USER GUIDE

MADCAP FLARE ONLINE

Source Control Guide

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

When you upload a project to Flare Online, the files are connected to Flare Online via an integrated source control system (Git). Your interaction with source control can follow one of two models—single-bound (recommended) or dual-bound. Git is the source control solution that works behind the scenes to integrate Flare Desktop with Flare Online.

NOTE If an author needs to work with the project in Flare Desktop after it is created in Flare Online—because advanced features are needed for the project—the user needs to (1) have access to the project, and (2) open Flare Desktop and import the project from Flare Online. If additional changes are made in Flare Desktop or Flare Online, the work would need to be synchronized between the local and remote repositories.

CHAPTER 2

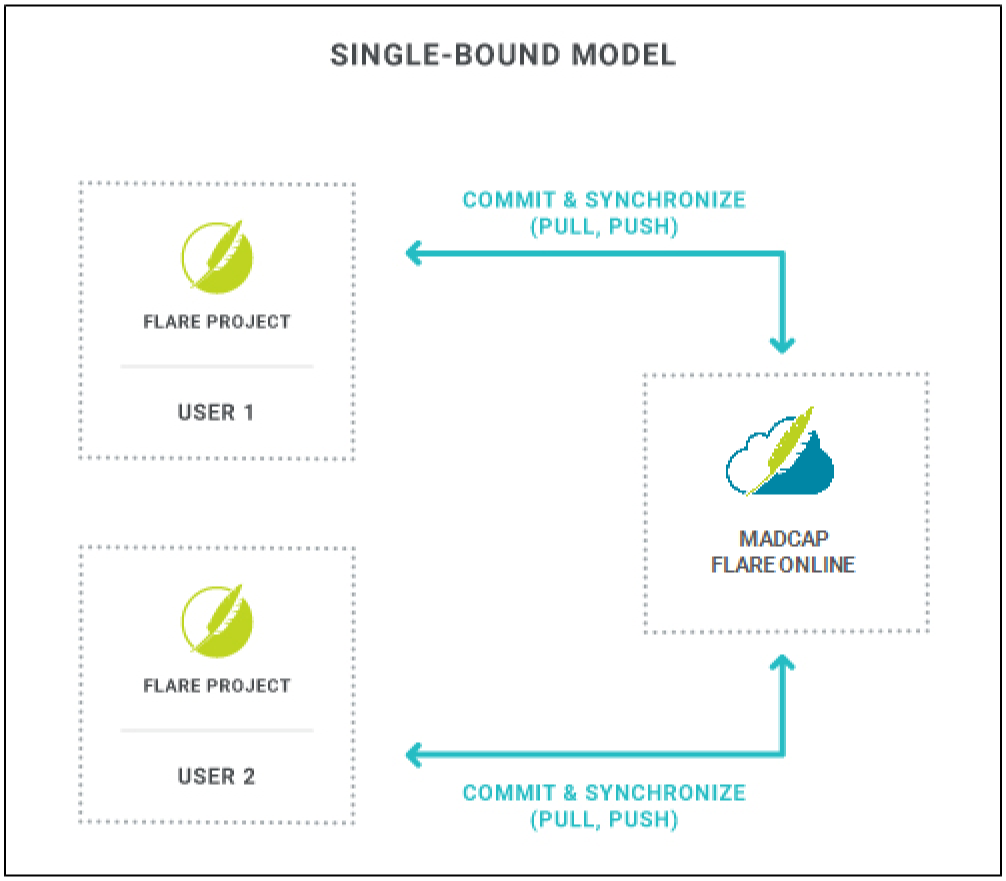
Binding Models

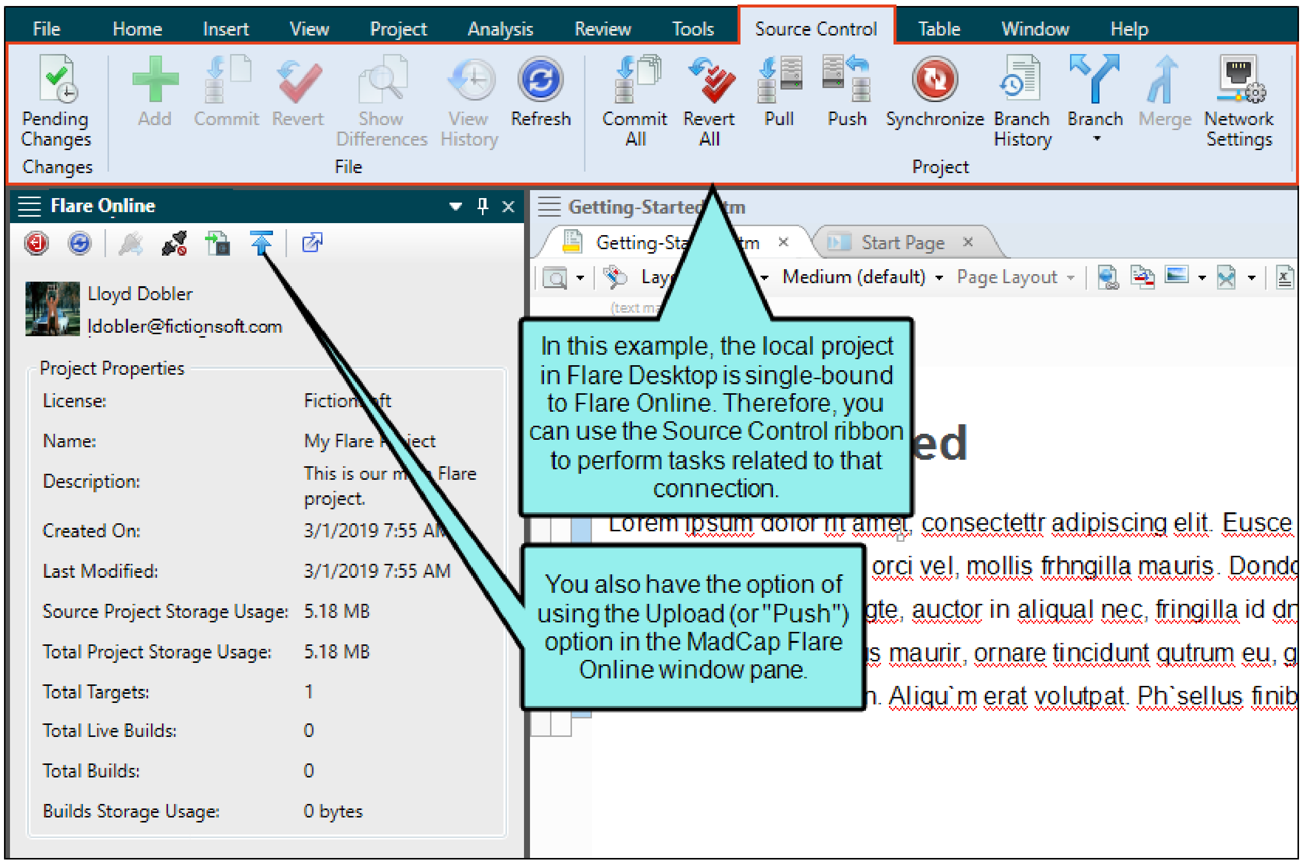
You have the option of more than one binding model when using Flare Online and Flare Desktop.

This chapter discusses the following:

Single-Bound Projects

If your local project in Flare Desktop is not already bound to a third-party source control tool and you upload the project to Flare Online, you will have only one source control binding. Therefore, you can use the source control connection between your local project repository and the remote Flare Online repository as your primary source control solution. With this model, you can use the Source Control ribbon in Flare Desktop to perform regular source control tasks such as committing and synchronizing (i.e., pulling and pushing) files. However, you also have the option of using tools outside of Flare Desktop (e.g., Git Bash) to perform source control tasks (e.g., in case there are certain source control tasks not supported in the Flare Desktop interface that you need to complete).





Lloyd Dobler
ldobler@fictionsoft.com

Project Properties

License:	FictionSoft
Name:	My Flare Project
Description:	This is our main Flare project.
Created On:	3/1/2019 7:55 AM
Last Modified:	3/1/2019 7:55 AM
Source Project Storage Usage:	5.18 MB
Total Project Storage Usage:	5.18 MB
Total Targets:	1
Total Live Builds:	0
Total Builds:	0
Builds Storage Usage:	0 bytes

In this example, the local project in Flare Desktop is single-bound to Flare Online. Therefore, you can use the Source Control ribbon to perform tasks related to that connection.

You also have the option of using the Upload (or "Push") option in the MadCap Flare Online window pane.

How to Use a Single Binding

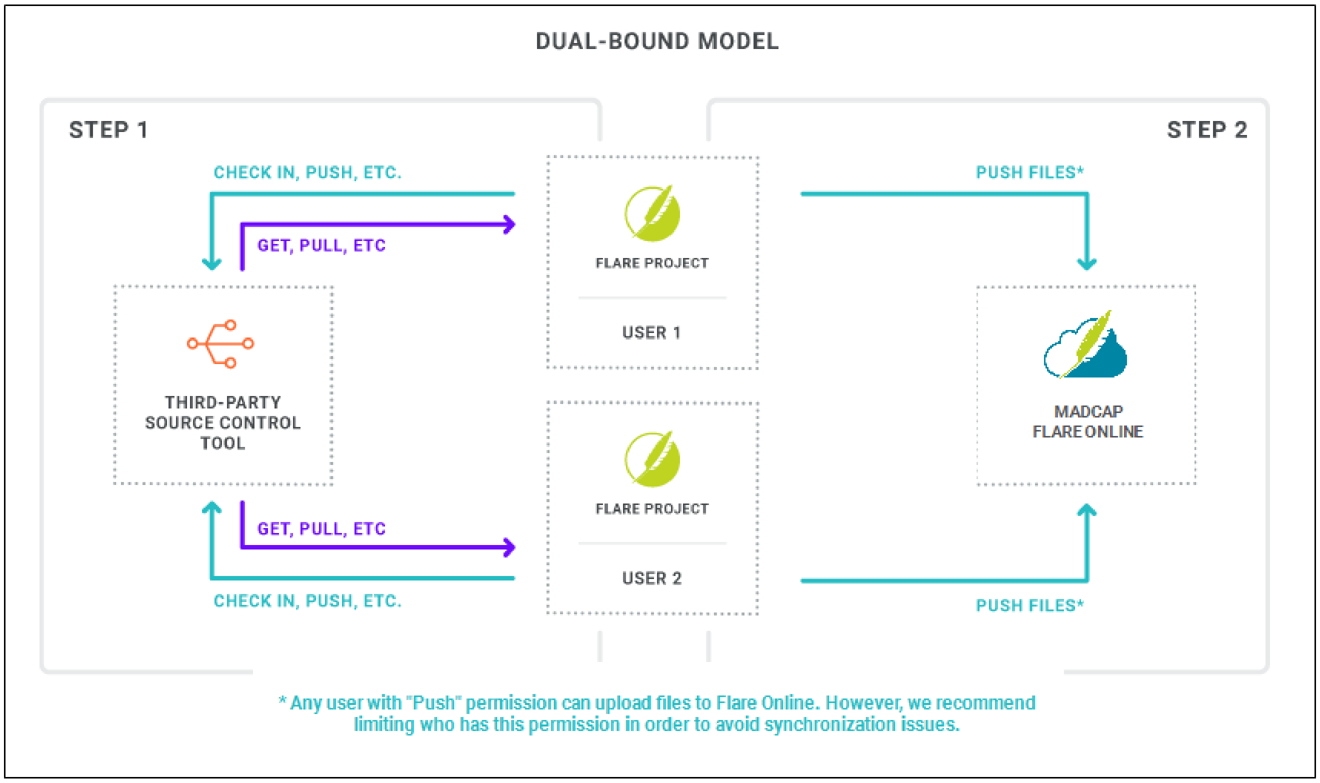
1. Open the project in MadCap Flare Desktop.
2. Select **View > Flare Online**. The Flare Online window pane opens.
3. Be sure to log in if you aren't already.
4. Upload (bind) your project to Flare Online.
5. In Flare Online, assign users to the new project and make sure they have “Import/Pull” and “Push” permissions. This enables users to import the project using the Flare Online window pane. It also allows users to push changes up to Flare Online.
6. Newly assigned users must now import the project from Flare Online using the Flare Online window pane in Flare Desktop. This allows each user to work on the project locally.
7. In Flare Desktop, make changes to your files. Then commit those files and synchronize them (i.e., pull, then push) with the cloned project in Flare Online. See "Committing and Synchronizing in a Single-Bound Model" on page 30.

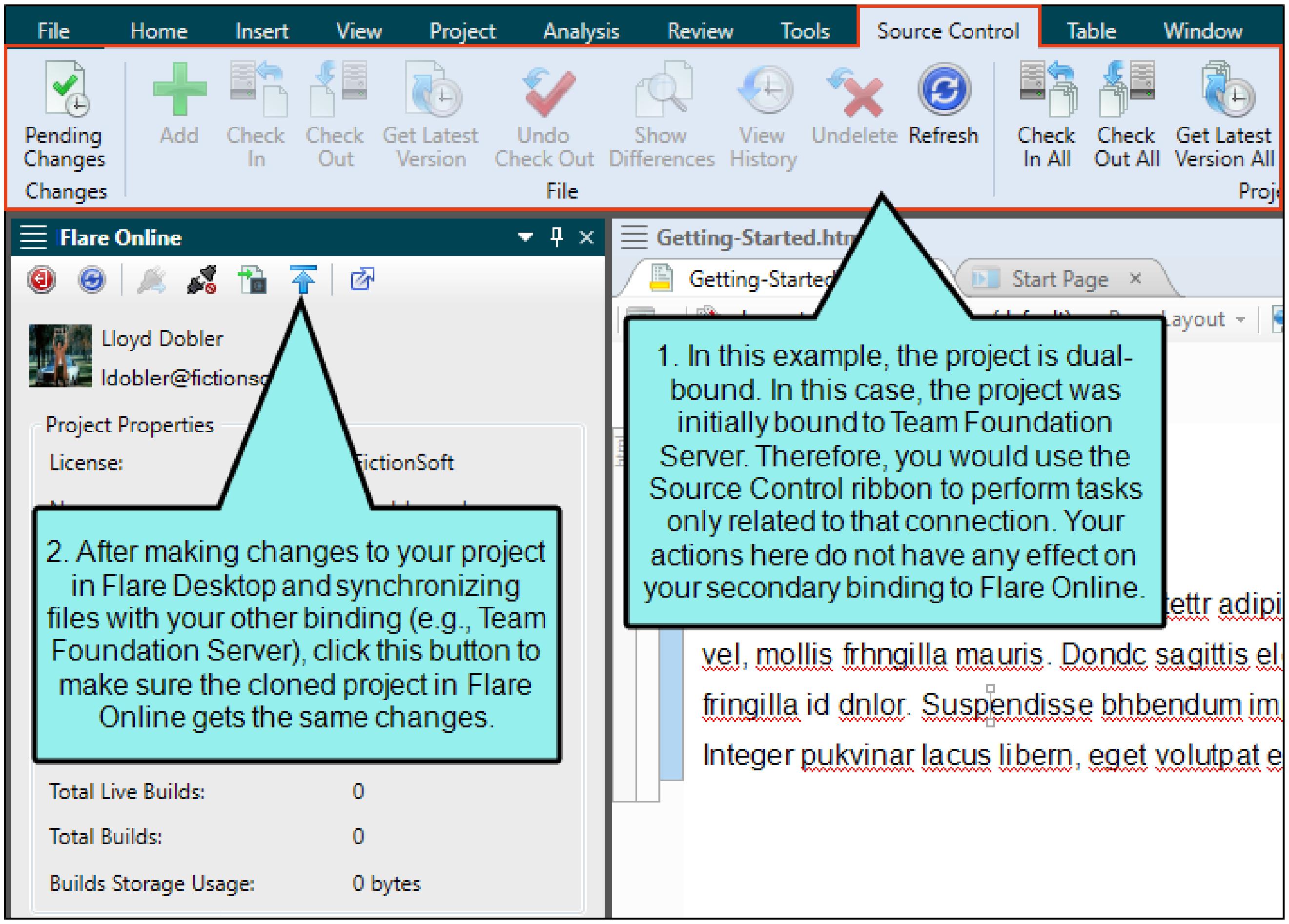
NOTE If your team is using Git branching, you can create branches in Flare Desktop and push them up to Flare Online. Other writers on your team who need to work in the same branches can pull them down from Flare Online. For more details and steps on creating and managing branches, see the Flare Desktop Help system.

Dual-Bound Projects

A dual-bound project has a first binding to a third-party source control provider—such as Git, Perforce Helix Core, Subversion, or Team Foundation Server—and a second binding to Flare Online. Some might choose this model because it allows them to use a source control provider they're comfortable with, while also taking advantage of features in Flare Online.

However, if possible, it's recommended that you use the single-bound setup since it's simpler and more streamlined. With the dual-bound model, you use the first source control binding to do most of the version control work; the Source Control ribbon in Flare Desktop is used only for tasks related to the first binding. Then periodically, you will use the Flare Online window pane in Flare Desktop to upload (or "push") the latest files from your local copy of the project up to Flare Online via the second binding; the Source Control ribbon in Flare Desktop is not used for the second binding. However, keep in mind that creating and synchronizing branches (other than master) between Flare Desktop and Flare Online is supported only with a single-bound project or with a Git-Flare Online dual-bound setup.


DUAL-BOUND MODEL
STEP 1
STEP 2
CHECK IN, PUSH, ETC.
GET, PULL, ETC
FLARE PROJECT
USER 1
PUSH FILES*
THIRD-PARTY SOURCE CONTROL TOOL
MADCAP FLARE ONLINE
FLARE PROJECT
USER 2
GET, PULL, ETC
CHECK IN, PUSH, ETC.
PUSH FILES*
* Any user with "Push" permission can upload files to Flare Online. However, we recommend limiting who has this permission in order to avoid synchronization issues.



1. In this example, the project is dual-bound. In this case, the project was initially bound to Team Foundation Server. Therefore, you would use the Source Control ribbon to perform tasks only related to that connection. Your actions here do not have any effect on your secondary binding to Flare Online.

2. After making changes to your project in Flare Desktop and synchronizing files with your other binding (e.g., Team Foundation Server), click this button to make sure the cloned project in Flare Online gets the same changes.

NOTE It is possible for multiple people working on a dual-bound project to push files to Flare Online. However, if you are using a source control provider other than Git for the first binding, the most recently pushed files are the ones that are used in Flare Online. In other words, the last person to push "wins." To avoid issues, be sure that you have the most recent version of file changes from all other writers in your local project before you push. You may even want to limit users' permissions so only one or two people are allowed to push files to Flare Online.

NOTE Your first source control binding (between Flare Desktop and a third-party provider) should be done from the Flare Desktop interface, rather than from another tool. Otherwise, Flare Desktop and Flare Online will not recognize that binding. If you already have a project that was bound using another tool, you have a couple of options:

- You can remove the binding and then bind again using Flare Desktop.
- You can create a new project, importing from source control. This method allows you to retain the repository.

How to Use a Dual Binding—Perforce, Subversion, and TFS

NOTE Flare Online supports branches from Git only. Therefore, if you want to use branching in your project and work with those branches in Flare Online, we recommend that you use a single-bound setup instead, where Git will be operating behind the scenes.

1. Open the project in MadCap Flare Desktop, and make sure your project is already bound to a third-party source control provider.
2. Select **View > Flare Online**. The Flare Online window pane opens.
3. Be sure to log in if you aren't already.
4. Upload (bind) your project to Flare Online.

   Because you are dual-bound, Flare Desktop will prompt you to check out your project file before you can upload the project.
5. Check in the project file to your third-party source control provider.
6. In Flare Online, assign users to the new project and make sure they have “Push” permissions. This gives the users the ability to push local content to the project in Flare Online.

7. Newly assigned users must now do one of the following, depending on whether or not they already have a copy of the project on their local machine:

   - **Have a Local Copy** Get the latest version of the project file from the third-party source control provider.
   - **Do Not Have a Local Copy** Import the project from the third-party source control provider using the Import Project from Source Control Wizard.

     NOTE If your project is already bound to a third-party source control provider other than Git—i.e., you are working in a dual-bound model as opposed to a single-bound model—the Import option in the Flare Online window pane will be disabled. You will have to import your project directly from source control using your non-Flare Online binding.

     If you already have the project on your machine, you do not need to reimport the project from source control. Instead, you can get the latest version from source control by using the Flare Online Source Control ribbon.

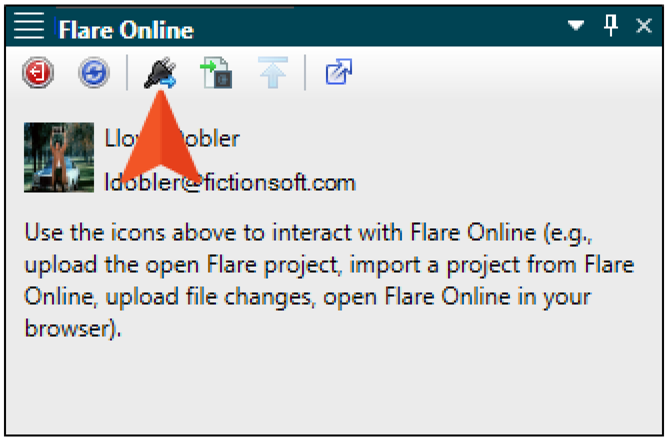
     Be sure that you have the latest version of the project.

   Either option will give users the most current version of the project.

8. In Flare Desktop, make changes to your files. You should manage all of your file changes using your third-party source control provider, using the following actions:
   - **Check In** Check in your changes to source control.
   - **Get Latest** Get the latest version of your teammates' changes from source control and add them to your local project.
9. When you are done making changes, push your final changes to Flare Online. To do this, click in the Flare Online window pane. See "Pushing in a Dual-Bound Model" on page 36.

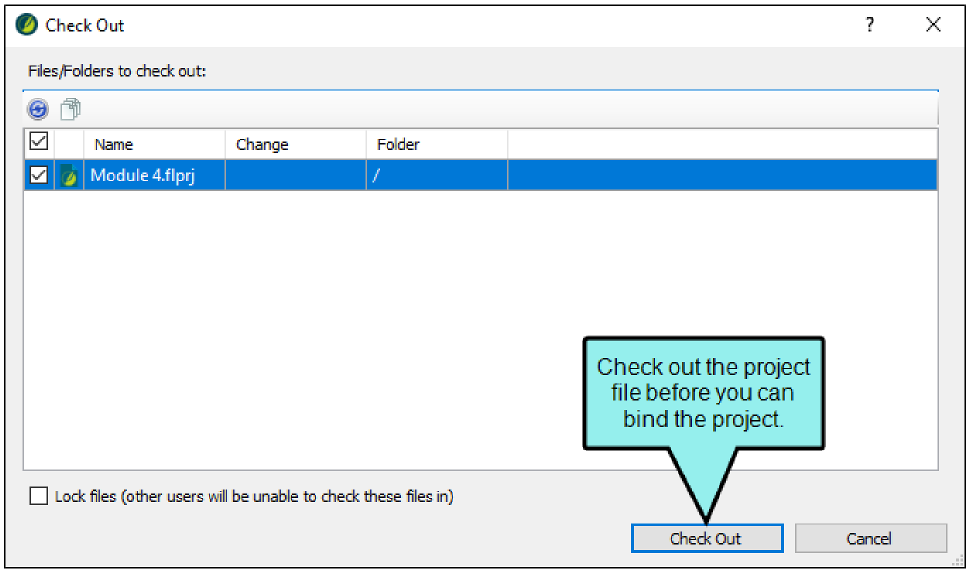
☆ EXAMPLE You are working on a team of writers and your project is bound to Microsoft Team Foundation Server (TFS). Therefore, that is your primary source control provider, and the source control connection between Flare Desktop and Flare Online serves as a secondary source control provider.

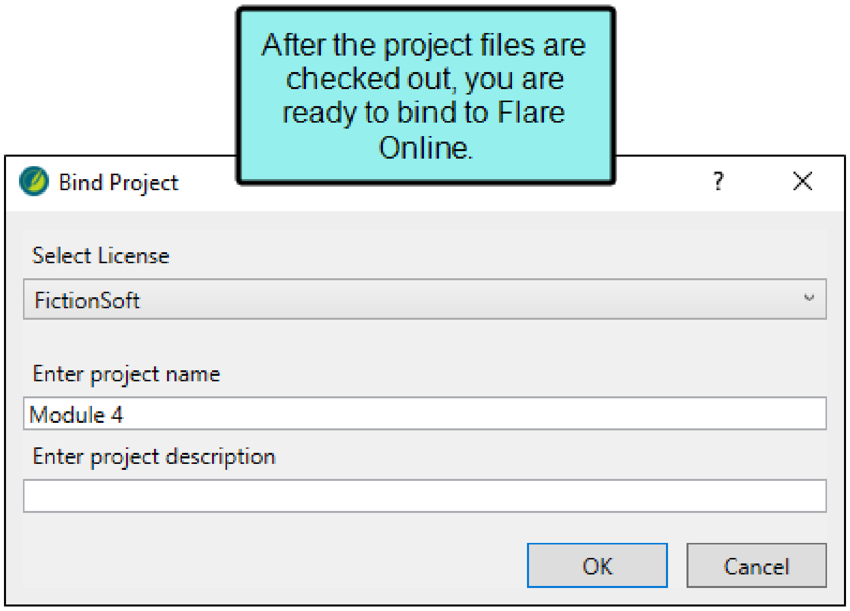
When you first start working with Flare Online, you have to upload (or bind) the project to Flare Online. To do this, click in the Flare Online window pane.

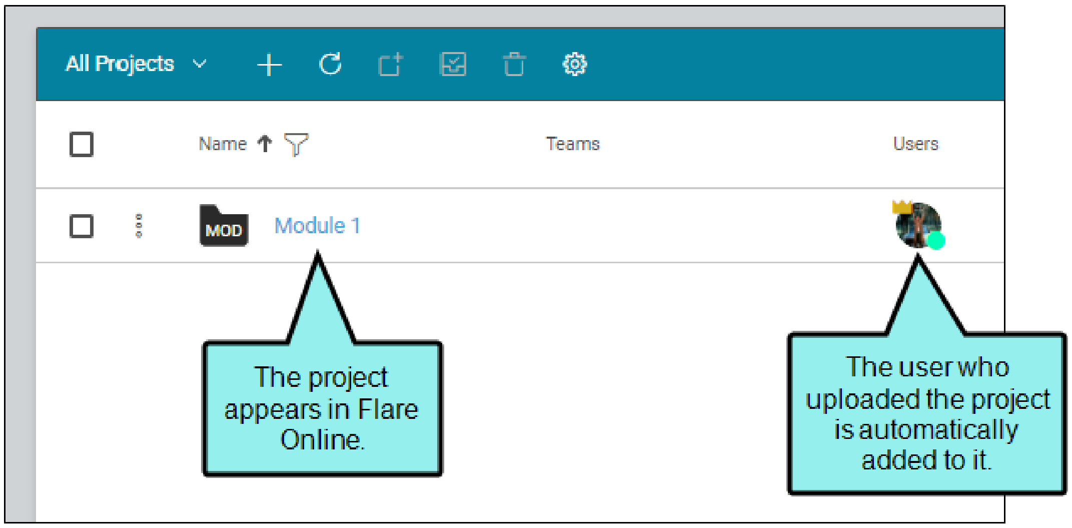


Use the icons above to interact with Flare Online (e.g., upload the open Flare project, import a project from Flare Online, upload file changes, open Flare Online in your browser).

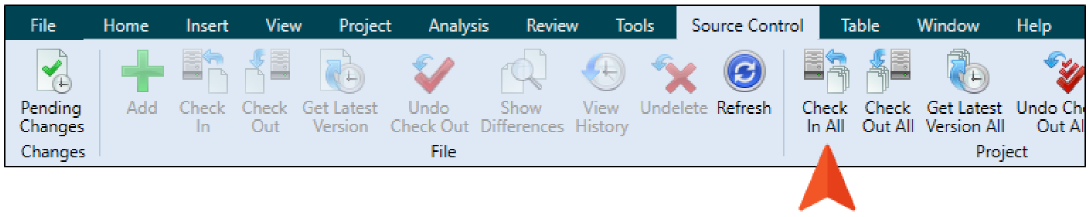
☆ Because the project is bound to TFS, you will have to check out the project file before you can bind the project (Flare Desktop will prompt you to do this). Binding the project will establish the connection between Flare Desktop and Flare Online, and creates a copy of the project in Flare Online.


Check Out
?
Files/Folders to check out:
Name
Change
Folder
Module 4.flprj
/
Check out the project file before you can bind the project.
Lock files (other users will be unable to check these files in)
Check Out
Cancel


After the project files are checked out, you are ready to bind to Flare Online.
Bind Project
?
Select License
FictionSoft
Enter project name
Module 4
Enter project description
OK
Cancel

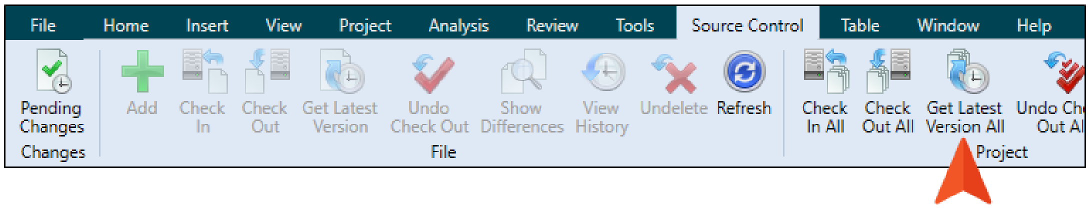

All Projects
Name
Teams
Users
MOD
Module 1
The project appears in Flare Online.
The user who uploaded the project is automatically added to it.

☆ Click **Check In All** in the Flare Desktop Source Control ribbon.

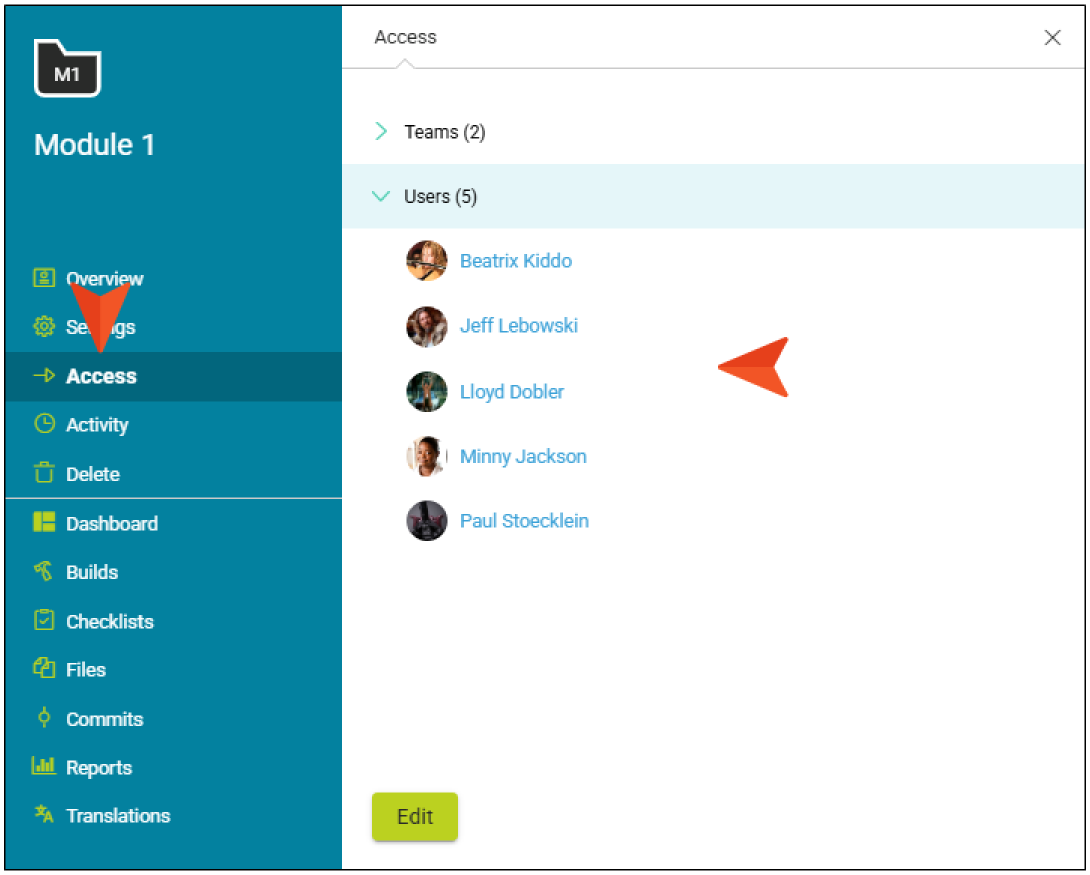


If any writers on the team do not already have a local copy of the project, they should import the project from TFS so they can work on the project locally (they cannot import from Flare Online because the project's primary source control location is TFS). To import the project, they will use the Import Project From Source Control Wizard. For more information about importing projects from source control, see the Flare Desktop Help system.

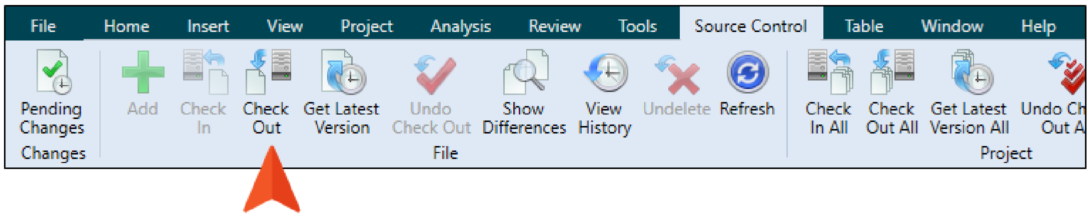
Writers on your team who already have the project locally do not need to reimport. They simply need to get the latest version of the project by clicking **Get Latest All** in the Flare Desktop Source Control ribbon.



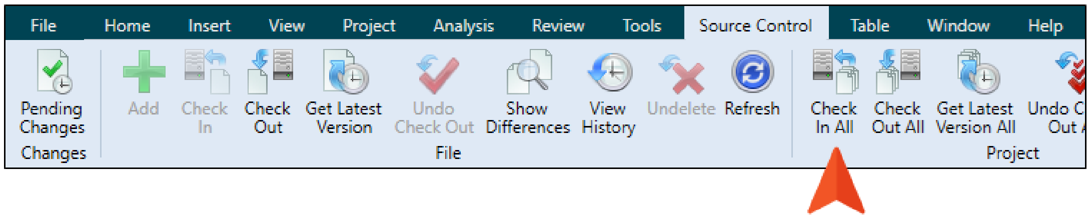
Now that everyone has the updated project file, you can all work on the project and make changes using TFS. However, a user with the "Manage Teams/Projects" permission also needs to assign your teammates to the project in Flare Online .



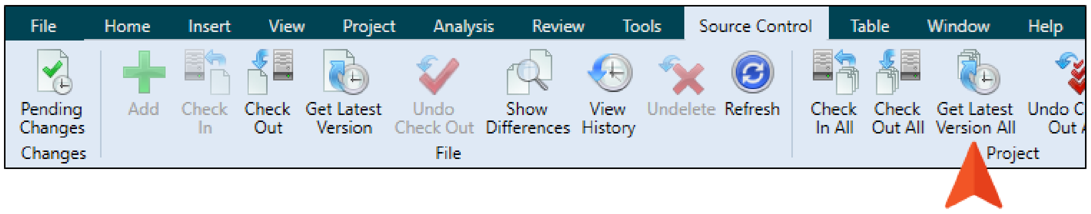
As you work on files in Flare Desktop, you should check out your files using Flare Desktop's **Check Out** feature on the Flare Desktop Source Control ribbon (or using automatic checkouts, if you have this enabled).



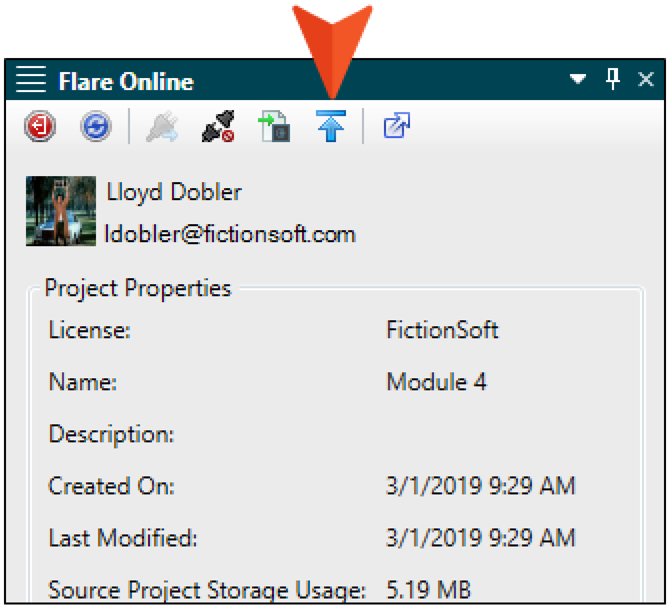
After you are finished making changes, you should check in your changes to TFS. You can do this by clicking **Check In All** on the Flare Desktop Source Control ribbon. This will upload your changes to TFS.



Once your changes are checked in, you should make sure that you have the latest version of your teammates' files, as well. To do this, click **Get Latest Version All** on the Flare Desktop Source Control ribbon. This will download the latest changes of all of the files to your local version of the project.



When you and your teammates are finished making changes, you need to synchronize your local copy of the project with the copy of the project in Flare Online. To do this, click in the Flare Online window pane.



Lloyd Dobler
ldobler@fictionsoft.com

Project Properties

License:	FictionSoft
Name:	Module 4
Description:	
Created On:	3/1/2019 9:29 AM
Last Modified:	3/1/2019 9:29 AM
Source Project Storage Usage:	5.19 MB

This will push your local changes up to Flare Online. It is important to remember that the last person who pushes their changes to Flare Online wins, so be sure that everyone has the latest version of the project files before you push (or even limit some users' permissions so they are not able to push;) so you do not have out-of-date files in Flare Online.

Once your files are up in Flare Online, you can publish output and manage your project.

How to a Use Dual Binding—Git

In the past, the primary reason for having a Git-Flare Online dual-bound setup was to be able to use branching in your project. That's because in past versions you could only push files to Flare Online from the master branch. However, you can now push files and changes from any Git branch up to Flare Online. Therefore, unless you have another good reason for using a Git-Flare Online dual-bound setup, we recommend that you use a single-bound setup instead, with just one binding between Flare Desktop and Flare Online. But if you still would like to use a Git-Flare Online dual-bound relationship, the following steps show how to set it up.

1. Open the project in MadCap Flare Desktop.
2. Be sure your project is already bound to Git.
3. Make sure all changes are committed before binding to Flare Online.
4. Select **View > Flare Online**. The Flare Online window pane opens.
5. Be sure to log in if you aren't already.
6. Upload (bind) your project to Flare Online.
7. Push the project file to Git.
8. In Flare Online, assign users to the new project and make sure they have “Push” permissions. This gives the users the ability to push local content to the project in Flare Online.
9. Newly assigned users must now do one of the following, depending on whether or not they already have a copy of the project on their local machine:
   - **Have a Local Copy** Pull the latest version of the project file from Git.
   - **Do Not Have a Local Copy** Import the project from Git using the Import Project from Source Control Wizard.

IMPORTANT If you are using a Git-Flare Online dual-binding, you can import projects from Git (using the Import Project From Source Control Wizard) or from Flare Online. The project will be the same. However, if you import from Flare Online, you will only be able to push changes to and pull changes from Flare Online. If you import from Git, you will be able to push changes to and pull changes from Flare Online *as well as* your main Git repository. This is because the project in Flare Online has no connection to the original Git repository, and if you import from Flare Online you will not have those source control bindings.

Either option will give users the most current version of the project.

10. In Flare Desktop, make changes to your files. When working in a Git-Flare Online dual-bound situation, you can make changes to both Flare Online and Git, using the following actions:
    - Pull Pull your teammates' changes from Git and add them to your local project.
    - Push Push your changes to source control.

    NOTE It does not matter which location (i.e., Git or Flare Online) you push to or pull from first, as long as you push the files to both Git and Flare Online and that everyone on your team uses the same workflow.

11. (Optional) If the changes in Git or Flare Online become out-of-sync with each other (i.e., changes are made in one location but not the other), the Resolve Conflicts dialog opens. If this happens, use the dialog to accept or reject other users' changes.
12. When you are done making changes, push your final changes to Flare Online. To do this, click in the Flare Online window pane. See "Pushing in a Dual-Bound Model" on page 36.

What’s Noteworthy?

IMPORTANT There are multiple workflows you can use when working with a Git-Flare Online dual-binding. The steps above are only one example of how you might perform this process. Because of the potential for file conflicts, it is essential that you establish a workflow for using a Git-Flare Online dual-binding, and make sure that all your team members follow the same steps.

NOTE If you are using Git as your third-party source control tool for the first binding, the dual-bound model works slightly differently than it does if you are using another source control provider. In this situation, you will still use the Flare Online window pane and Source Control ribbon in Flare Desktop to manage your changes. However, you are able to pull and push files from and to *either* Flare Online or your Git repository. As a result, your Git repository and your Flare Online repository might be completely different, and you may encounter conflicts. In this situation, Flare Desktop's Conflict Resolution dialog will open and you can accept or reject the changes. It is recommended that you establish an internal workflow to dictate the order in which you pull and push from and to each repository to prevent conflicts and ensure that your files in Flare Online stay up-to-date. However, an even better solution is to use a single-bound setup instead of dual-binding.

NOTE If you are using a Git-Flare Online dual-binding, and you encounter conflicts in one repository (i.e., Git or Flare Online), you will likely encounter them in the other location as well. Conflicts in one repository will most likely need to be resolved in the other repository. See the Flare Desktop Help system.

NOTE For more information about source control tasks and managing dual-bound projects with the various third-party providers, see the Flare Desktop Help system.

Moving From Dual-Bound to Single-Bound

If you have been using a dual-bound setup for source control, you might find that you want to change to a single-bound project. One reason you might decide to do this is branching, which previously was not available in MadCap Flare Online, but is now supported for Git.

The steps for moving from a Git-Flare Online dual-bound setup to a single-bound setup are slightly different than those for other provider configurations (i.e., Perforce Helix Core, Subversion, Team Foundation Server). Keep in mind that if you are using a branching solution with a non-Git providers, those branches are still not be supported in Flare Online. You can only push branches to Flare Online if you have a single-bound setup or a Git-Flare Online dual-bound setup.

How to Move From Dual-Bound to Single-Bound for a Git-Flare Online Setup

If your first binding is to Git and your second is to Flare Online, complete the following steps locally in Flare Desktop.

1. Each author on the team should commit all outstanding changes and synchronize each local branch with the corresponding remote branch.
2. It's possible, or even likely, that not all writers on the team have all of the remote branches locally, and they don't necessarily need to. But between all of the team members, all of the remote branches that you want to keep should be pulled down to someone's local repository by selecting the remote branches and switching to them via Flare Desktop's Branch Management dialog. In fact, just one person on the team could do that. Then, once the first binding is removed using the steps below, the local branches can be published (i.e., pushed) to the remote Flare Online repository.
3. Each author on the team should then complete the following steps:
    1. Select **Project > Project Properties**.

       You can also remove the first binding by using the Settings view of the Source Control Explorer.
    2. Select the **Source Control** tab.
    3. Next to the **Remotes** field, click ⋯.
    4. Select the **origin** row (which represents your first binding) and click **Remove**. This leaves just the Flare Online binding.
    5. Click **OK**.
    6. In the Project Properties dialog, click **OK**.

NOTE Your Git commit history will be retained when you move from a dual-bound configuration to a single-bound setup. That's because the history is recorded in your local Git repository where your project exists. Removing the first binding will have no effect on the commit history.

How to Move From Dual-Bound to Single-Bound for Non-Git Providers

If your first binding is to a non-Git provider (Perforce Helix Core, Subversion, Team Foundation Server), complete the following steps in your project.

1. Each author on the team should commit all outstanding changes and synchronize.

   NOTE Since Flare Online does not support branches or streams from non-Git providers, keep in mind that you will not be able to move those up to Flare Online. However, you could begin creating new branches in Flare Desktop and pushing those up to Flare Online.

2. Each author on the team should then complete the following steps:

   a. Select **Project > Project Properties**.

   b. Select the **Source Control** tab.

   c. In the section showing your first binding, click **Unbind Provider**. (The second binding to Flare Online is shown in a separate section; leave that binding alone.)

   d. Click **OK**.

   e. In the Project Properties dialog, click **OK**.

CHAPTER 3

Main Activities for Source Control

Some activities are particularly common and important when it comes to this feature.

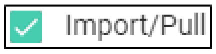
This chapter discusses the following:

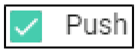
Committing and Synchronizing in a Single-Bound Model

If you are working in a single-bound model (see "Single-Bound Projects" on page 8), you author content in Flare Desktop. When you want to transfer your changes to Flare Online, you must commit and synchronize the changes.

Permission Required?

For this activity, you must have the following permission settings:


✓ Import/Pull


✓ Push

Commit

To commit a changed file is to record it to the local repository. Essentially you are saying, “I’m ready to transfer this content up to the cloned project in Flare Online.” Committing files gives you the opportunity to organize them into different groups when you add them to Flare Online. You can also add a unique comment to each commit.

☆ EXAMPLE You’ve made changes to 23 files in your project. Maybe 17 of the files are related to Feature A that you are documenting and the other 6 are related to Feature B.

Suppose your company policy is that you must add a comment each time you upload changes and provide a summary of what you did. To keep the summary for Feature A separate from Feature B, you decide to do two commits.

First, you use the Pending Changes window pane in Flare Desktop to select the 17 changed files related to Feature A. Then you perform the commit and add a relevant comment.

After this, you select the 6 changed files related to Feature B and perform another commit with a different comment.

How to Commit Files in a Single-Bound Project

1. Open the project locally in MadCap Flare Desktop and make your changes.

   > NOTE If you are using Git branching, make sure the appropriate branch is active. When you commit the changes, it will be for that branch.

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

   - **Status Bar** In the lower-right of Flare Desktop, click (a number indicates how many files contain changes that need to be committed).

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

   - **Pending Changes Window Pane** From the **Source Control** ribbon, open the Pending Changes window pane. Select the files in the window pane that you want to commit, and in the local toolbar click .
   - **Ribbon** Select **Source Control > Commit** (for selected files) or **Source Control > Commit All** (for all files in the project).
   - **Right-Click** If you have the Content Explorer, Project Organizer, Pending Changes window pane, or File List open, 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.

3. 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. Click **Commit**.

Synchronize

To synchronize means to pull, then push, committed changes between the cloned project in Flare Online and your local project in Flare Desktop. This allows you to retrieve changes that other writers have uploaded to Flare Online from their local projects, and they can get your changes as well. You cannot synchronize changes from Flare Online, but you can from Flare Desktop.

How to Synchronize Files in a Single-Bound Project

It is possible to use the Pull and Push options in the Flare Desktop interface individually to synchronize your files, but it is more convenient to use the Synchronize option instead. Flare Desktop will first pull changes from the project in Flare Online. After that, it will push changed files from the local project up to Flare Online.

1. Do one of the following in the project locally in Flare Desktop, depending on the part of the user interface you are using.

   NOTE If you are using Git branching, make sure the appropriate branch is active. When you synchronize files, it will be for that branch.

   - **Status Bar** In the lower-right of Flare Desktop, click 2↑ 0↓ (a number indicates how many commits need to be pushed or pulled).

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

   - **Ribbon** Select **Source Control > Synchronize**.
   - **Right-Click** If you have the Content Explorer, Project Organizer, Pending Changes window pane, or File List open, 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.

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

3. In the Select Remote for Synchronize dialog, select the remote repository (if necessary) and click **OK**. If you are using a dual-bound setup, origin is typically the name of the repository for the first binding, and MadCapCentral is the name for the Flare Online binding.

   If no conflicts were discovered during the synchronization, you do not need to continue with the following steps; you are finished.

   If conflicts were found (i.e., a remote file is different from the version in your local repository), the Resolve Conflicts dialog opens. Continue with the following steps.

4. Do one of the following:

   - If you want to accept all of the differences between the remote and local files, thus merging them, click **Auto Merge All**. If this step is a success, you do not need to continue with these steps.

   - If you want to review the differences in the files side by side and resolve each conflict (or if auto-merging is not possible due to conflicts occurring in the same location in a file), click **Resolve**. The Resolve Version Conflict dialog opens.

5. From the Resolve Version Conflict dialog you can choose from the following options:

   - **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 authors.

   - **Discard external changes** Automatically removes changes from other authors and keeps your changes.

6. 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** Use the key at the top of the dialog, as well as the color coding on the local and server sides, to determine if a change has been added (new), deleted, changed, moved, or is in conflict (difference occur in the same paragraph). For conflicts in the same paragraph (i.e., areas where a diagonal line is shown), you can click the icon next to either the local or server change and choose **Keep Change**. This will copy that change to the text area at the bottom of the dialog.
   - **Type Content** If you want to use your changes as well as those from another author, 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.
7. After all conflicts have been resolved, click **OK**. A message lets you know that a backup of the file has been created in case you need to roll back to it. Click **OK**.
8. Click **OK** in any remaining dialogs.
9. Because you encountered conflicts, your changes were not pushed up to Flare Online. Therefore, click the **Synchronize** option again to complete the process.

What's Noteworthy?

NOTE If an author needs to work with the project in Flare Desktop after it is created in Flare Online—because advanced features are needed for the project—the user needs to (1) have access to the project, and (2) open Flare Desktop and import the project from Flare Online. If additional changes are made in Flare Desktop or Flare Online, the work would need to be synchronized between the local and remote repositories.

NOTE From the Projects page, you can view the files and commit history for a project.

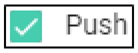
Pushing in a Dual-Bound Model

If you are working in a dual-bound model, you will author your content in Flare Desktop and then use your third-party source control tool to synchronize your files with those from other writers. After this, you can use Flare Desktop to push those changes to Flare Online.

NOTE It is possible for multiple people working on a dual-bound project to push files to Flare Online. However, if you are using a source control provider other than Git for the first binding, the most recently pushed files are the ones that are used in Flare Online. In other words, the last person to push "wins." To avoid issues, be sure that you have the most recent version of file changes from all other writers in your local project before you push. You may even want to limit users' permissions so only one or two people are allowed to push files to Flare Online.

Permission Required?

For this activity, you must have the following permission setting:



✓ Push

How to Push Files in a Dual-Bound Project

1. Open the project in MadCap Flare Desktop.
2. Get the latest version of the files from your third-party source control provider.
3. Select **View > Flare Online**. The Flare Online window pane opens.
4. Be sure to log in if you aren't already.
5. In the local toolbar of the Flare Online window pane, click .
6. In the dialog to select the remote repository, make sure **MadCapCentral** is selected. If it is not, click the drop-down and choose it. Then click **OK**.

   A progress dialog shows you the status of the push. If the changes are pushed successfully, a confirmation will appear.

What's Noteworthy?

NOTE When you are dual-bound using Git-Flare Online, you can push to both Flare Online and Git. It is possible that the files in the main Git repository and the files in Flare Online will become out-of-sync if changes are made in one repository (i.e., Git or Flare Online) and not made in the other. If this happens, you will see the Flare Desktop Conflict Resolution dialog. You can use this dialog to accept or reject other users' changes. For more information about merging source control files and conflict resolution, see the Flare Desktop Help system.

NOTE For more information about source control tasks and managing dual-bound projects with the various third-party providers, see the Flare Desktop Help system.

Setting Up Project Linking

Links to external projects—via Global Project Linking, runtime merging of projects, and multilingual output—are supported when you are building output. In order for this to work, you must have the necessary projects in Flare Online and set up the project linking properly.

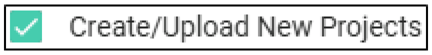
Depending on the type of project linking you are using, here is what we mean by the "main" project:

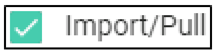
- **Global Project Linking** The main project is the child (or target). The linked project is the parent.
- **Runtime Merging** The main project is the parent. The linked project is the child.
- **Multilingual Output** The main project is the one using the original language. The linked project has a different language.

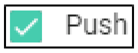
In other words, the project you are building output from is generally considered the "main" project. For more information about each of these types of project linking, see the Flare Desktop Help system.

Permission Required?

For this activity, you must have the following permission settings:


✓ Create/Upload New Projects


✓ Import/Pull


✓ Push

Use one of the following workflows, depending on your situation.

Projects Not Bound to Flare Online

If your projects are not already bound to Flare Online, do the following:

1. Upload both the main and the linked projects to Flare Online.
2. In Flare Desktop, open the main project and build the target where the merged output will occur (if using runtime merging or multilingual output).

   Otherwise, import content from the parent project into the child project (if using Global Project Linking).
3. Push your changes from the main project to Flare Online. The main and linked projects are now connected in Flare Online. See "Committing and Synchronizing in a Single-Bound Model" on page 30 or "Pushing in a Dual-Bound Model" on page 36.

Projects Bound to Flare Online

If your linked projects are already bound to Flare Online, do the following:

1. In Flare Desktop, open the main project and build the target where the merged output will occur (if using runtime merging or multilingual output).

   Otherwise, import content from the parent project into the child project (if using Global Project Linking).
2. Push your changes from the main project to Flare Online. The main and linked projects are now connected in Flare Online. See "Committing and Synchronizing in a Single-Bound Model" on page 30 or "Pushing in a Dual-Bound Model" on page 36.

Projects Bound to Flare Online but Not Linked

If your main and linked projects are already bound to Flare Online, but not linked, do the following:

1. In Flare Desktop, open the main project and “link it” to the other project(s), depending on the feature you are using (i.e., Global Project Linking, runtime merging, or multilingual output).
2. Push your changes from the main project to Flare Online. The main and linked projects are now connected in Flare Online. See "Committing and Synchronizing in a Single-Bound Model" on page 30 or "Pushing in a Dual-Bound Model" on page 36.

What’s Noteworthy?

NOTE In the case of Global Project Linking, there is an option in the Project Import Editor in Flare Desktop to “Auto-reimport before Generate Output.” This means that the child project will automatically include any changes from the parent project when you build a target. However, in Flare Online this does not mean that the actual source files with those changes are transferred to the child project; they’re simply included in the output. Therefore, if you want those changes to be included in the source for the child project, you will need to reimport them locally using the Project Import Editor. Then when you push the child project back up to Flare Online, the repository will indicate that the changes are included in the source files.

CHAPTER 4

Branching

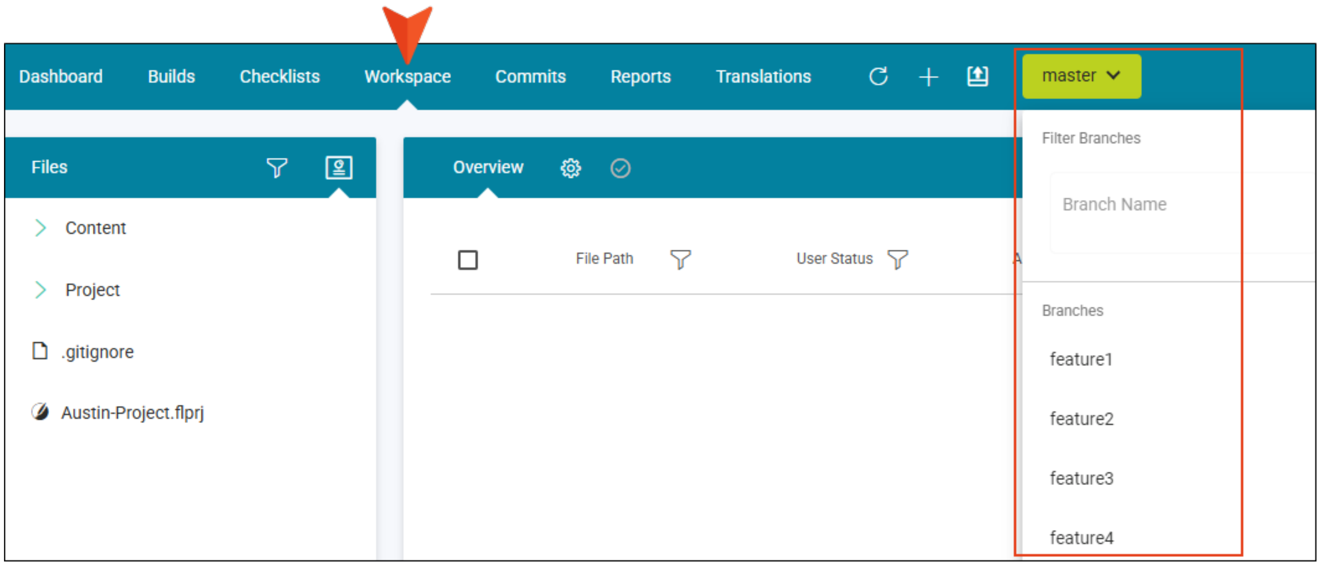
A Git branch is a pointer to a snapshot of your changes, or you can think of it as a variation from the original or main state of your files. Adding a branch lets you create a new development area for your work (e.g., when documenting a new feature, rewriting large sections of a topic, or making structural changes).

Branching is supported in Flare Online for editing files, checklists, topic reviews, translation, building output, and sites.

This chapter discusses the following:

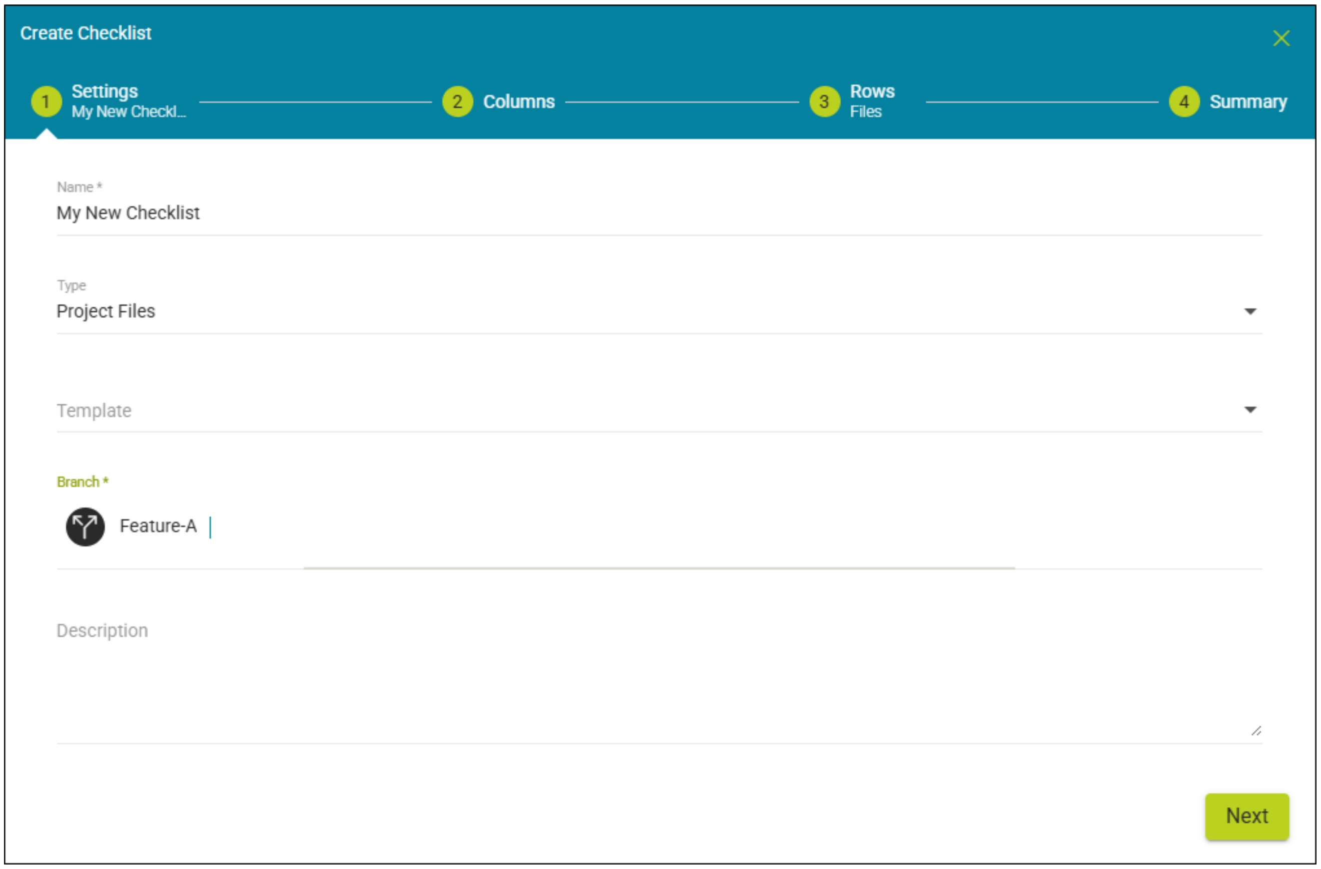
Editing Files

When you open a project and select the Workspace tab, you can select a particular branch from the top of the interface. You can then add or edit files for that branch.



Checklists

When creating a new project checklist in Flare Online, you can select a branch. This is extremely important due to the nature of content associated with checklists. When you are working on a checklist related to projects (as opposed to a generic checklist), it usually coincides with *content that is in a state of development*, rather than *content that has already been published*. And branching is the best way to isolate content that is in a state of development so that checklists can be associated directly with it.


Create Checklist

1 Settings My New Checkl... — 2 Columns — 3 Rows Files — 4 Summary

Name *
My New Checklist

Type
Project Files

Template

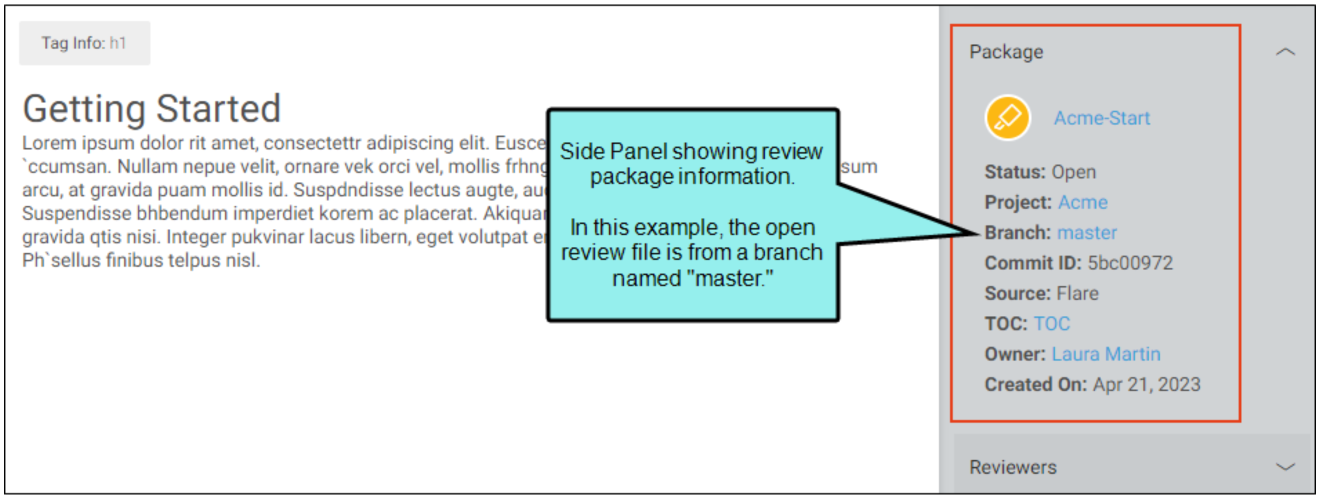
Branch *
Feature-A

Description

Topic Reviews

You can send files in a particular branch for a review in Flare Online. Reviewers (e.g., SMEs and authors) can see which branch a file is coming from on the right side of the interface. However, nothing special needs to be done in order to make edits or comments in the file.

NOTE Reviewers who are authors can alternatively click the file icon in the review package overview or at the top of the Review Editor. The Review Files profile opens to show package information including branch association, and also to update various file settings.


Tag Info: h1
Getting Started
Lorem ipsum dolor rit amet, consectettr adipiscing elit. Eusce `ccumsan. Nullam nepue velit, ornare vek orci vel, mollis frhng arcu, at gravida puam mollis id. Suspdndisse lectus augte, au Suspendisse bhbendum imperdiet korem ac placerat. Akiqua gravida qtis nisi. Integer pukvinar lacus libern, eget volutpat e Ph`sellus finibus telpus nisl.
Side Panel showing review package information.
In this example, the open review file is from a branch named "master."
Package
Acme-Start
Reviewers

Status: Open
Project: Acme
Branch: master
Commit ID: 5bc00972
Source: Flare
TOC: TOC
Owner: Laura Martin
Created On: Apr 21, 2023

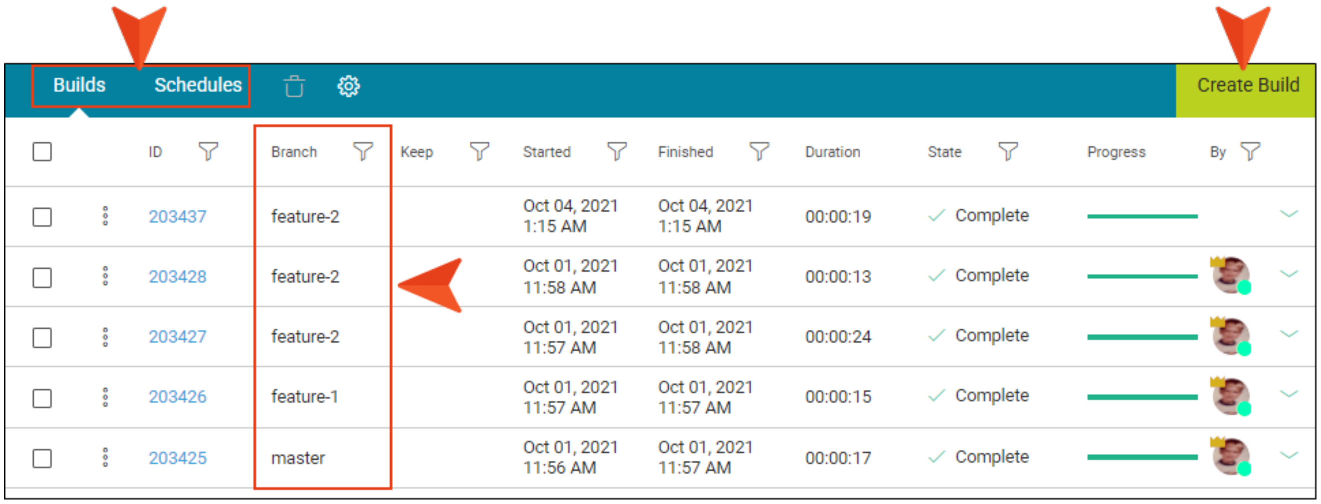
When you bring a reviewed file back into Flare Desktop and accept it so that it replaces the original file, make sure you first switch to the correct branch in Flare Desktop. The branch associated with each file is shown in a column in the File Reviews window pane in Flare Desktop.

Translation

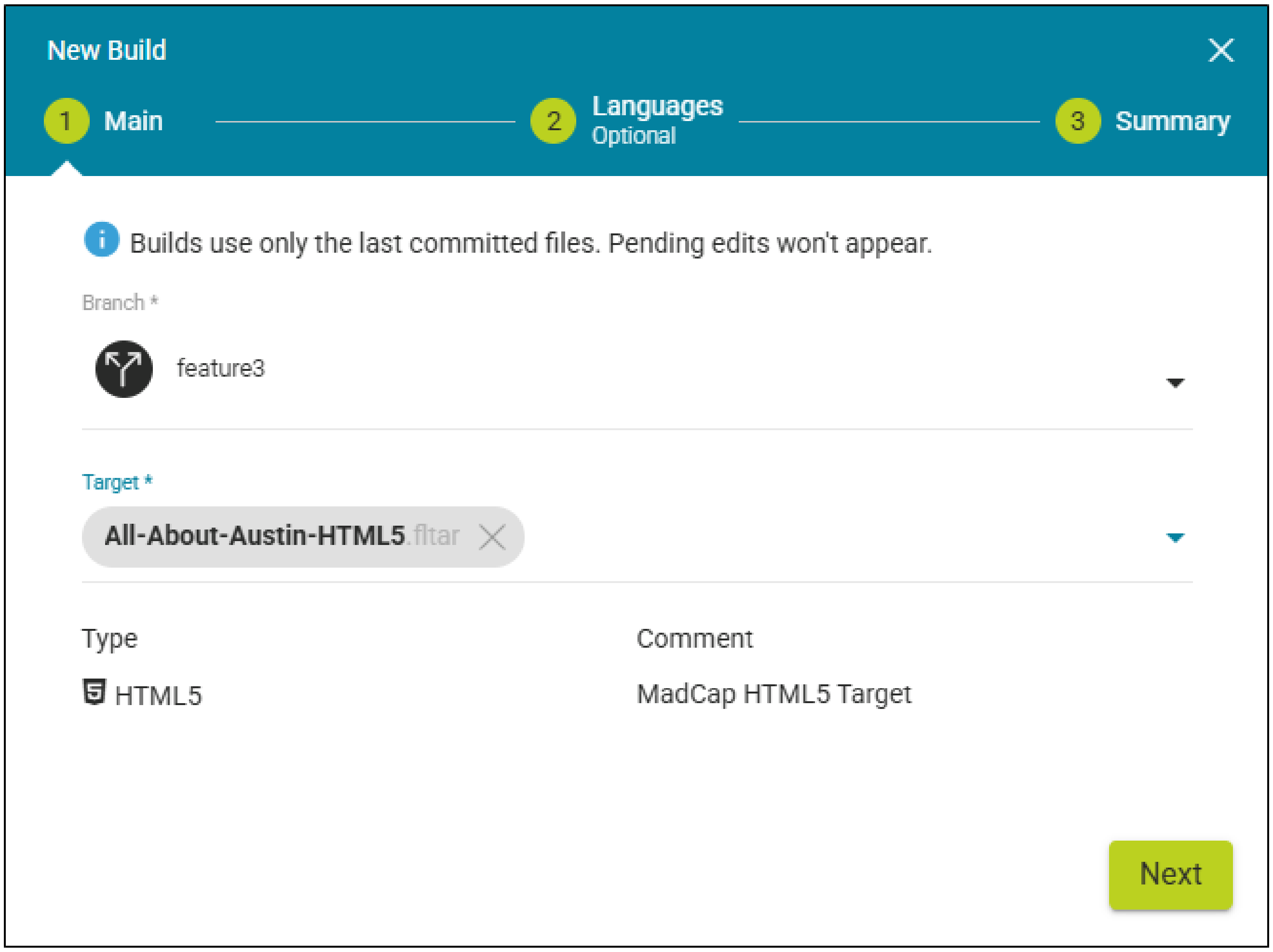
A translation branch is simply another development area for work specifically tied to the translation part of your project. Once created, a translation branch is available anywhere other project branches might be available in Flare Online.

Building Output

Flare Online supports building output for specific branches, rather than just the master branch. There are tabs in the toolbar to let you switch from manual builds to automatic scheduled builds. A button in the upper-right corner lets you initiate a build or schedule. Once you click this button, you can point to a specific branch. In addition, a Branch column in the grid displays the branch for which a particular build was generated.



ID	Branch	Keep	Started	Finished	Duration	State	Progress	By
203437	feature-2		Oct 04, 2021 1:15 AM	Oct 04, 2021 1:15 AM	00:00:19	✓ Complete		
203428	feature-2		Oct 01, 2021 11:58 AM	Oct 01, 2021 11:58 AM	00:00:13	✓ Complete		
203427	feature-2		Oct 01, 2021 11:57 AM	Oct 01, 2021 11:58 AM	00:00:24	✓ Complete		
203426	feature-1		Oct 01, 2021 11:57 AM	Oct 01, 2021 11:57 AM	00:00:15	✓ Complete		
203425	master		Oct 01, 2021 11:56 AM	Oct 01, 2021 11:57 AM	00:00:17	✓ Complete		



Builds use only the last committed files. Pending edits won't appear.

Branch *

feature3

Target *

All-About-Austin-HTML5.fltar

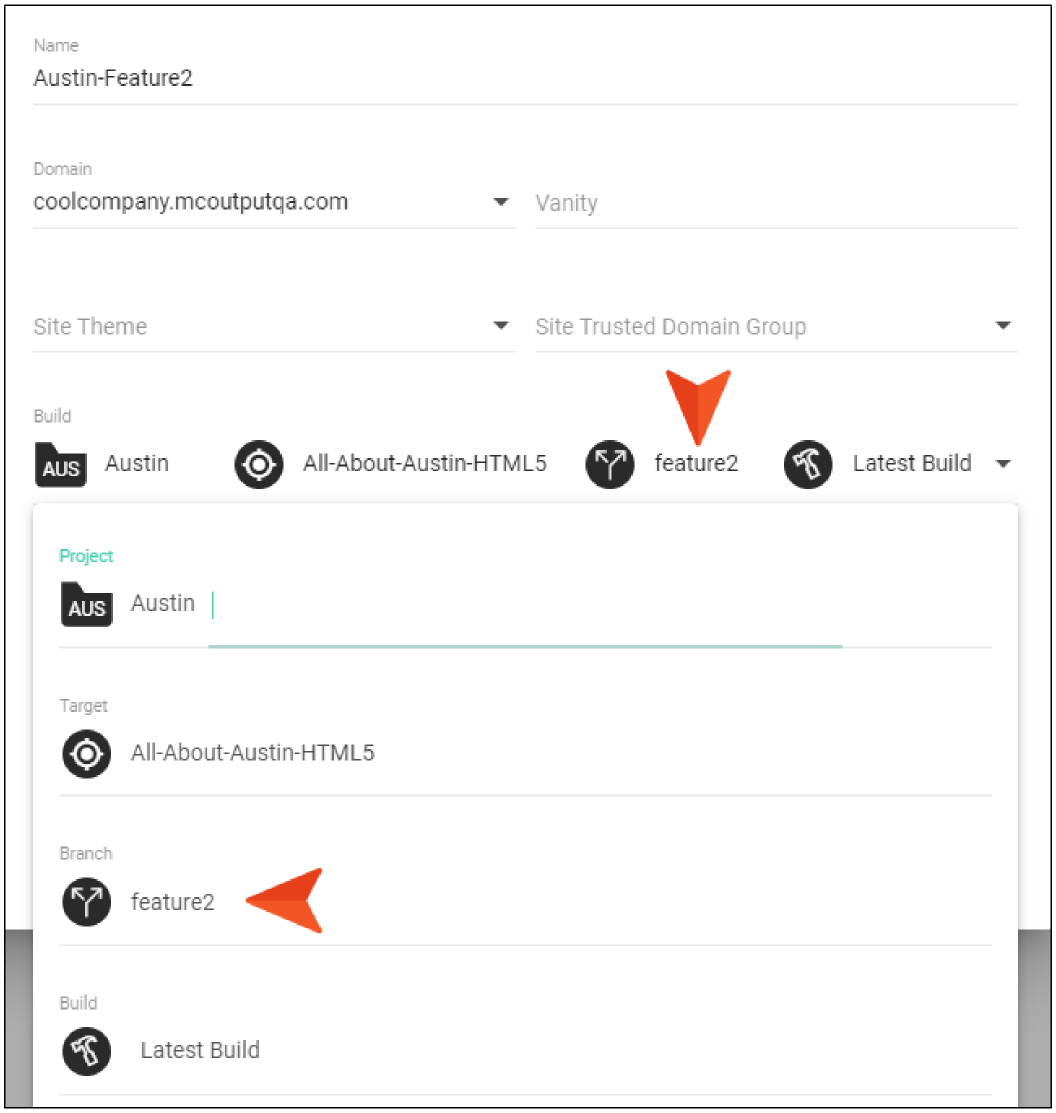
Type	Comment
HTML5	MadCap HTML5 Target

One of the advantages of using Flare Online for your translation needs is that you can easily create single language output (i.e., one target is generated for each language) or multilingual output (i.e., one target is generated for multiple languages).

NOTE If you build a target in the project for a particular branch and then publish the output from the Flare Desktop desktop directly to Flare Online, you will also see the associated branch listed in the Branch column on the Builds page.

Sites

When you create or edit a site, you can select a branch for the output. For example, this is useful if you are generating private output to circulate among individuals in your company so they can see content that is currently under development, but not accessible to the general public.



Name
Austin-Feature2

Domain
coolcompany.mcoutputqa.com

Vanity

Site Theme

Site Trusted Domain Group

Build
AUS Austin · All-About-Austin-HTML5 · feature2 · Latest Build

Project
AUS Austin

Target
All-About-Austin-HTML5

Branch
feature2

Build
Latest Build

What's Noteworthy?

NOTE For more detailed information on Git branching and how to use it with projects (including publishing branches and pushing changes to Flare Online), see the Flare Desktop Help system.

NOTE From Flare Online, you cannot create or merge branches. You can do this in Flare Desktop.

NOTE Branching is not yet available in Flare Online for reports. With reports, you are always working with files information from the master branch.

NOTE Branching in Flare Online is supported for projects that are single-bound, as well as dual-bound projects using a Git-Flare Online setup. Branching is not supported in Flare Online for dual-bound projects that are using Perforce Helix Core, Subversion, or Team Foundation Server as the first binding.

APPENDIX

PDFs

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

- *AI Assist Guide*
- *Analytics Guide*
- *Authoring Guide*
- *Branding Guide*
- *Building Output Guide*
- *Checklists Guide*
- *Conditions Guide*
- *Getting Started Guide*
- *Images and Multimedia Guide*
- *License Management and Purchasing Guide*
- *Links Guide*
- *Projects Guide*
- *Reports Guide*
- *Reviews Guide*
- *Security Whitepaper*
- *Sites Guide*
- *Snippets Guide*
- *Source Control Guide*
- *Targets Guide*
- *Tasks Guide*
- *TOC Guide*
- *Topics Guide*
- *Translation Guide*
- *Users and Teams Guide*
- *Variables Guide*
- *What's New Guide*
- *Widgets Guide*