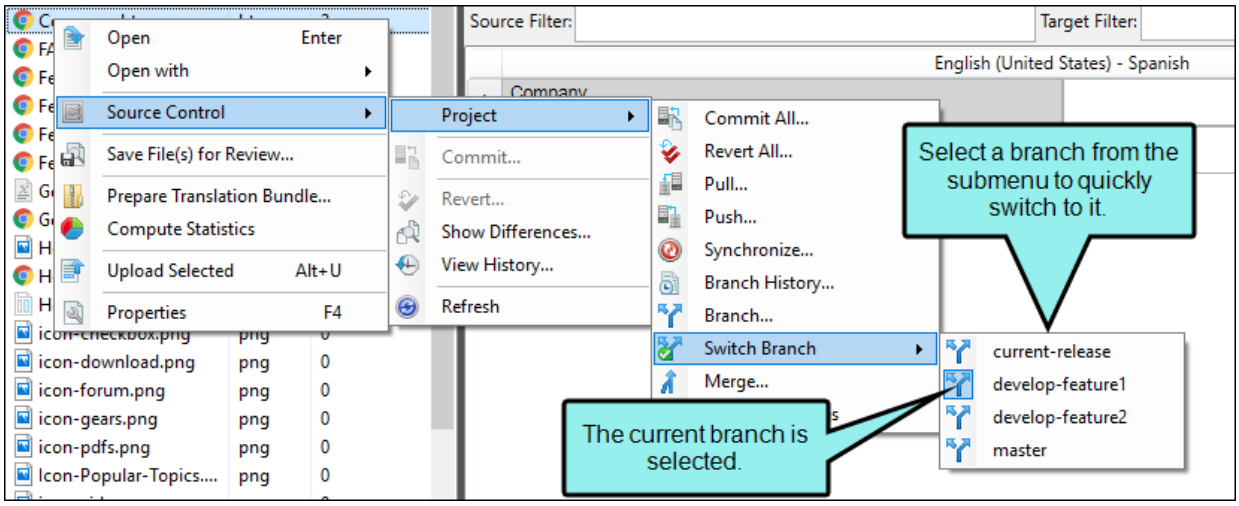
> **WARNING** You should commit all of your current changes before switching branches. If you have uncommitted changes when you switch branches, they will be lost.

How to Switch Branches—Status Bar

1. Make sure the status bar is enabled (View > Status Bar).
2. In the lower-right of the interface, click the branch drop-down, which displays the branch that is currently active. Then, select the branch you want to switch to. If you do not have any pending changes, your branch switches.

How to Switch Branches—Submenu

1. Do one of the following, depending on the part of the user interface you are using:
   - **Ribbon** Select **Source Control > Branch**.
   - **Right-Click** If you have the File List open, right-click on any file and select **Source Control > Project > Switch Branch**.
2. From the submenu, select the branch you want to use. If you do not have any pending changes, your branch switches.



> **NOTE** Your current branch is marked in the submenu.

How to Switch Branches—Branch Management Dialog

1. (Optional) With any branch selected as the active one, do a pull. This makes all remote branches available for selection in the Lingo interface, even if you haven't yet added a remote branch locally.
2. Open the Branch Management dialog in one of the following ways:
   - Status Bar In the lower-right of Lingo, click the name of the active branch.

     > NOTE If you do not see this option, make sure View > Status Bar is enabled.

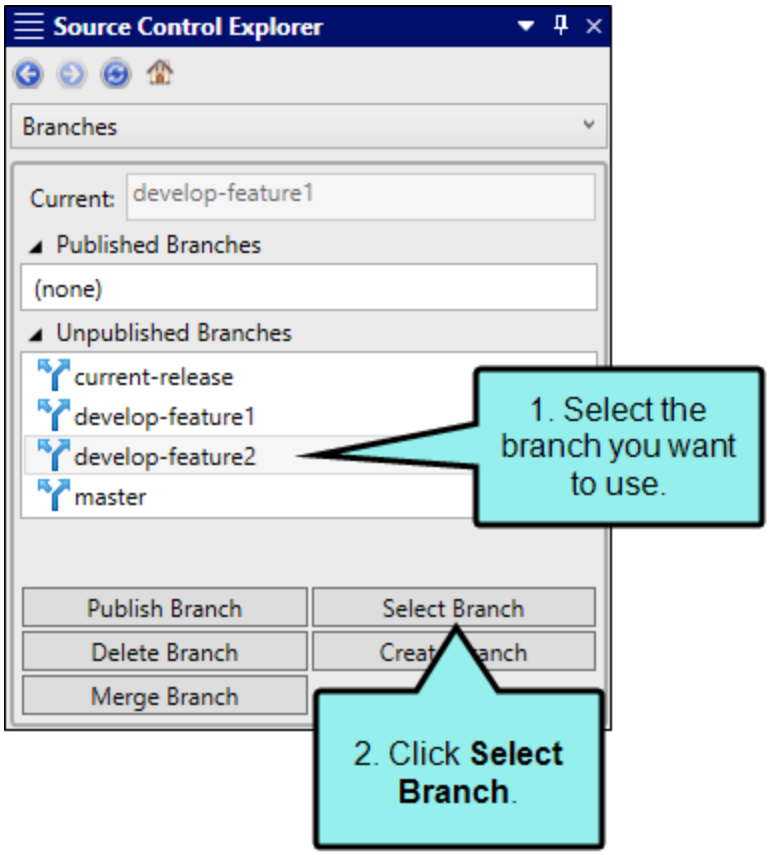
   - Ribbon Select Source Control > Branch (the face of the button, not the drop-down).
   - Right-Click If you have the File List open, right-click on any file and select Source Control > Project > Branch.
3. Select the Locals or Remotes tab, depending on which kind of branch you want to select. Switching to a remote branch will add it to the Locals tab, and make that branch the active one in the Lingo interface.
4. Select the branch you want to get (i.e., check out), and click Switch.
5. Close the Branch Management dialog.

How to Switch Branches—Source Control Explorer

You can also switch branches using the Source Control Explorer. You can view a list of published and unpublished branches in the Source Control Explorer, so this method is preferable if you want to pick a branch by its publication status.

1. Select **View > Source Control Explorer**. The Source Control Explorer opens.
2. From the drop-down or the Home pane, select **Branches**.

   The Branches pane opens. In the pane, you can see your current branch, as well as lists of your published and unpublished branches.
3. Select the branch you want to switch to.
4. Click **Switch Branch**. If you do not have any pending changes, your branch switches.



Switching With Pending Changes

If there are pending changes on your current branch, a dialog asks if you want to switch branches and discard your modifications. This is because the changes were made on the current branch, and do not carry over when you switch branches.

- Click Yes to switch branches. Your changes will be lost.
- Click No to cancel the switch. Commit your changes before attempting the switch again. See "Committing Source Control Files" on page 27.

Getting Remote Branches

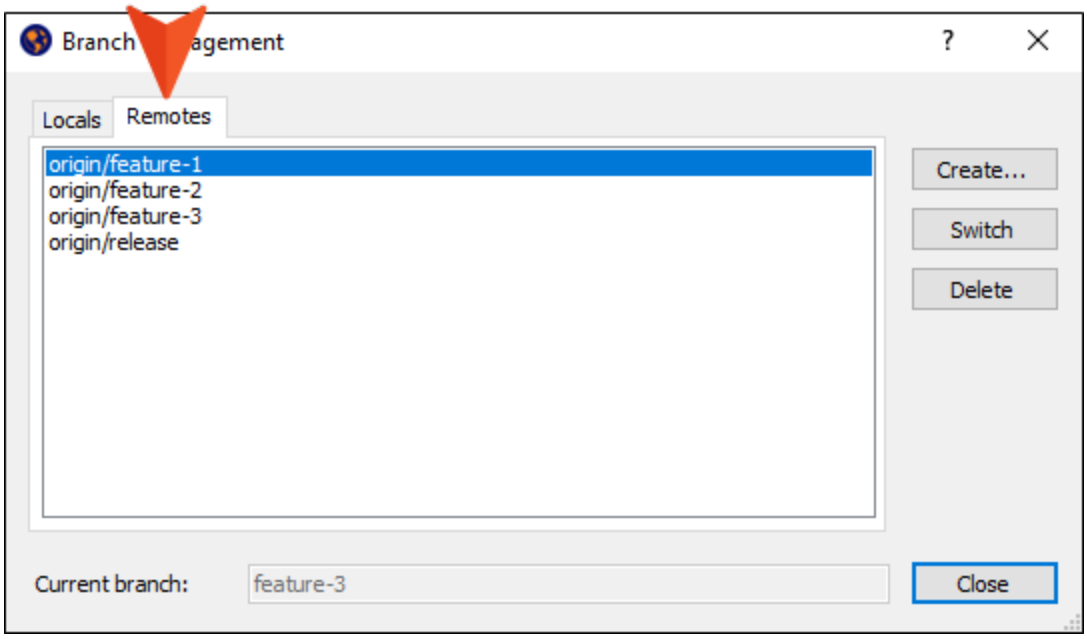
If there is a branch on your remote repository, you might want to add it locally so you can work in it. This can be done from the Branch Management dialog, which lets you see both local and remote Git branches.

How to Get a Remote Branch

1. With any branch selected as the active one, do a pull. This makes all remote branches available for selection in the Lingo interface, even if you haven't yet added a remote branch locally.
2. Open the Branch Management dialog in one of the following ways:
   - **Status Bar** In the lower-right of Lingo, click the name of the active branch.

     > NOTE If you do not see this option, make sure **View > Status Bar** is enabled.

   - **Ribbon** Select **Source Control > Branch** (the face of the button, not the drop-down).
   - **Right-Click** If you have the File List open, right-click on any file and select **Source Control > Project > Branch**.
3. Select the **Remotes** tab.



4. Select the branch you want to get (i.e., check out) from the remote repository, and click **Switch**. This adds the branch to the Locals tab, and it also makes that new local branch the active one in the Lingo interface.
5. Close the Branch Management dialog.

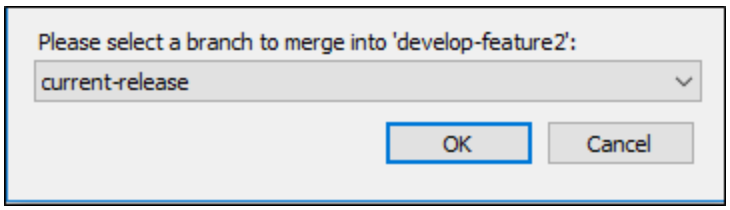
Merging Branches

There are times where you may have to merge branches into a common branch.

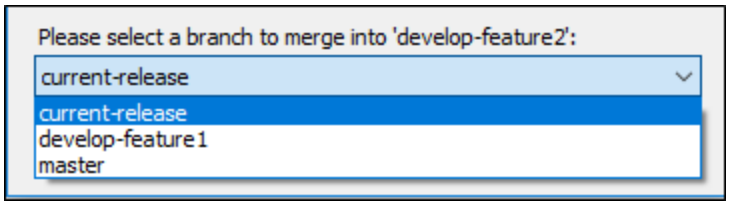
How to Merge Branches in Git

1. Do one of the following, depending on the part of the user interface you are using:
   - **Ribbon** Select **Source Control > Merge**.
   - **Right-Click** If you have the File List window pane open, right-click any file and select **Source Control > Project > Merge**.
   - **Source Control Explorer** From the drop-down, select **Branches**. Then select any branch except the active one, and click the **Merge Branch** button.

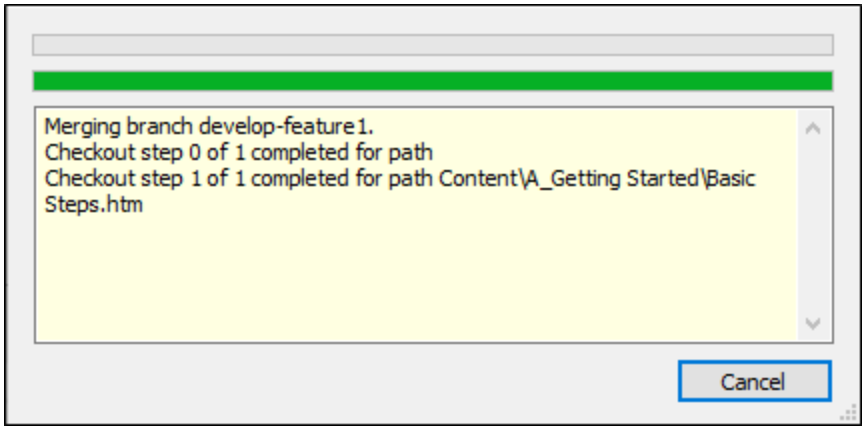
   The Select branch to merge dialog opens.



2. From the drop-down list, select the Git branch you wish to merge into the active branch.



3. Click **OK**. A dialog will display the status of the merge in progress.



4. When the merge operation is completed, a dialog will display indicating the merge was a success. Click **OK** to close the dialog.

☆ EXAMPLE

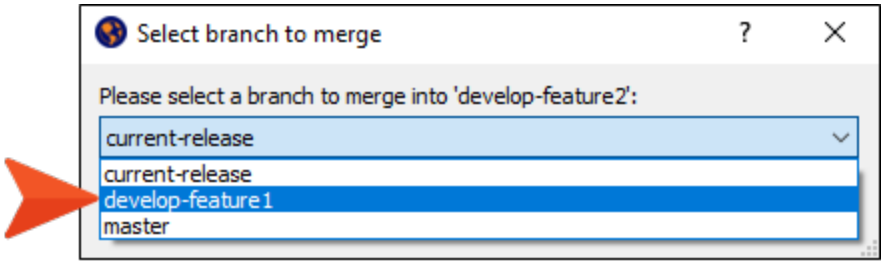
You have two feature branches that are bound in Git source control.

In the first feature branch (develop-feature1), you make changes to a file called “Getting-Started” and commit those changes. These changes are specific to the first feature being developed.

In the second feature branch (e.g., develop-feature2), you make changes to the same "Getting-Started" file and commit those changes. These changes are specific to the second feature being developed, and are being made in a separate part of the topic from the develop-feature1 changes.

After you complete and commit your changes on the second feature branch, you decide you want to merge your changes from the first feature branch to the second feature branch.

In the Source Control Explorer, you select the **develop-feature2** branch. Then, you click the **Merge Branch** button. From the drop-down, you select **develop-feature1**.



You click **OK** to merge. Since the changes took place in different sections of the "Getting-Started" file, the merging of the first feature branch into the second feature branch is successful.

Reverting Branches

A revert is a type of commit that undoes a prior commit on your branch. Reverting becomes necessary if changes have been made that are no longer needed. For example, if you have committed changes for a feature branch that is no longer going to be included in a release, those changes need to be reverted.

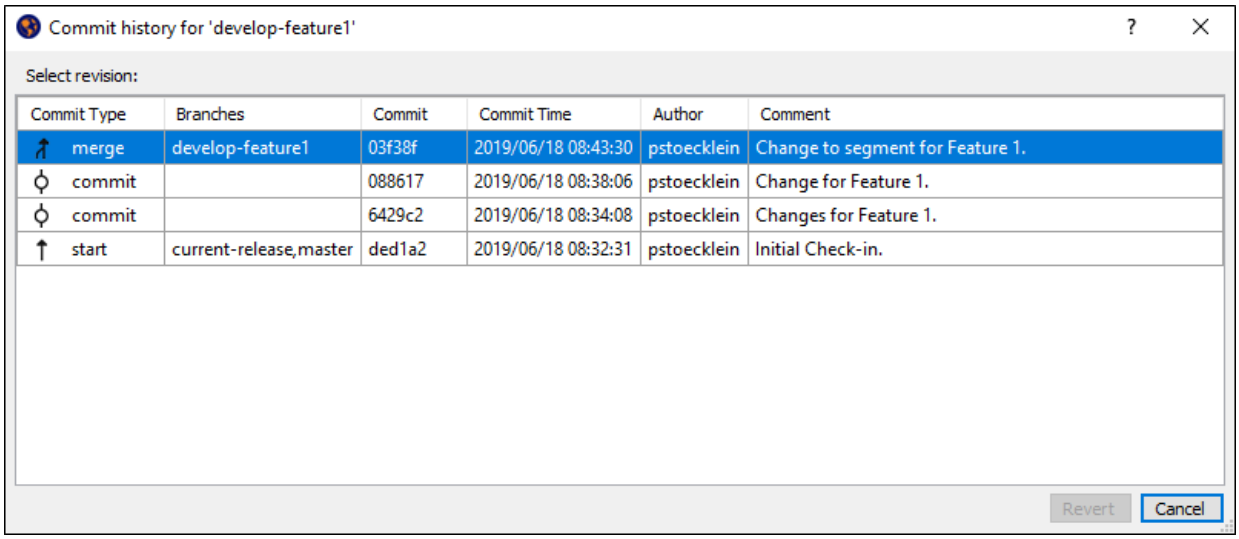
In the Source Control ribbon, the Branch History button displays your history of commits on the active Git branch. This dialog displays the history in descending order, with the most recent action displayed on the top line.

An advantage of reverting a prior commit on your branch is that it will back out any unwanted changes. If other writers are making changes on the same branch, any reverted commits will be picked up by the other writers when they pull from that branch.

But one disadvantage of reverting commits in your branch is that it will remove part of your branch history. The revert removes the part of your branch history that contained the commits that were undone.

How to Revert Commits in a Git Branch

1. In the Source Control Explorer, select your branch containing changes that you want to revert.
2. In the **Source Control** ribbon, click **Branch History**. The dialog that opens lets you view and revert specific commits within your branch.



Commit Type	Branches	Commit	Commit Time	Author	Comment
merge	develop-feature1	03f38f	2019/06/18 08:43:30	pstoecklein	Change to segment for Feature 1.
commit		088617	2019/06/18 08:38:06	pstoecklein	Change for Feature 1.
commit		6429c2	2019/06/18 08:34:08	pstoecklein	Changes for Feature 1.
start	current-release,master	ded1a2	2019/06/18 08:32:31	pstoecklein	Initial Check-in.

The following columns are displayed:

- **Commit Type** This indicates the type of changes checked in on the branch. You can view when a branch was started, when a commit was made, or when a merge occurred.
- **Branches** This column displays the branches that are affected by the action taken. If the changes affect more than one branch then all of the affected branches are displayed in this column.
- **Commit** This alphanumerical code provides the commit number used by Git.
- **Commit Time** This displays the date, followed by the time the commit was made in your Git branch.
- **Author** This displays the name of the author that committed the specific change in the branch.
- **Comments** This column displays the comments made for each commit.

3. Select the row that contains the commit you want to back out.
4. Click **Revert**.The Accept reverted modifications dialog is displayed.

5. Click **Accept**.
6. A confirmation message confirms the revert was successful. Click **OK** to close this dialog.

Deleting Branches

If you no longer need a branch, you can delete it. This also deletes any commits on that branch. The Branch Management dialog lets you delete either local or remote branches, or both.

You can delete any branch except the current branch. If you need to delete the current branch, you will have to switch to a different branch first. See "Switching Branches" on page 46.

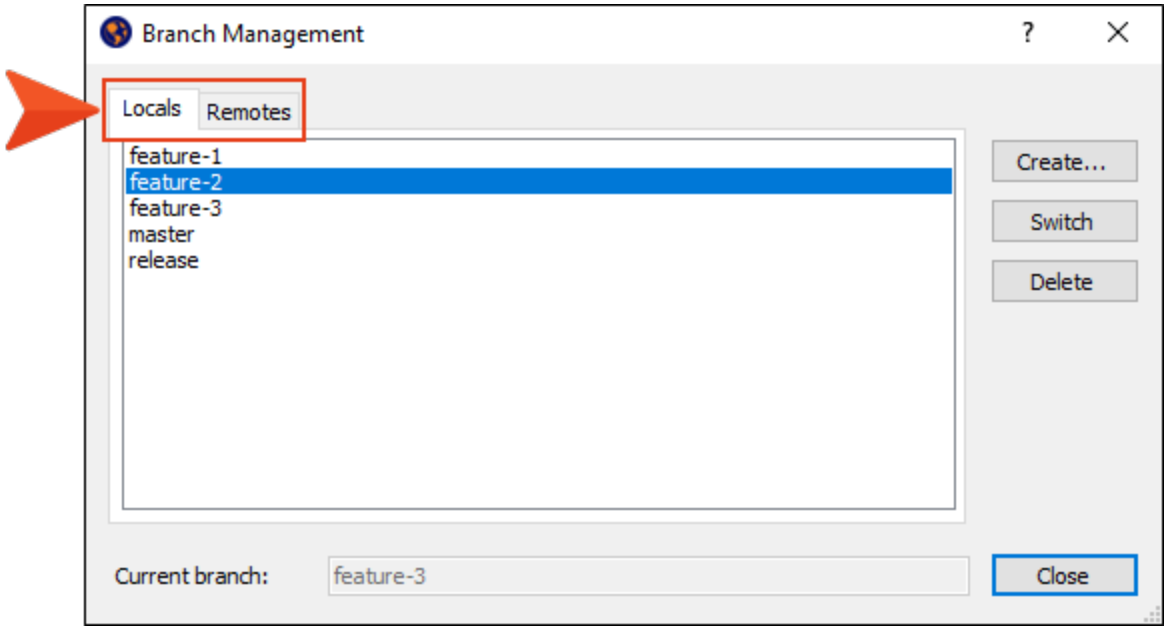
How to Delete Branches—Branch Management Dialog

The benefit of using the Branch Management dialog is that you can choose either local or remote branches to remove.

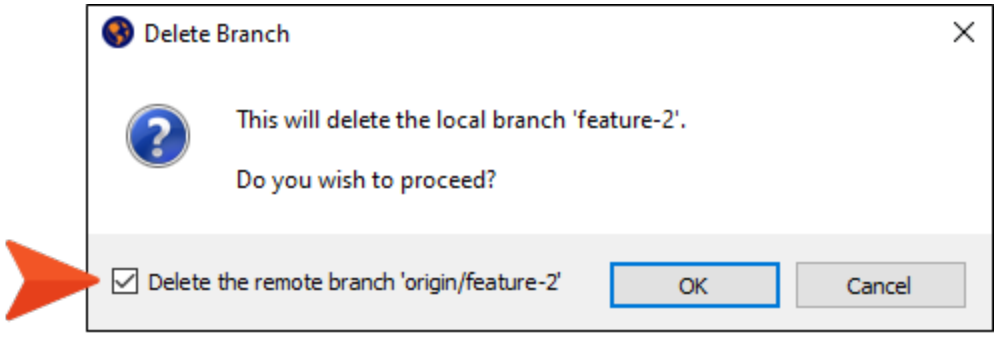
1. Make sure the branch you want to delete is not the active branch.
2. Open the Branch Management dialog in one of the following ways:
   - **Status Bar** In the lower-right of Lingo, click the name of the active branch.

     > NOTE If you do not see this option, make sure **View > Status Bar** is enabled.

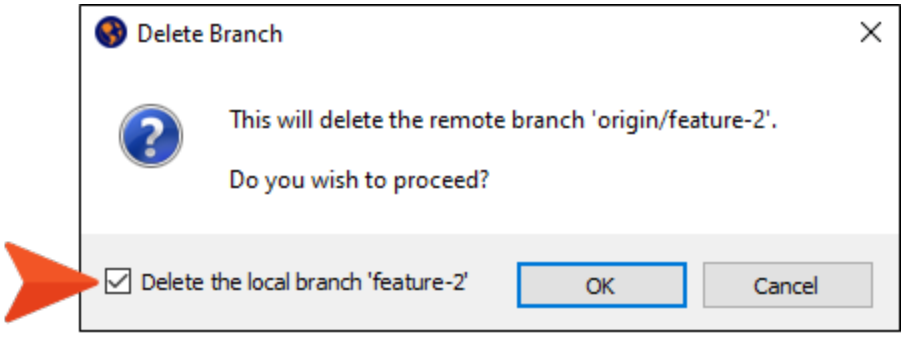
   - **Ribbon** Select **Source Control > Branch** (the face of the button, not the drop-down).
   - **Right-Click** If you have the File List open, right-click on any file and select **Source Control > Project > Branch**.
3. Select the **Locals** or the **Remotes** tab, depending on which branch you want to delete. If you want to delete a branch both locally and remotely, select either tab.



4. Select the branch you want to remove and click **Delete**.
5. If you selected a local branch, you are asked if you also want to delete the remote branch (if one exists). Click the check box if you do; otherwise, leave it disabled.



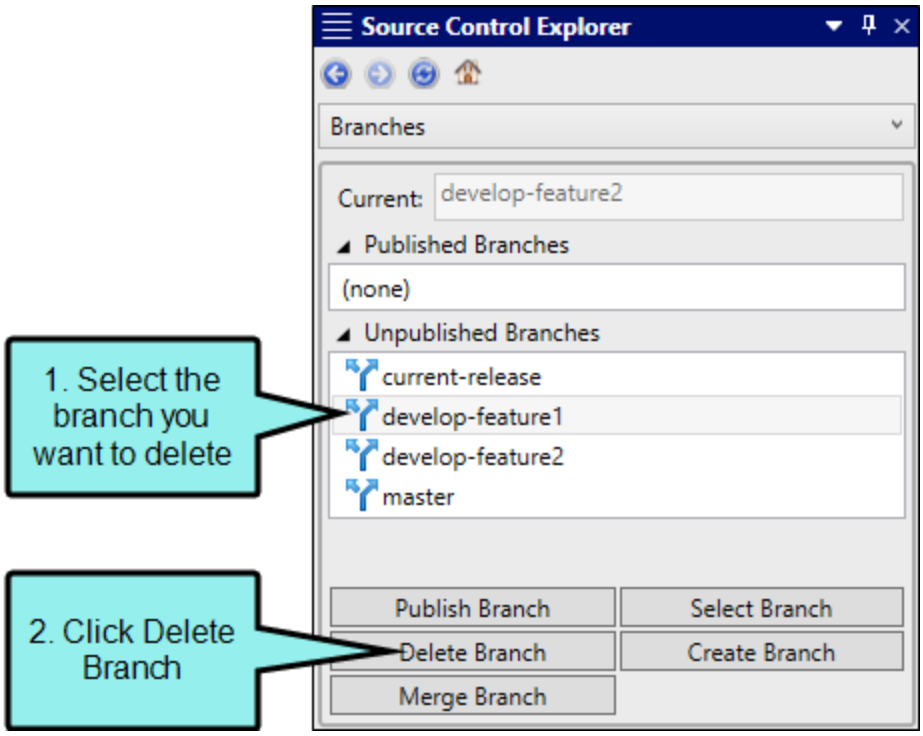
If you selected a remote branch, you are asked if you also want to delete the local branch (if one exists). Click the check box if you do; otherwise, leave it disabled.



6. Click OK.
7. Close the Branch Management dialog.

How to Delete Branches—Source Control Explorer

1. Select **View > Source Control Explorer**. The Source Control Explorer opens.
2. From the drop-down or the Home pane, select **Branches**. The Branches view opens. In the pane, you can see your current branch, as well as lists of your published and unpublished branches.
3. Select the branch you want to delete. You can delete any branch except the current branch.
4. Click **Delete Branch**.



5. A dialog asks if you want to proceed. By clicking the check box in the dialog, you can also delete the remote branch (if one exists).

   Click **OK**. The branch is removed from the list.

CHAPTER 5

Other Activities for Git

In addition to the main activities, there are some other tasks you might perform regarding this feature.

This chapter discusses the following:

Adding and Editing an Ignore File 65
Disabling the Get Latest Prompt for Source Control 69
Disabling a Git Provider 70
Enabling Source Control Status Checks 73
Modifying Network Settings 74
Reverting Modified Source Control Files 77
Rolling Back to an Earlier Version of a File 78
Setting Color Options for Project File Differences 84
Unbinding a Git Provider From a Project 86
Using Git for Windows 90
Viewing Differences in Source Control Files 92
Viewing Modified Files 95
Viewing the History of Source Control Files 97

Adding and Editing an Ignore File

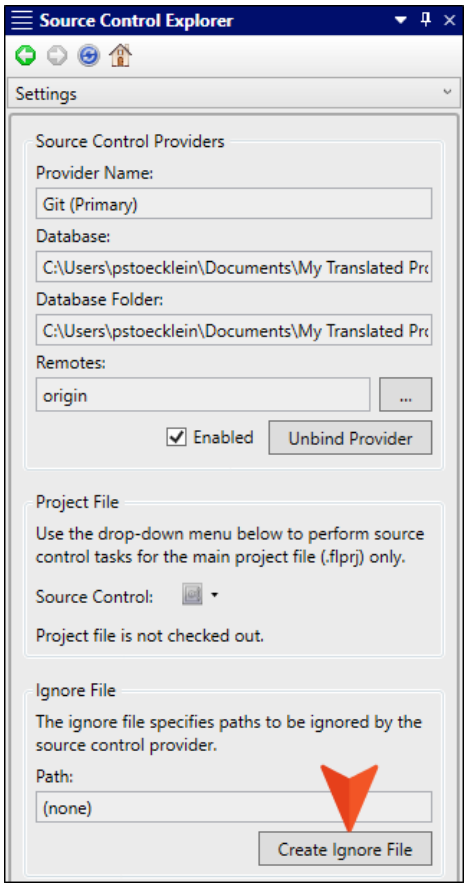
The .git folder is the local repository for a Lingo project and is included when your project is bound to Git. The contents of this folder are updated automatically when you perform source control tasks. The additional .gitignore file is necessary to make sure that certain folders are ignored when you push and pull files. In most cases, you do not need to do anything with the .git folder or .gitignore file.

If you bind a project to Git outside of Lingo (i.e., you do not use Lingo's interface to do the binding), you should make sure that you have a .gitignore file. You can add the .gitignore file by selecting an option in the Source Control Explorer or Project Properties dialog. Once you have a project containing a .gitignore file, the button in the interface changes to "Edit Ignore File," so that you can open the file to make edits to it.

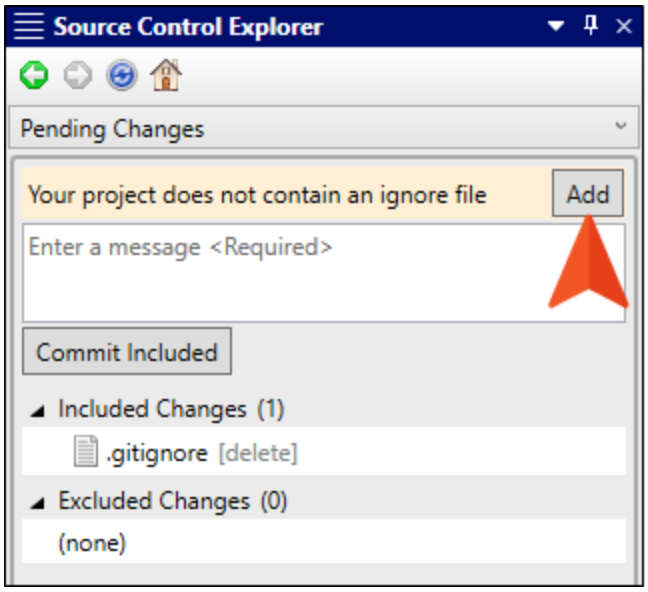
How to Add an Ignore File

The following steps show how to do this in the Source Control Explorer. You can also accomplish this in the Project Properties dialog (**File > Project Properties**) on the **Source Control** tab.

1. Select **View > Source Control Explorer**.
2. Do one of the following:
   - Click **Settings**. Then in the **Ignore File** section, **Create Ignore File**.



- Click **Pending Changes**. Then next to the message at the top, click **Add**.

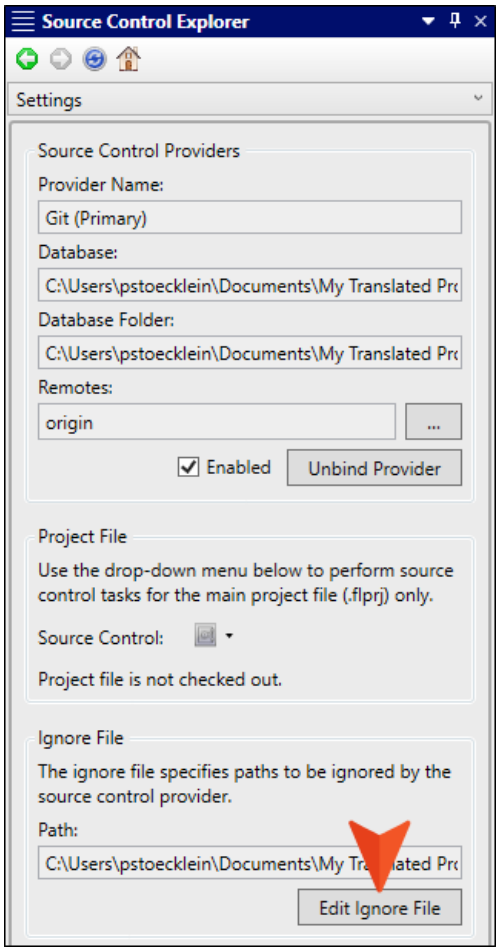


3. Make edits in the Text Editor and save your changes.

How to Edit an Ignore File

The following steps show how to do this in the Source Control Explorer. You can also accomplish this in the Project Properties dialog (**File > Project Properties**) on the **Source Control** tab.

1. Select **View > Source Control Explorer**.
2. Click **Settings**.
3. In the **Ignore File** section, click **Edit Ignore File**.



4. Make edits in the Text Editor and save your changes.

Disabling the Get Latest Prompt for Source Control

By default, when you open a project that is bound to source control, a message automatically asks if you want to get the latest version of files. However, you can disable this prompt in the Source Control tab of the Options dialog (**File > Options**). Therefore, in the future when you open the project you will no longer see the message, and the project will open without replacing any local files with the latest ones from source control.

How to Disable the Get Latest Prompt for Source Control

1. Select **File > Options**. The Options dialog opens.
2. Select the **Source Control** tab.

   NOTE This tab will not be visible if your project is not yet bound to source control. See "Binding a Project to Git" on page 13.

3. Click the check box **Do not prompt to get latest when opening source control bound projects**.
4. Click **OK**.

Disabling a Git Provider

By default, when a project is bound to source control, the provider (Git, Perforce Helix Core, Subversion, or Team Foundation Server) is enabled. This means that the source control interface elements in Lingo are visible, and you can use them to perform various tasks (e.g., commits, synchronize changes).

Disabling a provider means that the source control interface elements are no longer shown. This does not mean you cannot use source control. As long as the provider is still *bound* to the project, you can perform source control tasks in a third-party tool outside of Lingo.

How to Disable a Provider for All Projects Globally

This is the easiest method if you have multiple projects and you want to disable the Git provider in Lingo for all of them.

1. Select File > Options.
2. Select the Source Control tab.
3. In the Bind Detection section, remove the check mark next to Git.
4. Click OK.
5. Close the project and then reopen it so that the option can take effect.

   The provider option that is found in the Project Properties dialog or the Settings view in the Source Control Explorer (i.e., the "Enabled" check box) will then automatically be disabled.

How to Disable a Provider in an Individual Project

Use this method if you want to disable a provider in just one project, rather than many projects.

1. Do one of the following, depending on the part of the user interface you are using:
   - **Project Properties** Select **File > Project Properties**.
   - **Source Control Explorer** Select **View > Source Control Explorer**. Then, in the window pane, click **Settings**.
2. Click **Enabled** to remove the check mark.
3. If you used the Project Properties dialog, click **OK**.

What's Noteworthy?

NOTE If you disable a Git provider, the local repository will continue to track your changes in case you later decide to enable the provider once again.

If you disable one of the other providers (Perforce Helix Core, Subversion, Team Foundation Server), your changes after that point will not be tracked. Therefore, if you later enable the provider again, it will not have recorded any changes made since the time that you disabled it.

NOTE When you disable a provider, that information is written to the registry on your computer.

NOTE If you disable a provider, but then perform one of the following actions in the Lingo interface, the provider will automatically become enabled once again.

- Bind an existing project
- Bind a new project
- Import a project from source control

NOTE Having a provider enabled in Lingo does not interfere with your workflow if you are performing source control actions exclusively outside of Lingo. Even if a provider is enabled in the project and the source control user interface elements are visible, this does not mean Lingo is automatically performing any source control actions with your files. It simply means Lingo is recognizing the binding, so it reflects your activities (e.g., the Pending Changes window is populated when you make edits in topics). However, if you prefer not to see any of this in Lingo, you can disable the provider.

Enabling Source Control Status Checks

If you are using source control integration in Lingo, you can check for frequent status changes automatically. You can specify the number of minutes and seconds when you want Lingo to ping the source control repository and get status changes for files that have been committed, synchronized, moved, deleted, etc. The upside of this feature is that you can ensure that the source control status information is always up to date. The downside is that you may experience slower performance due to this constant communication over the network.

How to Enable Source Control Status Checks

1. Select **File > Options**. The Options dialog opens..
2. Select the **Source Control** tab.

   NOTE This tab will not be visible if your project is not yet bound to source control. See "Binding a Project to Git" on page 13.

3. Click the check box **Enable background status checks**. A check mark in the box indicates that the feature is enabled.
4. Enter the number of minutes and or seconds between each status update.
5. Click **OK**.

NOTE If you elect to disable this feature (disabled is the default setting), you can manually check for status updates by refreshing the Pending Changes window pane or Source Control Explorer. See "Viewing Modified Files" on page 95.

Modifying Network Settings

You can view and change various source control network settings while working in Lingo.

How to Modify Network Settings

1. Do one of the following, depending on the part of the user interface you are using:
   - Ribbon Select Source Control > Network Settings.
   - Right-Click If you have the File List open, right-click on any file and select Source Control > Project > Network Settings.
   - Local Toolbar In the local toolbar of the File List, click , then select Project > Network Settings.
   - Source Control Explorer From the View ribbon, open the Source Control Explorer. From the drop-down, select the Home Page view. Click Network Settings.

   The Network Settings dialog opens.
2. In the Group field, select the group for which you want to change the settings.

   OR

   Do one of the following:
   - (Optional) If you want to add a custom group, type its name in the Group field, then click Add.
   - (Optional) If you want to remove a group, select it from the Group field, then click Remove.
3. (Optional) If you entered a custom group name, in the Remote Host field, enter the name of the domain to which the network settings should apply.

4. In the grid, make changes to the network settings as necessary. Click to sort the settings by category, or click to sort them alphabetically.

GIT NETWORK SETTINGS

- Cookies
  - Cookie File
  - Save Cookies
- HTTP Proxy Options
  - Proxy Host
- Identity
  - User Agent
- Performance
  - Enable EPSV
  - Max Requests
  - Maximum High Latency Time
  - Min Sessions
  - Minimum Transfer Speed
  - POST Buffer Maximum
- SSL Configuration
  - SSL CA File
  - SSL CA Path
  - SSL Certificate
  - SSL Key File
  - SSL Password Prompt
  - Try SSL
  - Verify SSL

> **NOTE** For more information about each of these settings, refer to:
>
> `https://git-scm.com/docs/git-config`

5. Click **Save**.

Reverting Modified Source Control Files

If you have modified files from source control but do not want to keep your modifications, use the "Revert" option instead of committing the files. While committing the file would save changes to source control, reverting a file returns it to its previously committed state and does not commit any of your new changes to source control. When reverting changes made in Git, you only revert changes to the file on the branch you are currently editing. If you have a file that resides on multiple branches, copies of the file on other branches are preserved.

How to Revert a Source Control File

1. In the Source Control Explorer or File List, select the relevant file(s).
2. Do one of the following, depending on the part of the user interface you are using:
   - Ribbon Select Source Control > Revert (for selected files) or Source Control > Revert All (for all files in the project).
   - Right-Click In the File List, right-click the files you want to revert and select Source Control > Revert (for selected files) or Source Control > Project > Revert All (for all files).
   - Local Toolbar In the File List, select the file(s) you want to revert. In the local toolbar of the File List, click , then select Revert (for selected files) or Project > Revert All (for all files in the project).

   A dialog opens. The selected files are listed with check boxes next to them.
3. (Optional) If you want to see all files with pending changes (rather than only those you selected), click .
4. Make sure to click the check box next to each appropriate file so that it contains a check mark.
5. Click Revert.

Rolling Back to an Earlier Version of a File

One of the benefits of Lingo's integrated source control is that you can view the history and differences for a particular file. You can view code and content differences between two source control versions of the same file. This is useful if you need to roll back to an earlier version of a file.

See "Viewing the History of Source Control Files" on page 97 and "Viewing Differences in Source Control Files" on page 92.

☆ EXAMPLE You have been working on translating a particular topic for a few days. Each day you pull the remote commits to your local database, make your changes, and commit and push the file back to the remote repository at the end of the day. At a certain point, you determine that you need to "roll back" to an earlier version of the file, using it to replace the latest version. Therefore, you use this feature to view the highlighted differences between the current version and an older version of the file. Once you have identified the older version that you want to use, you can retrieve that version.

How to Roll Back to an Earlier Version of a File

1. In the Source Control Explorer or File List, select the relevant file(s).

   OR

   Open a file.

2. Do one of the following, depending on the part of the user interface you are using:

   - Ribbon Select Source Control > View History.
   - Right-Click If you have the File List open, right-click the file you want to roll back and select Source Control > View History.
   - Source Control Explorer With the Pending Changes pane open, right-click the file you want to roll back and select View History.

   The History dialog opens.

3. (Optional) View the differences between two versions of the file. This may help you decide which version of the file you want to retrieve. (Another way is to look at the dates for each version in the History dialog.)

   To view the differences, follow these steps.

   a. Select the first file version from the list.

   b. Hold the CTRL key and select the second file version from the list.

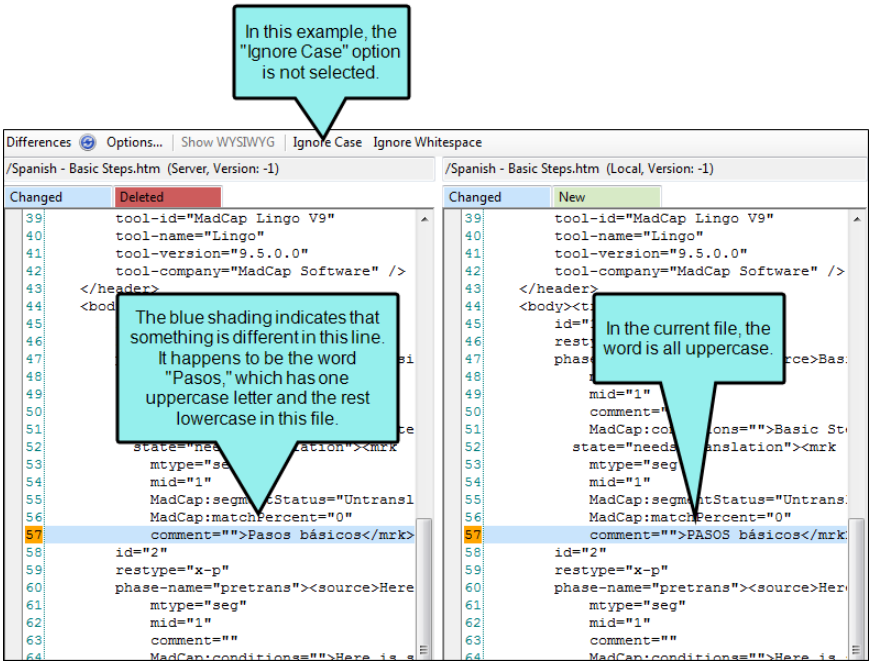
   c. Click Show Differences. The Differences Editor opens to the right, showing content from the backup file on the left and the current version of the file on the right.

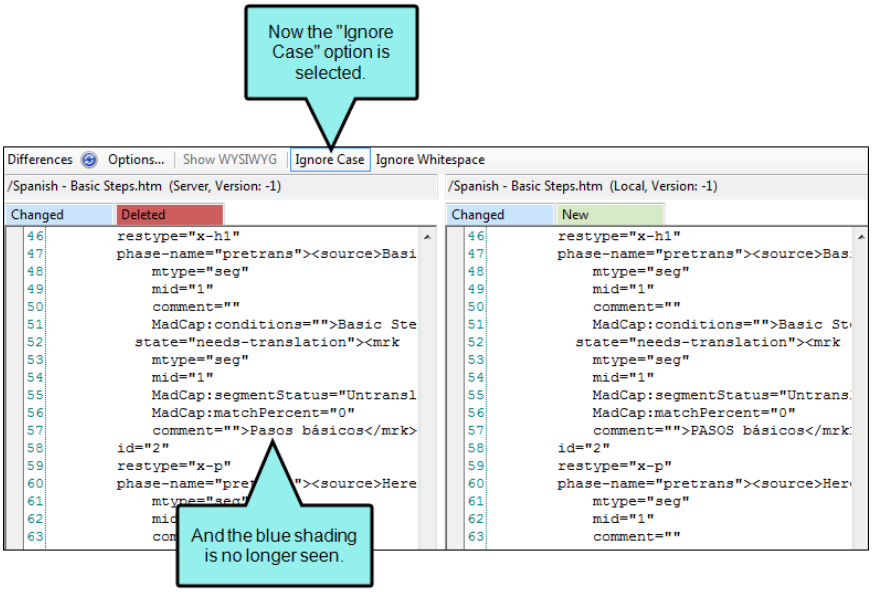
   d. In the local toolbar of the Differences Editor, you can click any of the following to make adjustments.

      - Options Click this to open the File Differences dialog, which lets you change the colors used to display content differences between the files.

- **Ignore Case** Click this to ignore changes in case when viewing differences.

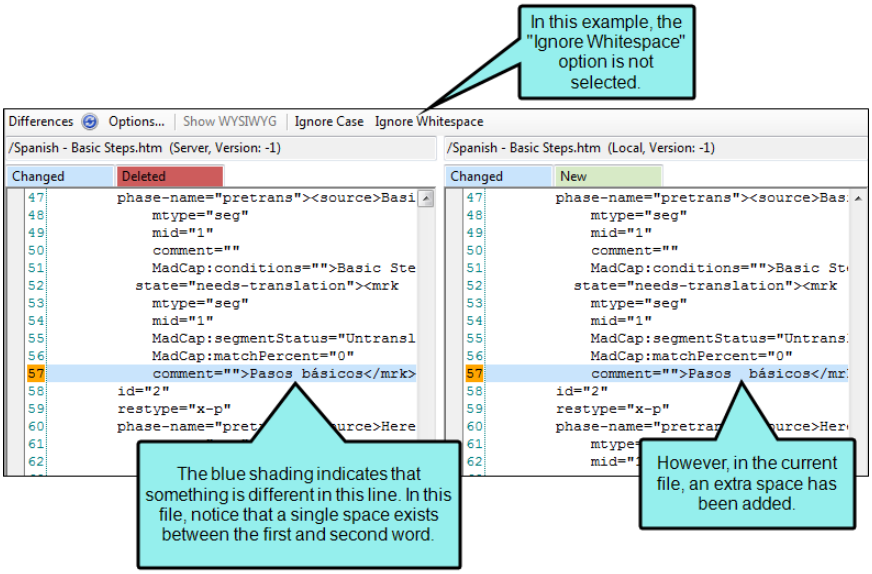
> ☆ EXAMPLE If a word is not capitalized in the original file but it is capitalized in the current file, this option does not highlight those differences.




Now the "Ignore Case" option is selected.
Differences
Options...
Show WYSIWYG
Ignore Case
Ignore Whitespace
/Spanish - Basic Steps.htm (Server, Version: -1)
/Spanish - Basic Steps.htm (Local, Version: -1)
Changed
Deleted
Changed
New
And the blue shading is no longer seen.

- **Ignore Whitespace** Click this to ignore whitespace when viewing differences.

☆ EXAMPLE A segment is identical in both files, except for an extra space that was added within the segment in one of those files. If you click this option, that difference is not highlighted.



e. When you are finished viewing the differences, close the window pane.

4. In the History dialog, select the version of the file to which you want to roll back.
5. Click **Get Selected Version**. That file is retrieved from the server and replaces the local copy of the file in your project.
6. In the History dialog, click **Close**.

Setting Color Options for Project File Differences

If you are using Lingo's integrated source control features, you can view differences between files in various ways. One way is to view file differences between a local version of a Lingo project and the source control version.

When viewing file differences between a local version of a Lingo project and the source control version, you can select color options to display the files. Color coding makes it easier to discern where differences between files occur.

For more information see "Viewing Differences in Source Control Files" on page 92.

☆ EXAMPLE By default the files that are included only in your local copy are displayed as green in the Differences Editor, and the files that are included only in source control are displayed in red. You can use this dialog to change the local-only files to blue and the source control–only files to yellow.

How to Set Color Options for Project File Differences

1. In the Source Control Explorer or File List, select the relevant file(s).

   OR

   Open a file.

2. Do one of the following, depending on the part of the user interface you are using:

   - Ribbon Select Source Control > Show Differences.
   - Right-Click If you have the Source Control Explorer or File List open, right-click the file you want to view and select Source Control > Show Differences.
   - Local Toolbar In the File List, select the file(s) you want to view. In the local toolbar of the File List, click , then select Show Differences.

   The Differences Editor opens.

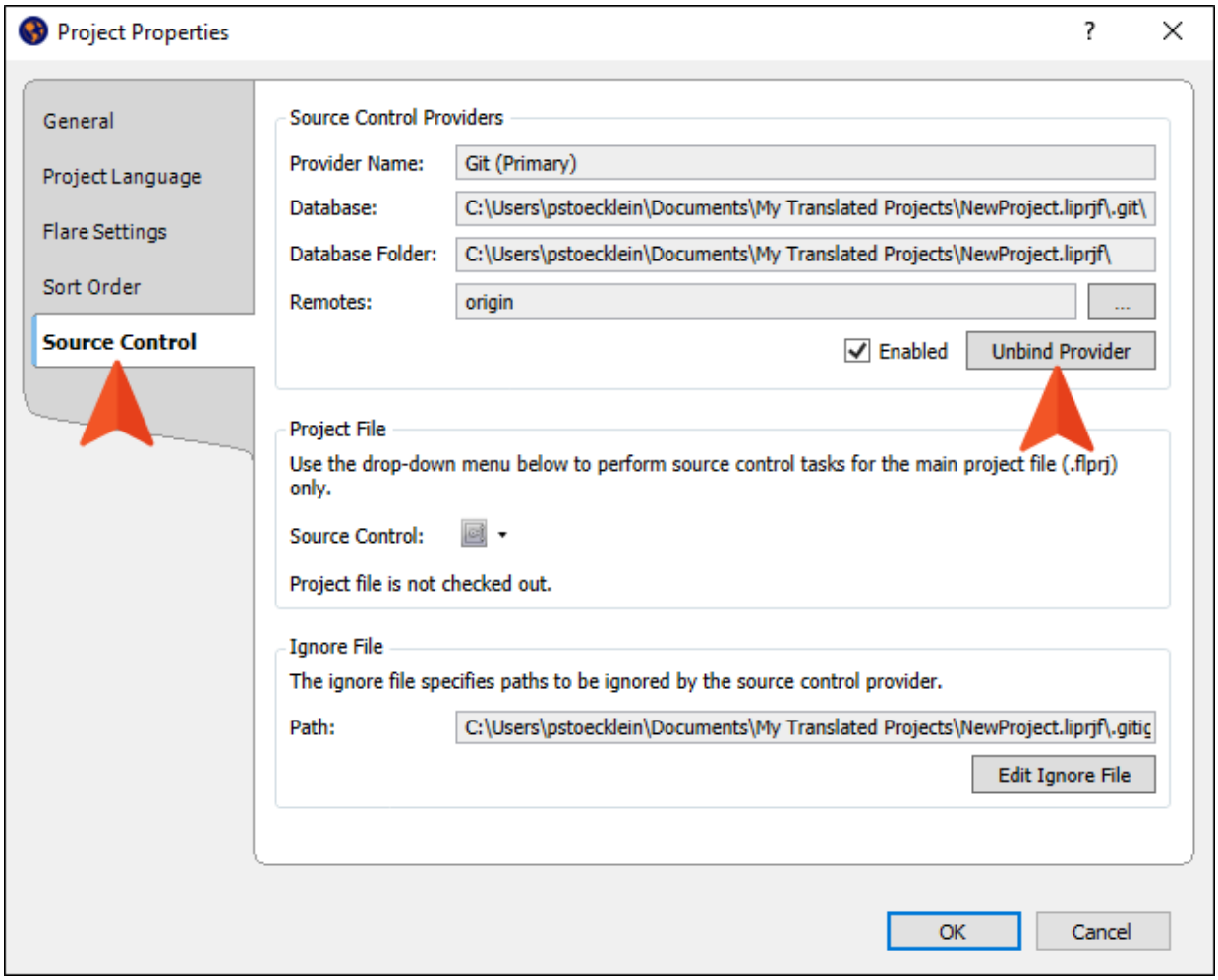
3. In the local toolbar of the Differences Editor, click Options. The File Differences dialog opens.
4. Change the text or background color for any of the difference types. To do this, simply click in the appropriate Text or Background cell and select Pick Color. In the Color Picker dialog, choose the new color.
5. Click OK.

Unbinding a Git Provider From a Project

When you unbind a provider, it means you are removing the connection altogether between the Lingo project and the local repository.

How to Unbind a Provider in the Project Properties

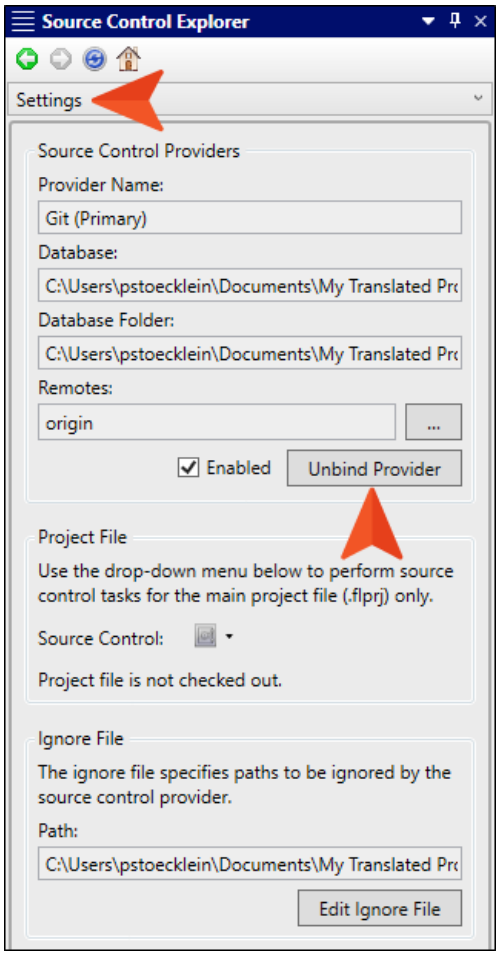
1. Open the project.
2. Select **File > Project Properties**.
3. Select the **Source Control** tab.
4. Click **Unbind Provider**.



5. Click **OK**.

How to Unbind a Provider in the Source Control Explorer

1. Open the project.
2. Select **View > Source Control Explorer**.
3. From the drop-down or the Home pane, select **Settings**. The Settings pane opens.
4. Click **Unbind Provider**. (If your project is dual-bound, you will also see a section for the other binding.)



5. In the confirmation dialog, click **Yes**.

What’s Noteworthy?

NOTE You can also disable a provider, which retains the binding but hides source control elements from the user interface.

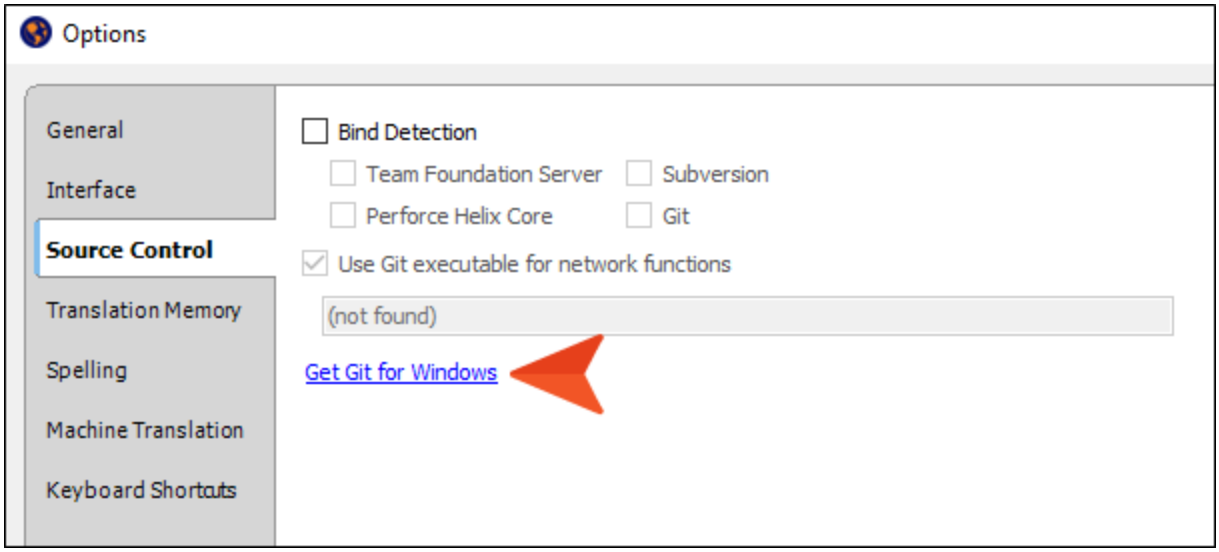
Using Git for Windows

In Windows 10 version 1903, Microsoft implemented OpenSSH, which is the open-source version of Secure Shell (SSH). As a result, a project bound to Git using SSH as the transfer protocol might experience error messages (e.g., failed to start SSH session). The solution is to download and install Git for Windows (Git.exe).

An upside to installing Git.exe is that it is likely to result in faster and better performance for file transfers.

How to Download and Install Git.exe

1. Select **File > Options**.
2. Select the **Source Control** tab.
3. Click **Get Git for Windows**. A page opens on your browser, and the download should begin automatically. If it doesn't, click the link on the web page to download it.

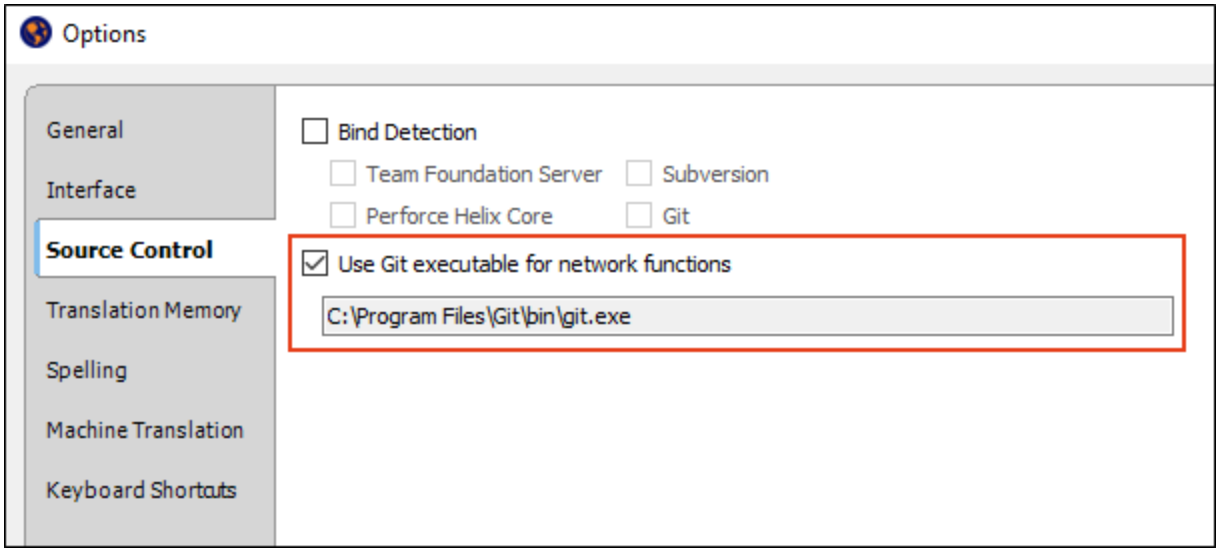


> **NOTE** If you already have Git.exe installed, but not the latest version, this link will instead display "Get an updated version of Git for Windows."

4. When the download finishes, open it.
5. Follow the steps in the wizard. We recommend you keep the default selections.

6. After you click **Install** and the installation completes, click **Finish**.

   After a few moments, the link in the Options dialog should go away and you should see this instead:



7. Click **OK**.

Viewing Differences in Source Control Files

One of the benefits of Lingo's integrated source control is that you can view the history and differences for a particular file.

Ways to View Differences Between Files

You can view differences between files in the following ways:

- Two Versions of Same Source Control File (History/Roll Back) You can view code and content differences between two source control versions of the same file. This is useful if you need to roll back to an earlier version of a file.

  > ☆ EXAMPLE You have been working on translating a particular topic for a few days. Each day you pull the remote commits to your local database, make your changes, and commit and push the file back to the remote repository at the end of the day. At a certain point, you determine that you need to "roll back" to an earlier version of the file, using it to replace the latest version. Therefore, you use this feature to view the highlighted differences between the current version and an older version of the file. Once you have identified the older version that you want to use, you can retrieve that version.

- Local Versus Source Control Version of a File You can view code and content differences between the local version of a file and the source control version of that file.

  > ☆ EXAMPLE You open a topic from source control and then make changes to some of the translations in your local copy of that file. You save your changes. Later that day, you want to revisit the new translations, but you cannot remember exactly which translations you added and which were there before. Therefore, you use this feature to highlight the text differences between your local version of the file and the version stored in the source control application. The new text is highlighted on the side displaying the local version of the file.

How to View Differences Between Two Versions of the Same Source Control File

1. In the Source Control Explorer or File List, select the relevant file(s).
2. Do one of the following, depending on the part of the user interface you are using:
   - Ribbon Select Source Control > View History.
   - Right-Click If you have the Source Control Explorer or File List open, right-click the file (s) you want to view and select Source Control > View History.
   - Local Toolbar In the File List, select the file(s) you want to view. In the local toolbar of the File List, click , then select View History .

   The History dialog opens.
3. From the list, select the first file version that you want to compare.
4. Hold the CTRL key and select the second file version from the list.
5. Select Show Differences. The Differences Editor opens.
6. (Optional) In the Differences Editor, use the buttons in the local toolbar to customize the information shown in the editor.
7. When you are finished viewing the differences, close the window.

How to View Differences Between the Local and Source Control Versions of a File

1. In the Source Control Explorer or File List, select the relevant file(s).
2. Do one of the following, depending on the part of the user interface you are using:
   - **Ribbon** Select **Source Control > Show Differences**.
   - **Right-Click** If you have the Source Control Explorer or File List open, right-click the file (s) you want to view and select **Source Control > Show Differences**.
   - **Local Toolbar** In the File List, select the file(s) you want to view. In the local toolbar of the File List, click , then select **Show Differences**.

   The Differences Editor opens.
3. (Optional) In the Differences Editor, use the buttons in the local toolbar to customize the information shown in the editor.
4. When you are finished viewing the differences, close the window.

What's Noteworthy?

> **NOTE** When you modify a file in source control, you are actually modifying the file's corresponding XLF file. You will see the XLF file if you open the Source Control Explorer. This is because you need the XLF file available in order to make changes, view the file's history, or view differences.

Viewing Modified Files

You can use the Source Control Explorer to view all of the files that you have modified and need to commit.

How to View Modified Files—Source Control Explorer

1. Select **View > Source Control Explorer**.

   The Source Control Explorer opens.

2. From the drop-down or the Home pane, select **Pending Changes**.

   The Pending Changes pane opens. Files that you have changed appear in the **Included Changes** or **Excluded Changes** section (depending on whether you are going to include or exclude them in your next commit; see "Committing Source Control Files" on page 27). You will not see other users' changes in the Source Control Explorer.

3. Take note of the file's status. The status is written in brackets next to the file name (e.g., edit, add).

What’s Noteworthy?

NOTE You can click the refresh button in the local toolbar to make sure you have the most recent status for each file. Another option is that you can use a feature to automatically ping the source control repository periodically, thus refreshing this information frequently. However, you may experience slower performance with this automatic status update option set. See "Enabling Source Control Status Checks" on page 73.

NOTE When you modify a file in source control, you are actually modifying the file's corresponding XLF file. You will see the XLF file if you open the Source Control Explorer. This is because you need the XLF file available in order to make changes, view the file's history, or view differences.

NOTE When you modify a file in source control, you may sometimes see a SKL file alongside the XLF and original files. This skeleton file is a placeholder file. Be sure to commit all three associated files together. Committing just the SKL file may result in errors.

Viewing the History of Source Control Files

One of the benefits of Lingo's integrated source control is that you can view the history for a particular file, including who committed the file and when it was committed. You can also view differences between different versions of the file and roll back to an older version if necessary.

For more information see "Viewing Differences in Source Control Files" on page 92 and "Rolling Back to an Earlier Version of a File" on page 78.

How to View the History of a Source Control File

1. In the Source Control Explorer or File List, select the relevant file(s).

   OR

   Open a file.

2. Do one of the following, depending on the part of the user interface you are using:
   - Ribbon Select Source Control > View History.
   - Right-Click If you have the Source Control Explorer or File List open, right-click the file you want to view and select Source Control > View History.
   - Local Toolbar In the File List, select the file(s) you want to view. In the local toolbar of the File List, click , then select View History .

3. The History dialog opens. Following are explanations of the different parts of this dialog.
   - Version Displays a number for each version of the file. The lower the number, the older the version. The higher the number, the more recent the version.
   - Users Displays the name of the user who has been working on the file.
   - Date Displays the date and time when the action has occurred.
   - Action Displays the action that has taken place for the file (e.g., modified).
   - Comment Displays the comment (if any) associated with the file. A comment can be added to a file when you commit that file to source control. This enables you to maintain an audit trail for the file's history.

- **Get Selected Version** Retrieves a particular version of a file, thus rolling back to that version of the file. The local version of the file is replaced with the source control version that you selected.
- **Show Differences** Opens a dialog where you can view the differences between two versions of a file. If you select one row in the History dialog and view the differences, you will see the content differences between the version that you selected and the version of the file in your local copy of the Lingo project. If you select two files in the History dialog (by holding down the CTRL key) and view the differences, you will see the content differences between those two versions of the file.

4. In the History dialog, click **Close**.

> **NOTE** When you modify a file in source control, you are actually modifying the file's corresponding XLF file. You will see the XLF file if you open the Source Control Explorer. This is because you need the XLF file available in order to make changes, view the file's history, or view differences.

PDFs

The following PDFs are available for download from the Help system.

Cheat Sheets

Shortcuts Cheat Sheet

User Guides

Alignment Guide

Getting Started Guide

Key Features Guide

Source Control Guide: Git

Source Control Guide: Perforce Helix Core

Source Control Guide: Subversion

Source Control Guide: Team Foundation Server

Termbases Guide

Touring the Workspace Guide

Translation Guide

What's New Guide