MADCAP FLARE 2024

Architecture

CONTENTS

APPENDIX

Introduction

Should you create one project or many? What kind of relationships should your projects have with one another (if any)? Which methods, processes, and features best fit your needs? How should you structure the files within your Flare projects?
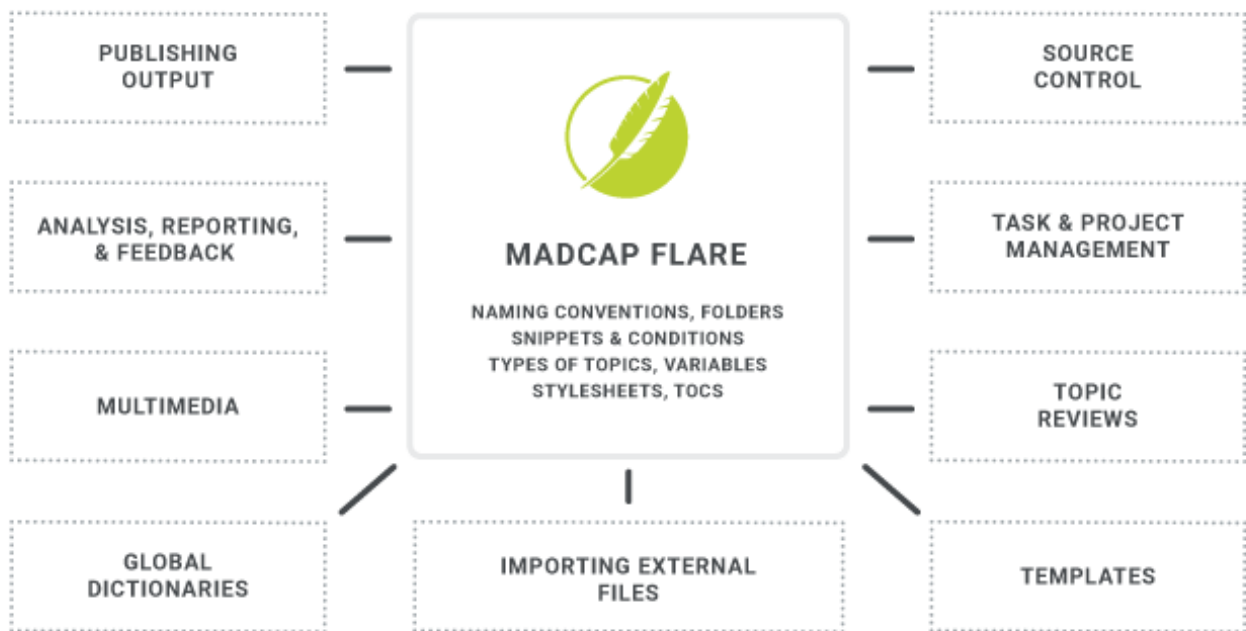
Flare is extremely flexible, which means your project universe (i.e., all of your projects, tools, features, elements, and content; and how it is all put together) might end up being somewhat unique. Therefore, you will want to take some time to plan your project architecture, both externally and internally.

- **External** The external project architecture has to do with your projects as entities, their structures, and what is *outside* of them. This includes the external processes, tools, and other factors surrounding your projects—such as source control, task management, templates, and more. See "External Architecture" on page 7.

- **Internal** The internal project architecture has to do with the structure of files and use of features *inside* a Flare project. It includes things such as naming conventions, folders, and structuring the project with snippets, conditions, variables and more. See "Internal Architecture" on page 90.

Along with the information about external and internal architecture, we have included explanations of how we (the MadCap Software Documentation Team) use the various features described and the methods we have adopted.

Keep in mind that there are best practices, or recommendations, here and there. However, there is no universal best practice for designing your architecture. What might be best for one author or writing team, but not be best for others. You need to be aware of the options available, explore the pros and cons of each, and make educated decisions.

CHAPTER 2

External Architecture

External project architecture has to do with your projects as entities, as well as the processes, tools, and other factors surrounding them.

This chapter discusses the following:

❚Flare Project Structures

You might be wondering how many Flare projects you should create. Should you put all of your content into one big project, should you split it out into several smaller projects, or should you do some of both? And if you create multiple projects that are related, which features or tools should you use? There isn't just one answer that fits everyone's situation. You will need to explore the different possibilities and consider the pros and cons of each. You might even want to create some small sample projects and test different methods.

Following are some possible project designs you might consider, and the features and tools that can be used.

Option 1: Single Project

The simplest structure is obviously to have only one Flare project, generating output from it as needed. A single project might contain files for only one purpose (e.g., to document one product or one subject), or it might contain files for many purposes (e.g., to document many products or subjects). Also, this project might be used by a single author or many authors.

Pros

It's quite simple, which means less coordination and management.

It's a good option if you have content for various purposes, but a large percentage of the files need to be shared for the different outputs. Therefore, instead of putting the content into different projects, you put it all into one project.

Cons

If you have multiple writers but no source control system, it's not a realistic solution.

Some might find it easier to manage an extremely large number of files in a single project, but others might find that a very big project with tens of thousands of files can become unwieldy. You are likely to experience slower performance in a very large project.

Option 2: Multiple Projects—No Relationship

With this model, there are more than one Flare projects, each producing output. However, there is no relationship between the projects (e.g., sharing files). The projects simply function independently of one another and all have files unique to that project.

PROJECT 1	PROJECT 2	PROJECT 3
↓	↓	↓
OUTPUT	OUTPUT	OUTPUT
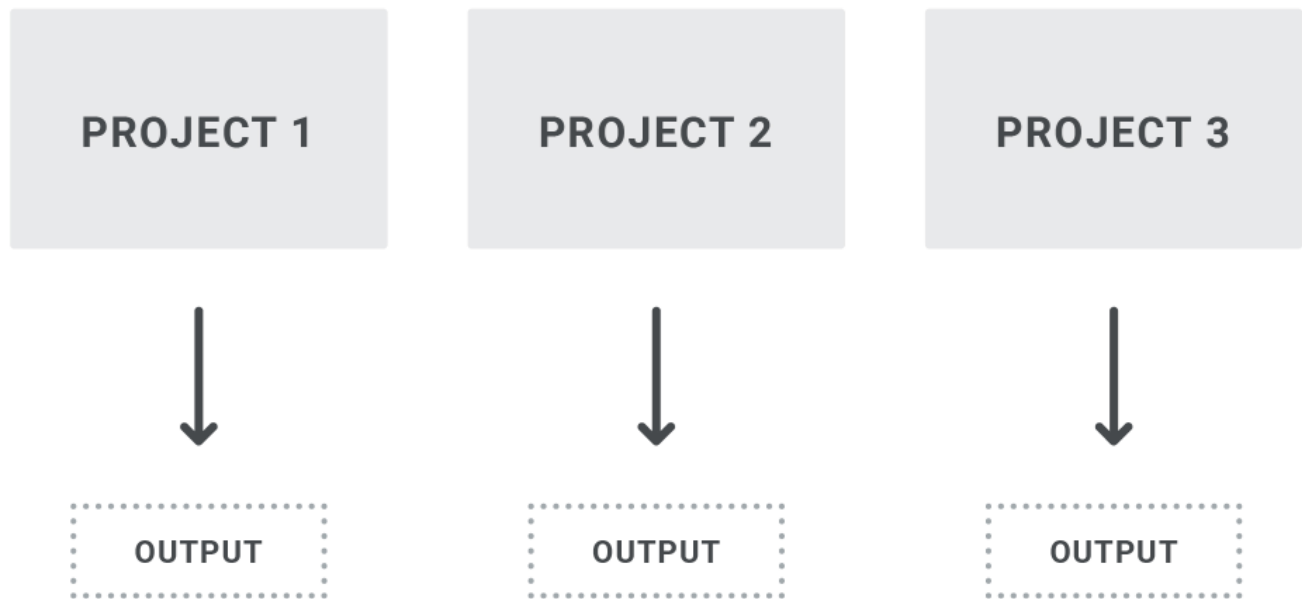
Pros

Although there are multiple projects, the structure is still simple, which means less coordination and management.

If you have multiple authors, this is a good way to let each author work independently, without having to worry about shared file conflicts.

If you break things up from one big project into multiple smaller projects, you might experience better performance.

Also when it comes to performance, keep in mind that the hardware you're using can affect that too. There are several things that could impact performance in a project.

Cons

If you have multiple writers working in the same project, but no source control system, it's not a realistic solution.


If you want the different projects to use some of the same files (e.g., a common stylesheet to keep the look and feel the same), this is not the best option. Doing this would mean maintaining a common file in multiple places, which can lead to out-of-date information and inconsistency.

Option 3: Multiple Projects—Global Project Linking

With this type of structure, you have more than one Flare project, with one or more "child" projects importing common files from a "parent" project via Global Project Linking. Rather than this being a one-time import process, a connection exists between the parent and child projects, allowing ongoing maintenance of the files to take place in the parent project and then periodically imported to the child projects. For more information see the online Help.

Because any project can serve as the parent project or as a child project, you have some flexibility in how you create your structure.

The most common structure has only one project serving as the parent, with child projects importing files from it. With the parent working merely as a repository, the outputs would be generated from the child projects.
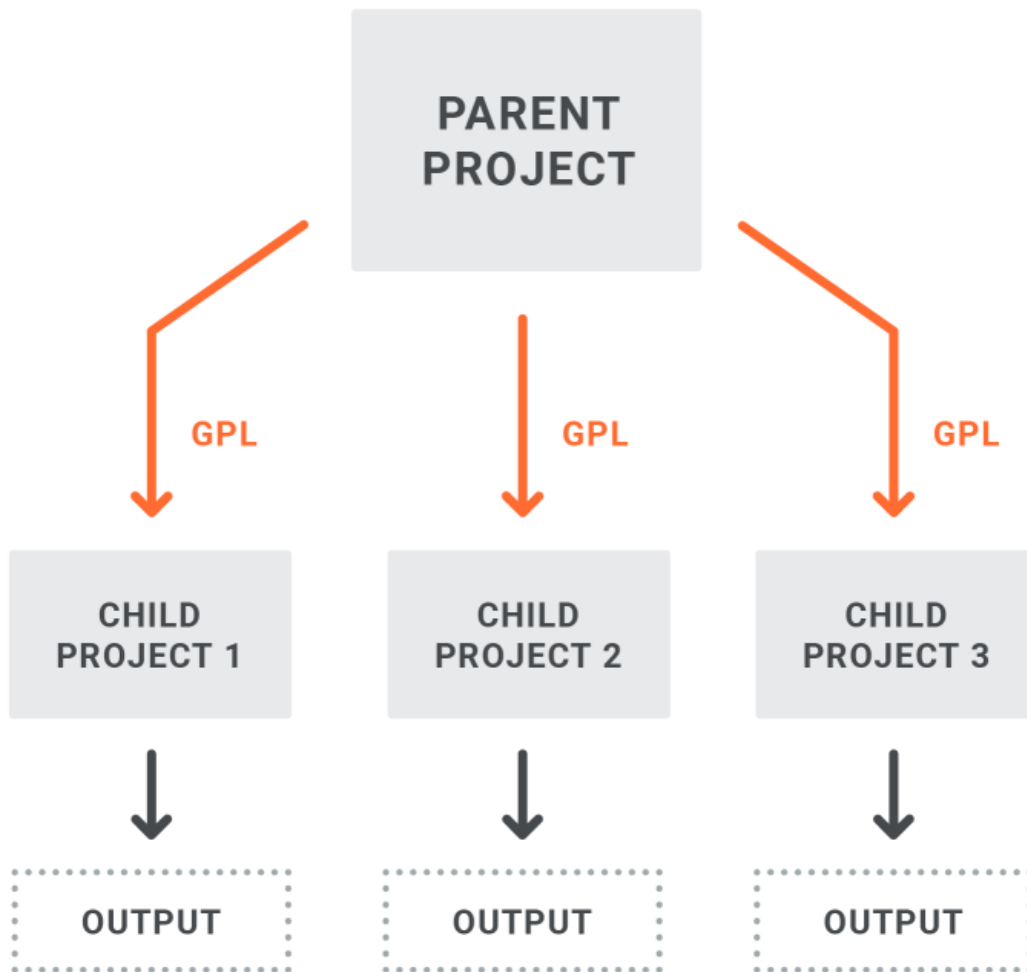
However, you might create a structure where the parent is more than just a repository, where output is generated from it as well.
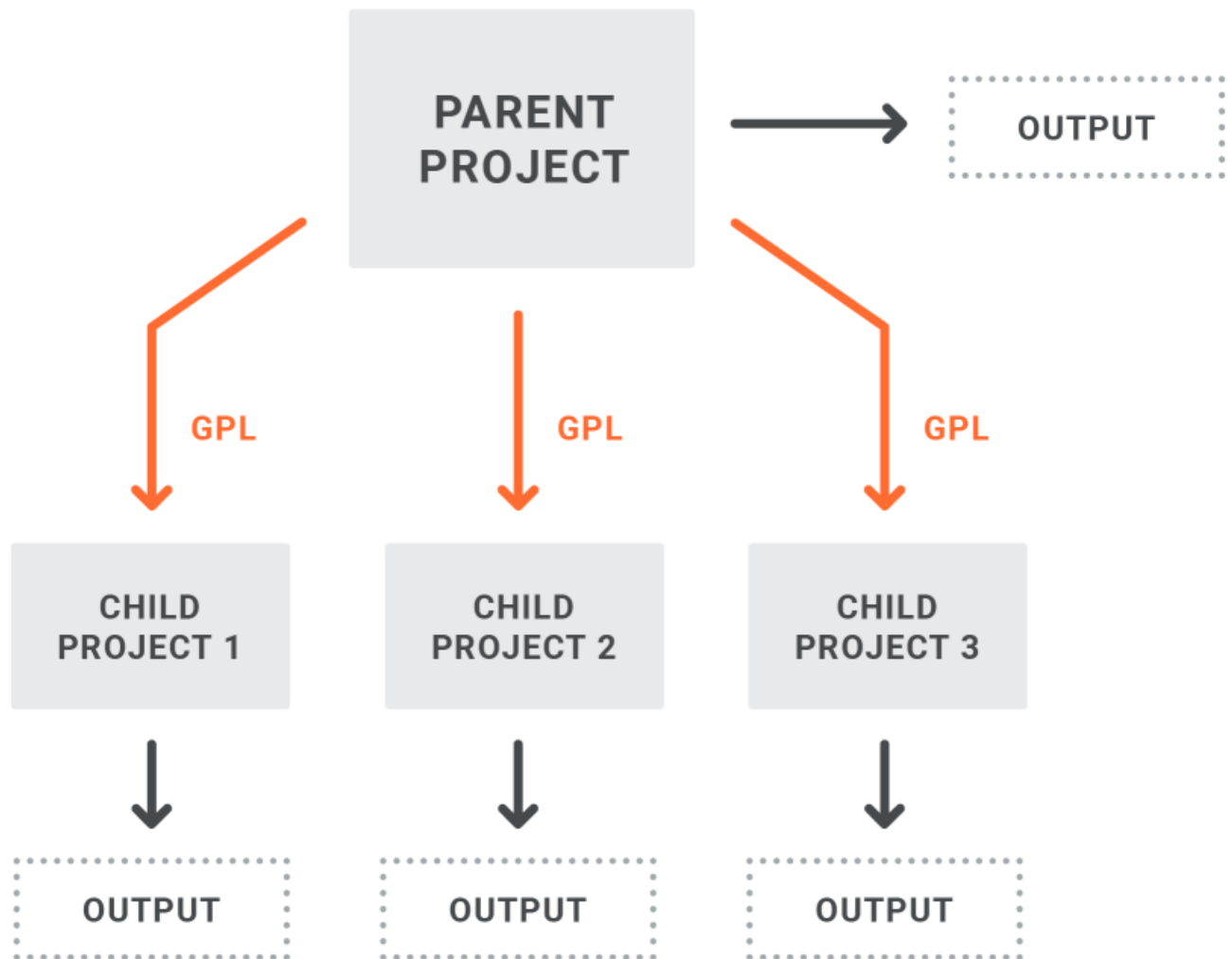
A less common solution—but certainly a possible one—might involve more than one parent project, with certain files imported between any projects you want.
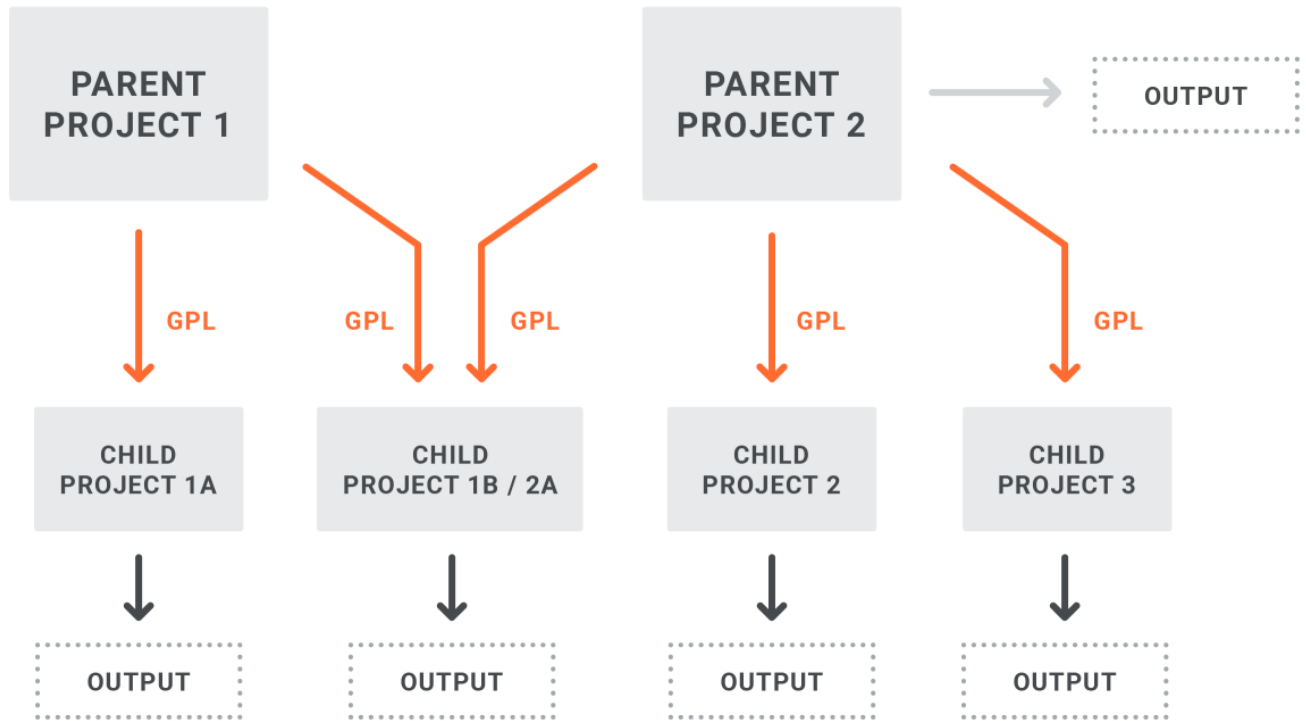
A parent project might even serve as both a parent and a child project, although that would be less likely.

<table>
<tr><td>PARENT PROJECT 1</td><td>PARENT PROJECT 2</td><td>OUTPUT</td></tr>
</table>

Pros

It allows for single-sourcing across projects by maintaining files in one place instead of many places. This helps keep all projects consistent and up to date.
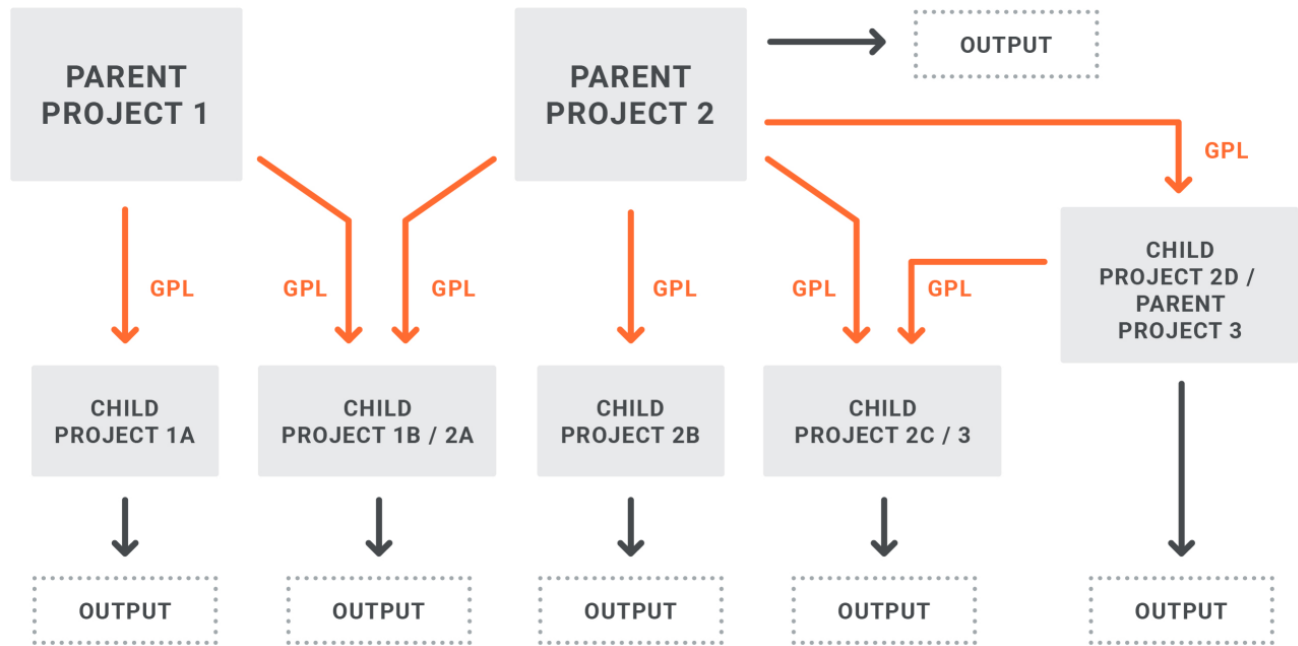
If you have multiple authors working on different projects, this is a good way to let each author leverage common files (e.g., stylesheets, page layouts, images).

Files can be imported manually or automatically from the parent when output is generated from a child project.

Imports can be controlled by file name, file type, conditions, or a combination of these.

Cons

Beware of the "threshold." In other words, you might find yourself importing such a high percentage of files from the parent that you need to ask, "Does Global Project Linking still make sense for me?"

However, this threshold is not the same for all authors. What is acceptable to one author might not be to another. It is recommended that you set up a Global Project Linking structure using copies of your projects and test it to see how it works for you.

Beware of the "domino effect." Global Project Linking tends to work best with certain kinds of files —the files in your Project Organizer and some of your resource files in the Content Explorer (e.g., conditions, skins, variables, images, page layouts). It was originally designed primarily for importing those kinds of ancillary files.

This does not mean you should never import content files, such as topics and snippets. Importing those kinds of files can be quite useful. Just know that those kinds of files tend to have links to other topics and snippets. And those other topics and snippets have links too. So when you import these types of files that have many links, you also need the files they're linked to. In that way, it is somewhat like a set of dominoes falling over. And before you know it, you're importing a far more files than originally intended and you need to consider whether you've passed your imaginary threshold.

Also, if you do not import all of the linked files, you might begin seeing many missing files (e.g., snippets, images) in the child projects. Even if you have those missing files conditioned out so that they do not affect outputs, the indication of missing files can cause extra time and effort to ensure everything is working as it should.

Option 4: Multiple Projects—Merging Projects

This structure involves multiple projects that are connected in a way that the source files are merged into a single online output. The merging is based on the table of contents (or browse sequence) in one parent project, where links point to one or more other child projects. So when you build output from the parent project, it brings in all that content from the child projects, and it looks like one seamless output from a single project, but it's actually made up of multiple projects. For more information see the online Help.
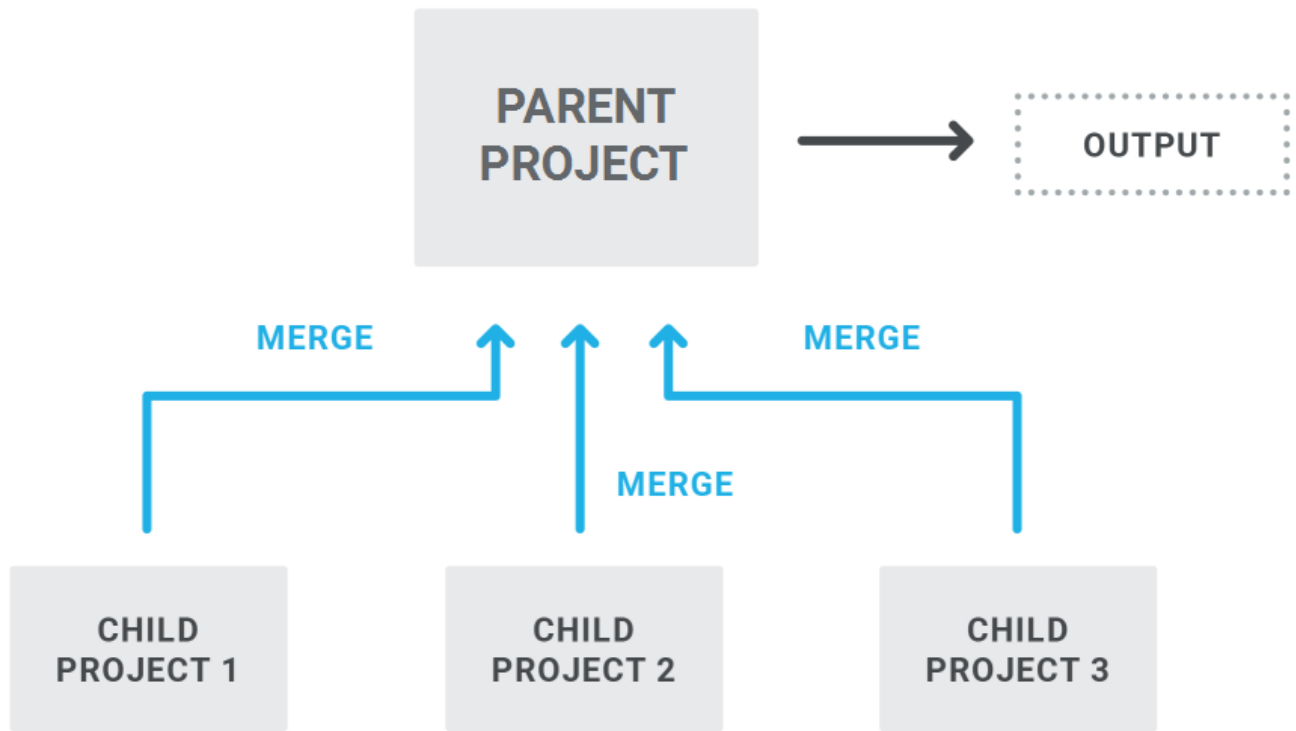
Pros

This structure can be quite simple to set up.

If you have multiple authors working on different projects, this is a good way to let each author work independently, but leverage their source files when output is generated.

Working in different projects can help prevent file conflicts.

Cons

In order to use this option, the author maintaining the parent project must have access to the targets or outputs of the child projects. This might be a concern if the various projects are not stored on the same network. Also, if a child project is moved at some point, the link from the parent project will be broken and need to be fixed.

If you want links (e.g., cross-references, hyperlinks) to exist between files of the various projects, authors must set up these links properly. This takes a little extra effort and advanced knowledge of the location of the final published output. For more information see the online Help.

If you need your different child projects to share some files (e.g., stylesheets), this structure will not solve that issue.

At this time, the most popular online outputs—HTML5 Side and Top Navigation—do not support runtime merging.

If you merge projects, synonym files will remain separate in each project. For example, if you create synonyms in Project A but not in project B, only the topics from Project A will use the synonyms when users perform searches in the output.For more information see the online Help.

Option 5: Multiple Projects—Multilingual Output

If you have translated your project into other languages, you can link Lingo projects or additional translated Flare projects to your parent project to create multilingual output. When you build the target, Flare will include translated content from each linked project. So this structure is similar to that for merging projects, except each project contains the same content but in a different language. For more information see the online Help.
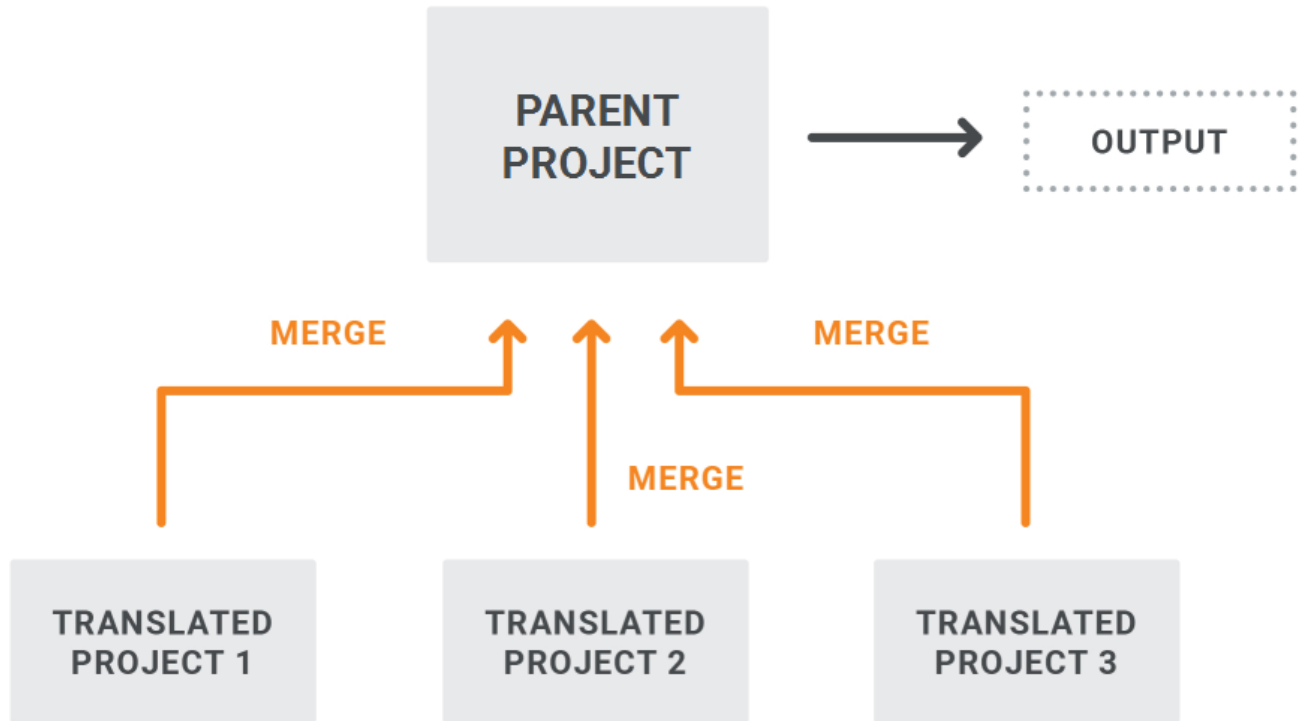
Pros

This structure can be quite simple to set up.

It is typical to have multiple translators for the different languages you need. So this is a convenient way to leverage the independent translations from separate projects.

Both online and print-based outputs are supported.
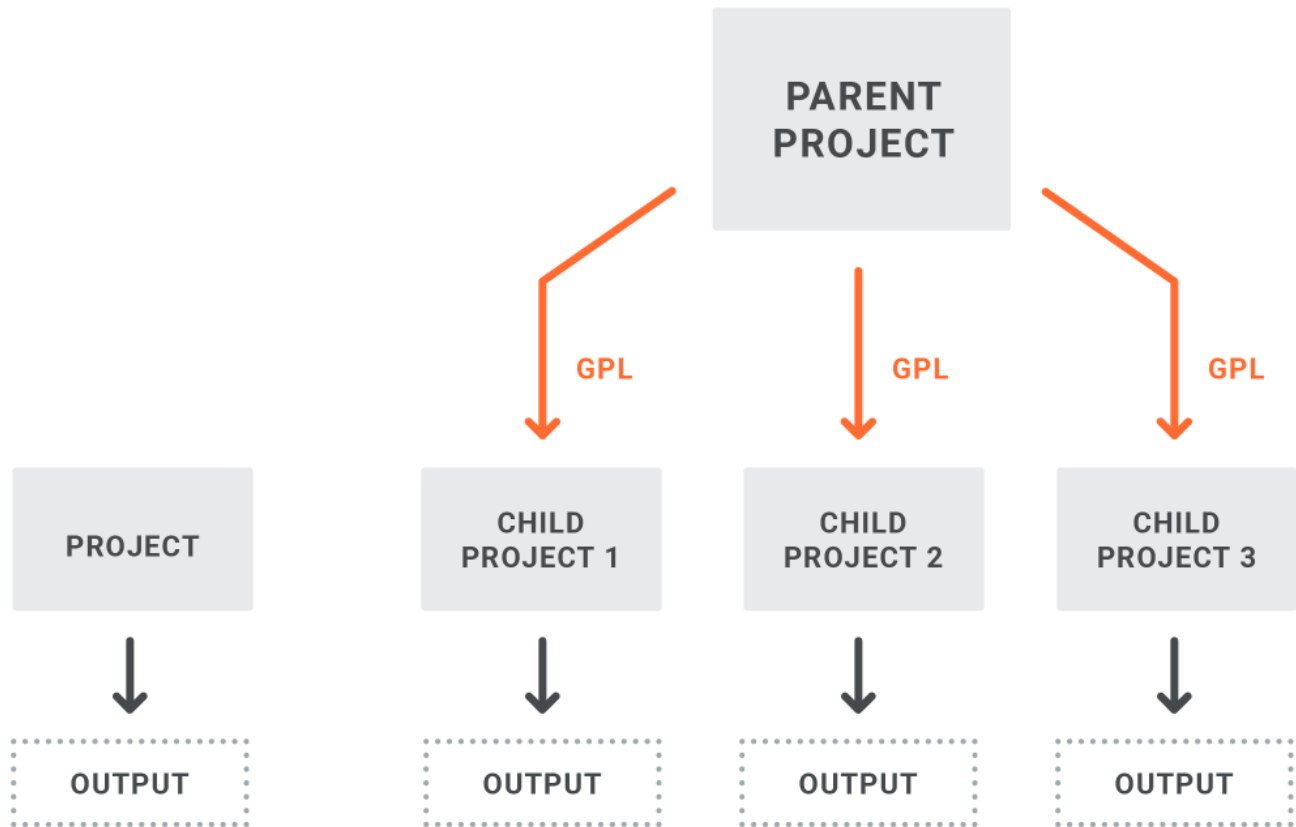
Cons

File names need to be the same in each project. This is typically not a problem if you are using Lingo to manage the projects. However, you might want to tell translators not to change any file names.

In order to build output that links directly to multilingual Lingo projects, the Flare user must have at least Lingo 10 installed on the same computer.
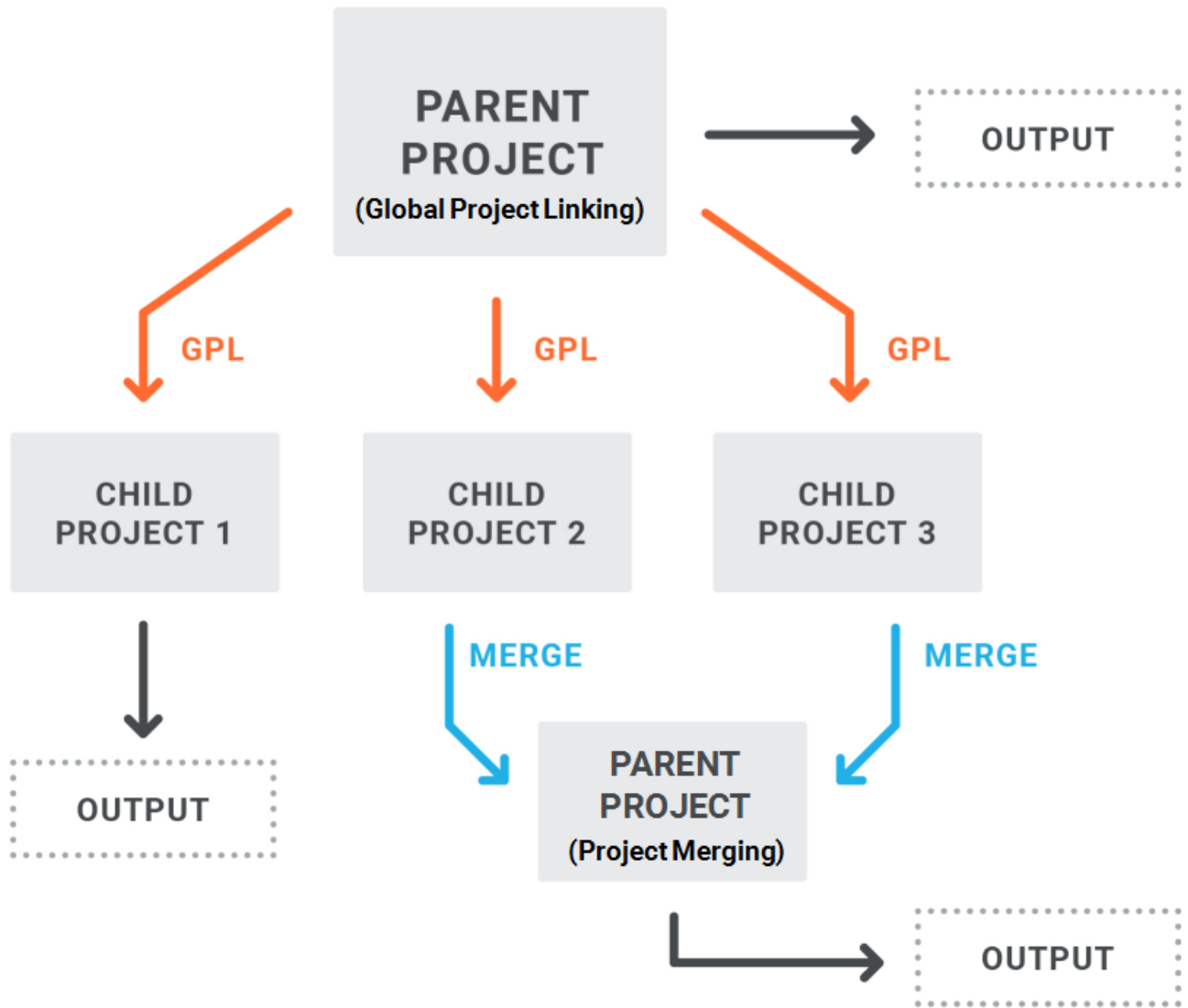
Option 6: Combination of Different Structures

It is quite common to need to use more than one of the structures just described. Therefore, your final architecture might be a combination of different structures.

Therefore, depending on your needs, you might have something like this:

Or this:

PARENT PROJECT (Global Project Linking)

OUTPUT

GPL

GPL

GPL

CHILD PROJECT 1

CHILD PROJECT 2

CHILD PROJECT 3

OUTPUT

MERGE

MERGE

PARENT PROJECT (Project Merging)

OUTPUT

Or this:

PARENT PROJECT → OUTPUT

MERGE   MERGE

CHILD PROJECT 1   CHILD PROJECT 2   CHILD PROJECT 3

MERGE   MERGE

TRANSLATED PROJECT 1   TRANSLATED PROJECT 2   TRANSLATED PROJECT 3
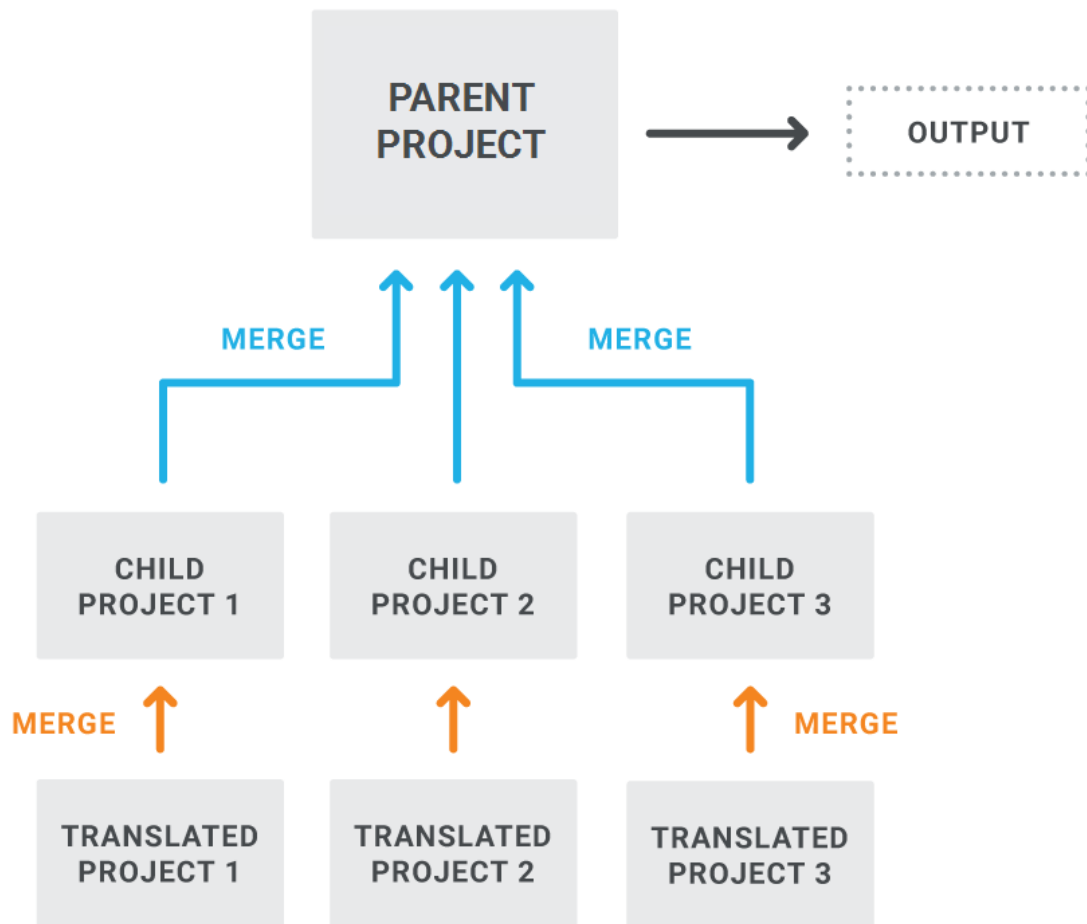
What the MadCap Documentation Team Does

Our structure has evolved over the years. At one time, our documentation for various products was split into individual projects. We then implemented Global Project Linking, with common files being imported into each child project. However, with so many files being imported, we eventually decided that it made more sense for our purposes to move all of the product documentation into a single project. Although this main project is quite large, it does not seem to be so big that it is causing any unmanageable issues. We continue to use Global Project Linking for a couple of different reasons, which are explained below.
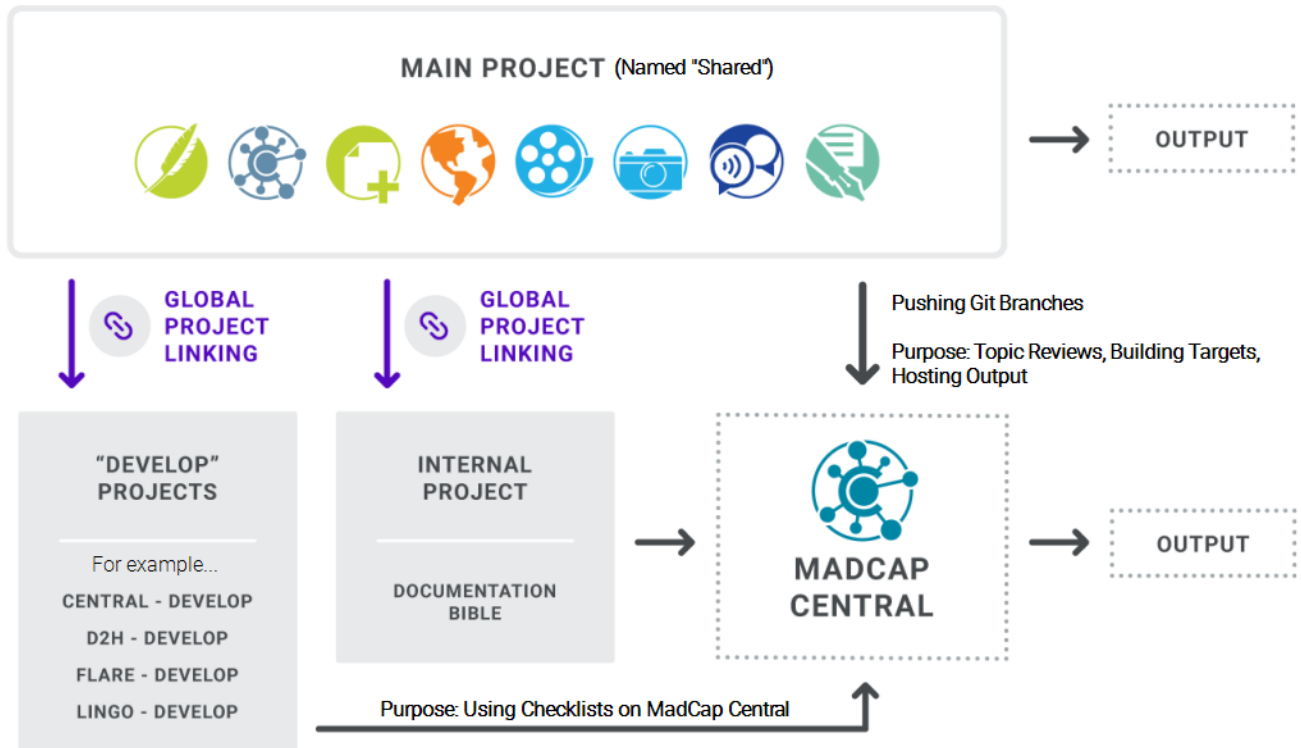
Our structure involves the following three kinds of projects.

Shared

We have one main Flare project called "Shared," where most work occurs on the various products that we need to document (e.g., Flare, Central, Doc-To-Help, Lingo). All of the output intended for end users is originates from this project. This project is single-bound to MadCap Central. We use Git branches to keep published content separate from documentation that is still in a state of development. By pushing specific branches to Central, we can have individuals review topics and snippets that aren't finished yet; we can also build "in-development" outputs on Central for various products, with MadCap Software employees being the intended audience (in addition to other outputs intended for the general public).

Documentation Bible

This project is used for internal information (e.g., documentation team processes, procedures, style guide). Most of the editing is done directly in this separate project, but we also import some resource files (e.g., stylesheets, page layouts, variables, skins) from the Shared project via Global Project Linking. This project is single-bound to MadCap Central, where we build output intended for internal consumption.

"Develop" Projects

These are smaller projects, each one based on one of the MadCap products (e.g., "Central - Develop," "D2H-Develop," "Lingo-Develop"). Based on conditions that are set on files in the main "Shared" project, we import all of the files for a particular product via Global Project Linking. We do not edit any of these child projects. These child projects are basically versions of the documentation *as it's being developed, not the finished result*. The main reason for doing this is to use checklists on MadCap Central. Since Git branching is not yet supported on Central for checklists (but it is supported for other tasks such as builds and reviews), we use this workaround for the time being. These projects are single-bound to Central. Within Central, we use these child projects for creating and managing checklists.

Where to Store Projects

When you create new Flare projects, the default location is the Documents\Projects folder. However, you can choose any location you want. For more information see the online Help.

It is recommended that you always store your project locally, rather than on a network. Of course, there may be times when you need to place a copy of your project on a network (e.g., to use source control, or to make a copy available to a co-worker). But performance and speed should be significantly better if you work on a local copy of the project rather than working on it from a remote server. Working on a remote copy of a Flare project is also not an ideal solution for team authoring, because there is not a good solution for resolving file conflicts between writers.

What the MadCap Documentation Team Does

We store projects locally in each writer's root C:\ drive for the following reasons:

- We wanted the shortest paths possible to our project files. This can make for a smoother experience when the output is integrated into a software application.

- We wanted the paths to be identical for each writer. If we had kept the projects in the default location, each path would have been different for each writer. Having the same path is important when we link to outside files. In our case, we use features such as Global Project Linking, image profiles, and image palettes. Having the same path for all writers makes these processes easier and more streamlined.

- We wanted to keep our permanent project files separate from any temporary project files (stored in the default Documents\Projects folder) that we create for testing purposes.

Source Control

In Flare, source control is the practice of managing file changes by synchronizing them with a copy of your project files on a server. Source control is even useful for single authors because it provides a means for maintaining a constant backup of all project files. For most Flare authors, source control is an integral part of their overall architecture. For more information see the online Help.
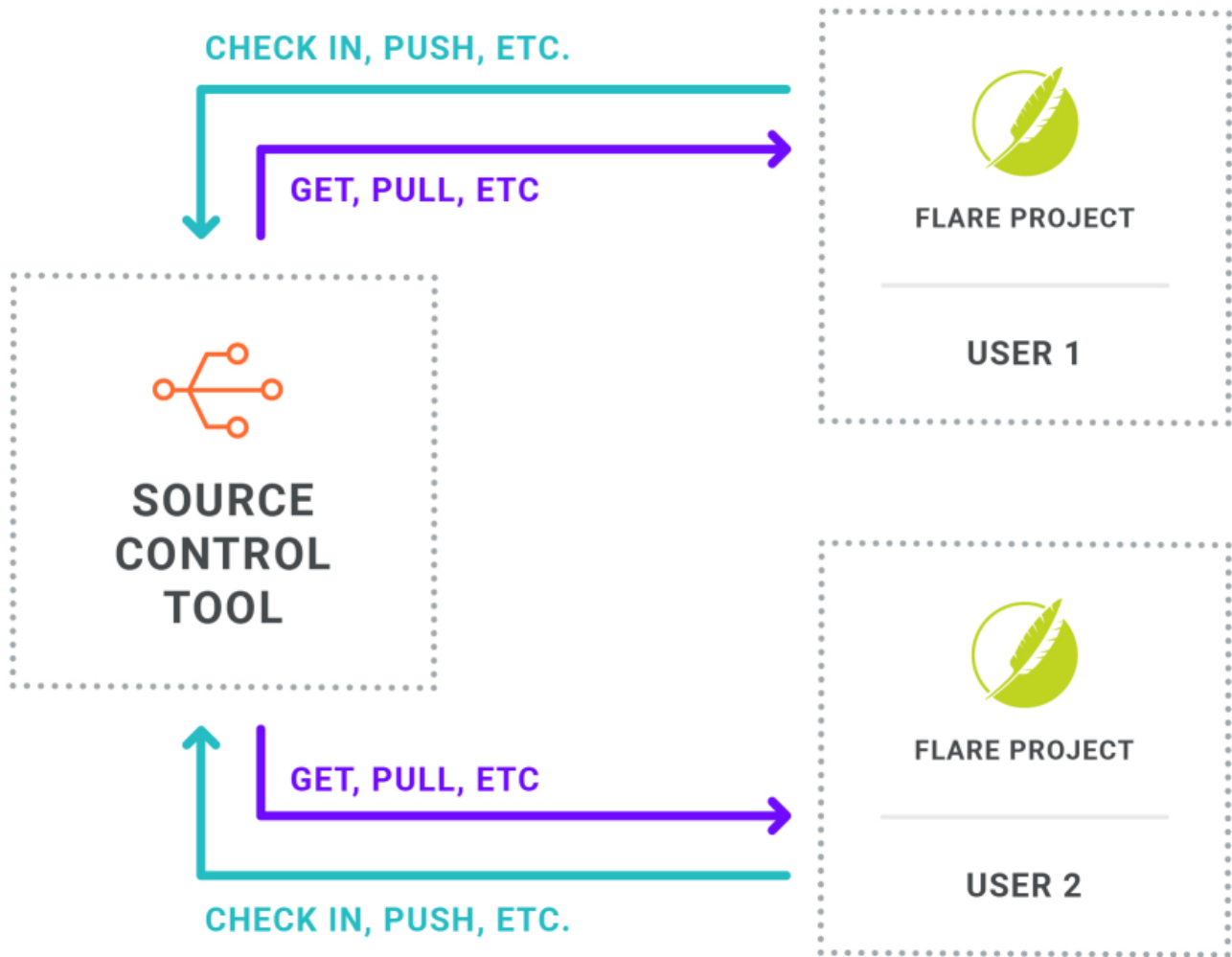
It might be tempting for some authoring teams to want to put a Flare project up on a network drive and simply let multiple authors open and work in it. However, this is not recommended. First, working on projects on a remote server, rather than locally on your computer, can result in lag time and poor performance. And perhaps more importantly, this type of work situation can result in conflicts that never get resolved (e.g., two writers work on the same file, and one writer's changes overwrite the other's). That is why a source control solution is highly recommended, especially when multiple writers are working in the same project.
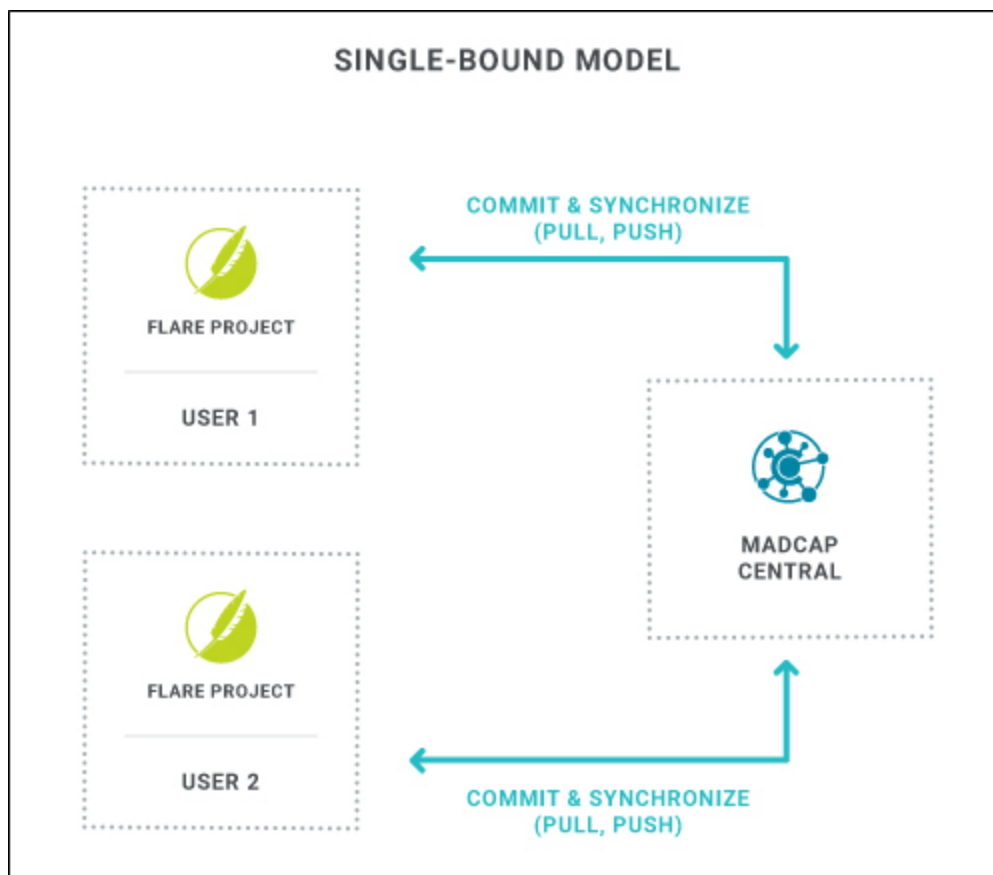
Source Control Tools

Source control isn't built in to Flare, but you can integrate a Flare project with any of the following third-party source control tools:
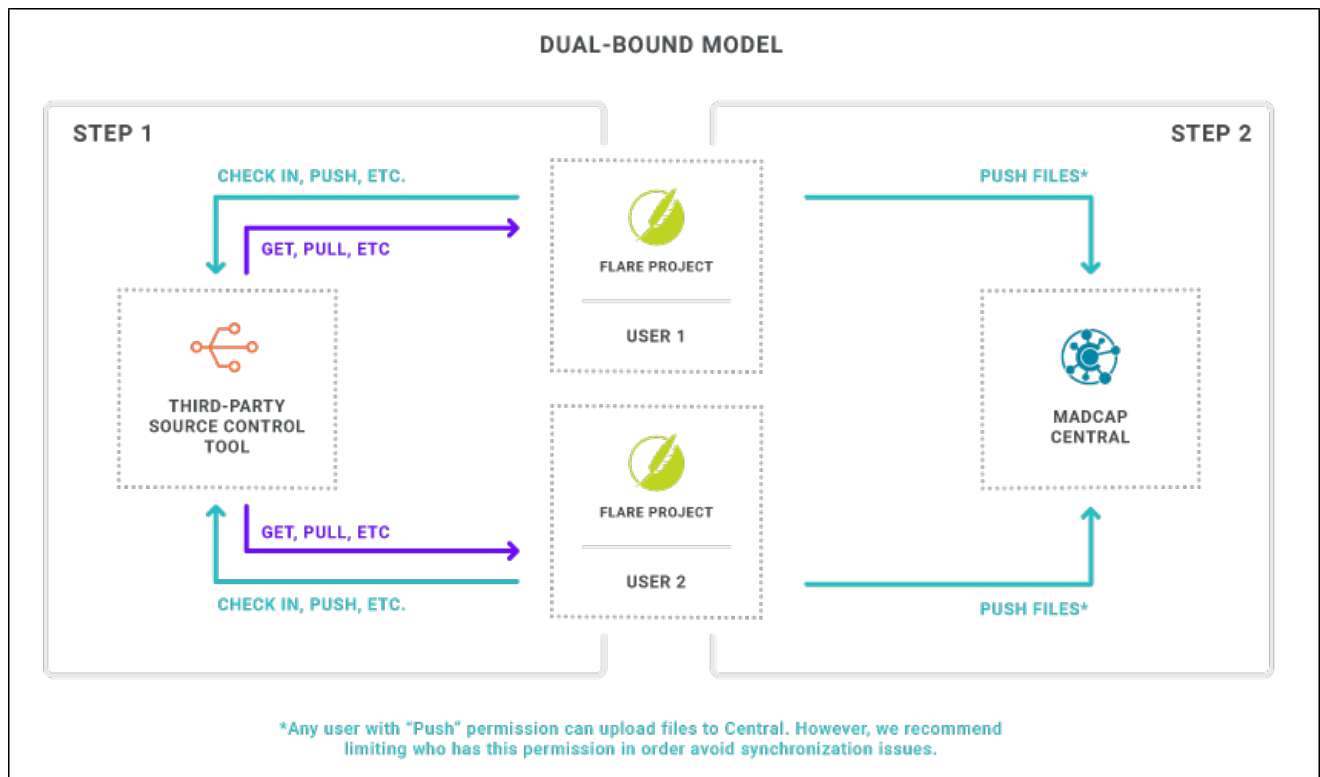
- Git

- Perforce Helix Core

- Apache Subversion

- Microsoft Team Foundation Server

As an alternative to these source control tools, authors can use MadCap Central, which employs Git behind the scenes.

Central offers a single-bound model, where your projects work directly with Central to manage source control.

Central also offers a dual-bound model. With this model, you can continue to use your third-party source control tool, as well as create a secondary binding to Central. Unless you have a really good reason for using the dual-bound setup, the single-bound model is recommended because it is easier and more streamlined.

DUAL-BOUND MODEL

STEP 1

CHECK IN, PUSH, ETC.

GET, PULL, ETC

THIRD-PARTY SOURCE CONTROL TOOL

GET, PULL, ETC

CHECK IN, PUSH, ETC.

FLARE PROJECT

USER 1

FLARE PROJECT

USER 2

STEP 2

PUSH FILES*

MADCAP CENTRAL

PUSH FILES*

*Any user with "Push" permission can upload files to Central. However, we recommend limiting who has this permission in order avoid synchronization issues.

Branching

Branching is a feature of some source control tools where you can work on different versions of the same project. It's a way of engaging in continuous parallel development, in which you can keep your new development separate from the finished documentation. This can be especially helpful when you are working with an Agile methodology. Flare's interface only supports Git branching, and it therefore also supports branching for projects bound to MadCap Central.
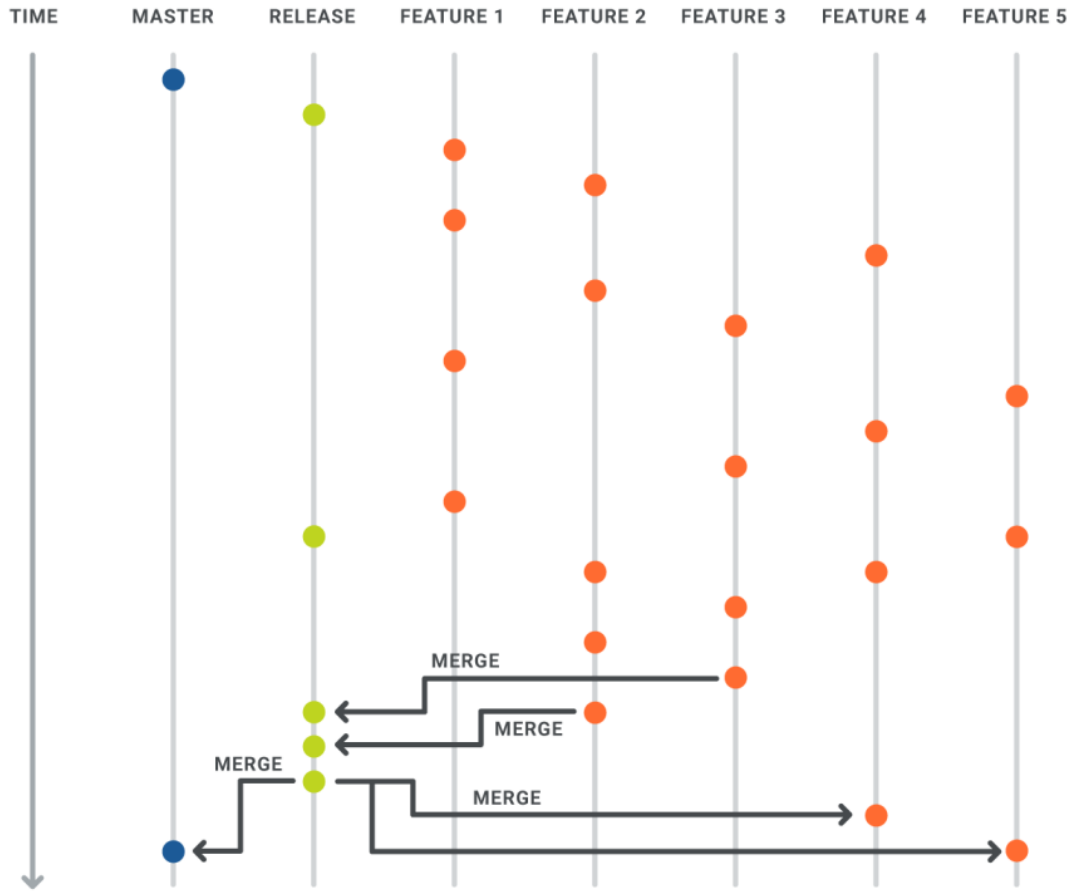
> ☆ **EXAMPLE** You might have a master branch, which is a version of your project the last time you published output.
>
> You also might have a release branch of your project, which is a version of your project for the next, upcoming publication of your output.
>
> Then you might have several feature branches, each one where you have a version of your project dedicated to a new feature you're working on. And when you're certain that a particular feature will be part of the next release, you merge it into your release branch, and finally into the master branch. Changes can also be merged back into ongoing feature branches so that they have the latest content from other branches.

FEATURE BRANCHES

TIME · MASTER · RELEASE · FEATURE 1 · FEATURE 2 · FEATURE 3 · FEATURE 4 · FEATURE 5

MERGE
MERGE
MERGE
MERGE

Pros and Cons of Source Control

Pros

Source control is simply the most efficient way for multiple writers to work on the same project. It allows multiple people to work simultaneously on the same files and merge their changes when needed. You can also use source control to resolve any file conflicts that you might have with other authors.

Even if you are a lone writer, using a source control solution provides a way to have a backup of your project on a server. This is important in case something happens to your computer and you lose all of your local files.

If you bind your project to Central using the single-bound model, the process is very simple with only a few clicks required.

Some source control tools (e.g., Team Foundation Server) have a check in/check out system that helps alleviate file conflicts.

Cons

Setup can be challenging for some source control tools when you bind them to a Flare project. You probably need to obtain information from your system administrator, and you must enter everything accurately in the Flare interface for the binding to work successfully.

There is a learning curve involved with source control solutions, especially when it comes to resolving file conflicts. Certain file conflicts can happen when writers are working on the same file. This can be frustrating and confusing at times if you are new to source control. This is why it is important to educate yourself on the particular source control tool you are using, and establish internal rules and guidelines for every member of your team to follow. This can help limit the conflicts that occur and make it less daunting when you need to resolve them.

Source control processes can slow down Flare projects as you work in them. A project that is not bound to source control typically runs faster than a project that is using source control.

What the MadCap Documentation Team Does

We use source control for all of our projects.

Shared

Our main Shared project is single-bound to MadCap Central (with Git behind the scenes). We rely heavily on Git's branching capabilities, employing a workflow system called "GitFlow," which makes our files and changes much easier to manage. In Flare, we perform most of the basic source control tasks (e.g., committing and synchronizing changes). For some more obscure source control tasks that cannot be done from the Flare interface, we occasionally use Visual Studio or Git Bash.

Documentation Bible

Our Documentation Bible project is also single-bound to MadCap Central. However, we don't require the use of branching like we do for our main Shared project. When authors need to work in the Documentation Bible project, they simply synchronize it with the cloned project on Central.

"Develop" Projects

Our "Develop" projects exist only as a workaround in order to create checklists for specific Git branches on MadCap Central. Checklists are one of the few features where branching is not yet supported on Central. So in order to create checklists associated with files for a specific branch, we need to create these child "Develop" projects. All of the edits in particular branch is first done in the main "Shared" project. Then, we use Global Project Linking to import changes for a specific branch into the child "Develop" project. So when we upload changes to Central from one of these child projects, we are uploading the master branch for that project, but the imported files are actually coming from specific branches via the parent "Shared" project.

> **NOTE** If you would like more details on the way that the MadCap Documentation Team uses source control—especially branching—you can download this PDF:
>
> ```
> https://docs.madcapsoftware.com/doc-team/Source-Control.pdf
> ```
>
> This PDF discusses concepts, reasons why we do things a certain way, procedures, daily tasks, and more.

▍Templates

A template is an existing project or file that serves as the basis for a new one, providing preset content, settings, or formatting. For more information see the online Help.

So after you create template files, you can store them anywhere on your computer or network. From there, you and other authors can access them when creating new files within Flare.
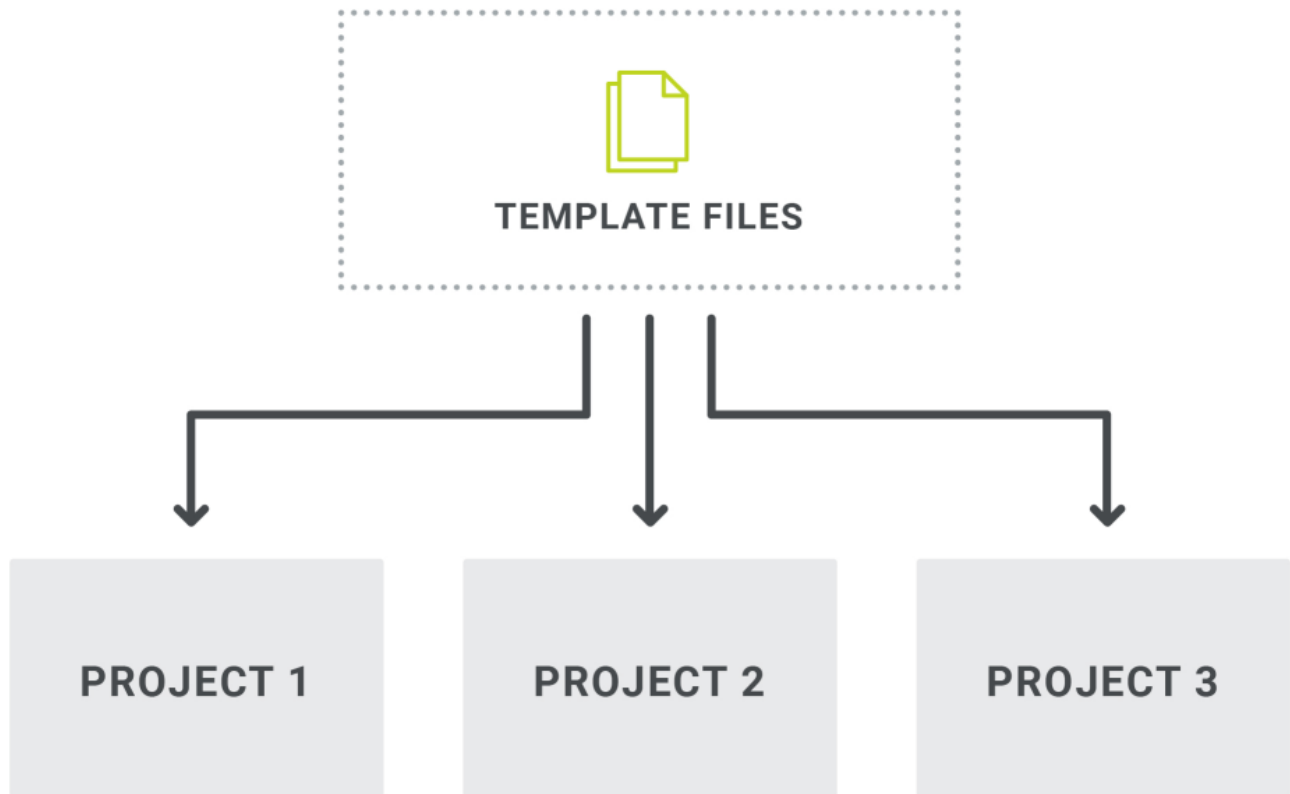
Pros and Cons

Pros

Creating your own templates can help with consistency in the structure of your topics and other files.

Templates are especially useful when you are working with a team of writers, which ensures everyone adheres to a particular structure or set of content rather than each person creating everything independently.

Cons

Over time, your standards, formatting, and content for templates might change, which means someone needs to keep the templates up to date.

If templates are changed now and then, those edits will not automatically be propagated to existing topics or other files. You would need to update existing files manually.

What the MadCap Documentation Team Does

We created skeleton versions of each of the types of topics we use (see "Types of Topics" on page 99):

- Concept
    - Major landing page
    - Minor landing page
    - General concept
- Task
- Reference
    - General reference
    - UI element
- Troubleshooting

These skeleton versions contain only the content that should always be used when a particular kind of topic is created—often some standard text and drop-downs in a certain order.

Since the vast majority of our files and content are stored in our main Shared project, we created a folder in the Content Explorer to store all of these topics templates, plus some snippet templates that are used with them. With source control, we know that each writer has access to the latest version of the templates, although they do not change often.
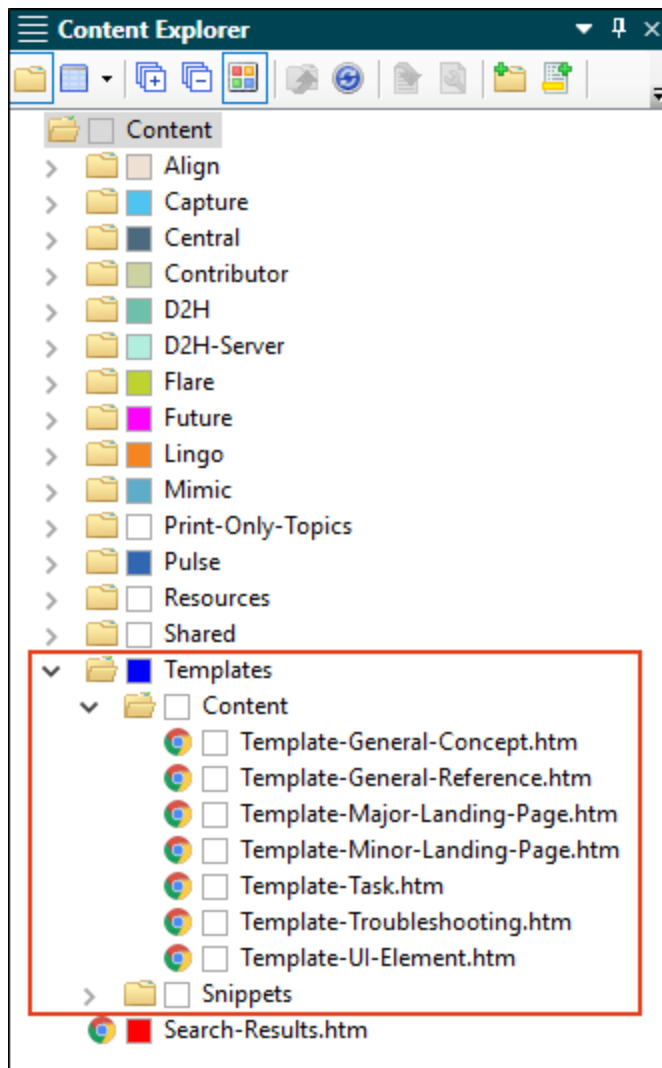
When writers create a new topic, they can choose one of these topic templates so that the necessary pieces are already in place.
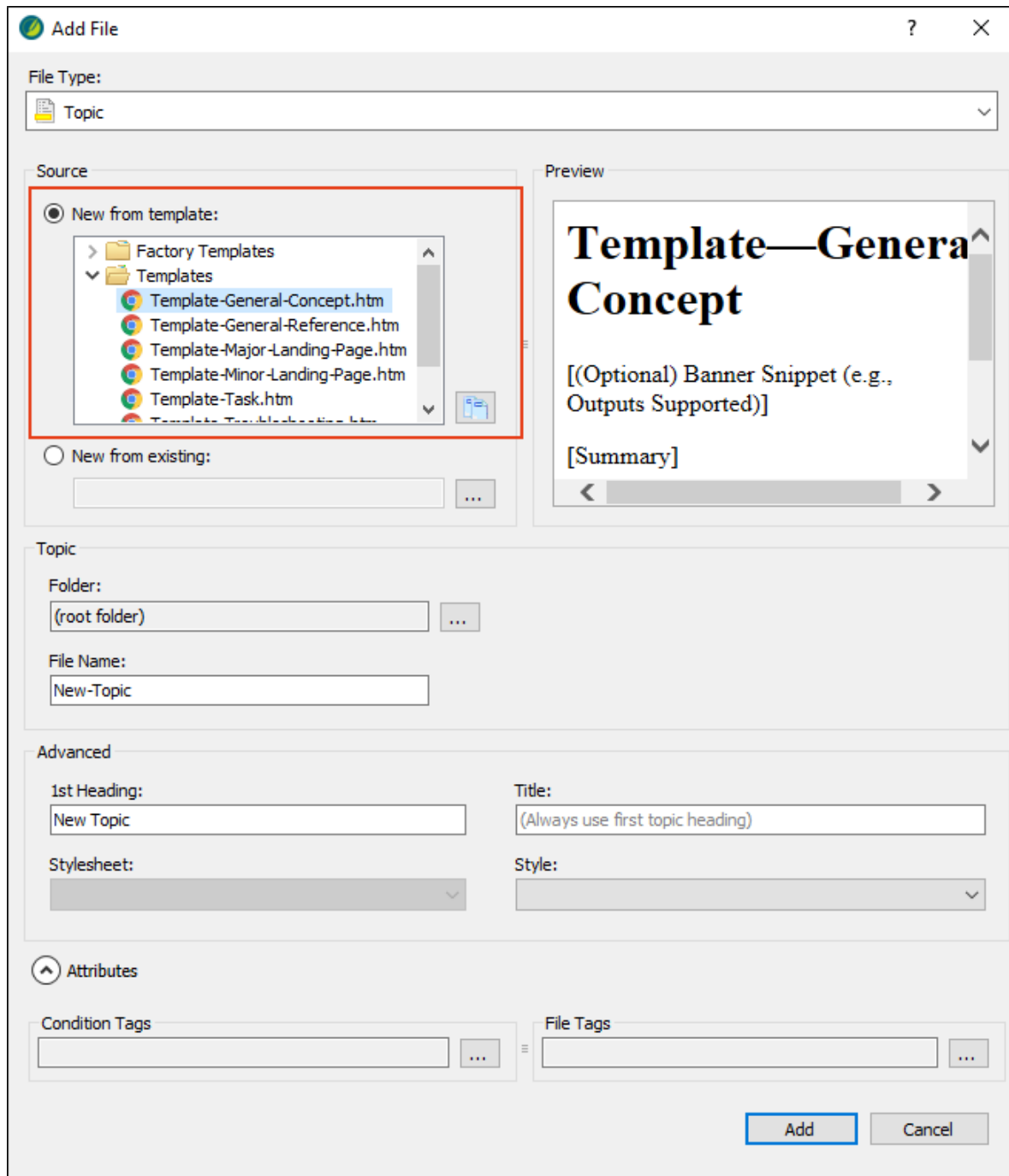
▮ Task Management

How do you keep track of all the work that you need to accomplish, both inside and outside of a Flare project? The good news is that you have lots of options. The bad news is that it's not always easy to decide which option is right for you.

Following are some of the common ways that authors use to management their workload.

Management Software Tools

Several tools designed to manage issues and tasks are available. This includes applications such as Jira, Trello, and many others.

Pros
Some of these tools are quite powerful and have some features that are quite useful. We are not going to attempt to compare any of these tools here. Just like any other software, these tools excel in some areas more than others.
If your company already uses a particular management tool, it's already available to you, and there might be training that's accessible.
Some of these applications can be inexpensive, or even free.

Cons
Some of these software applications can be expensive.
Depending on the tool, it can be difficult to learn to use.
There is no built-in integration with MadCap Flare. Therefore, the input is largely manual.

Microsoft Excel

If you are familiar with Excel, you might decide to create spreadsheets to track topic development and other kinds of work.

Pros

For many, Excel is readily available and familiar.

You can provide formulas to calculate information such as completion percentages for particular cells.

Cons

There can be somewhat of a learning curve for new users.

There is no built-in integration with MadCap Flare. Therefore, the input is largely manual.

Microsoft Word

You can create Word documents with tables to track information in a similar way that you might use Excel documents.

Pros

For many, Word is readily available and familiar.

Cons

You must create your own tables for tracking purposes, which can take some time.

Cons

There is no built-in integration with MadCap Flare. Therefore, the input is largely manual.

Old-Fashioned Methods

It's not high-tech at all, but some people choose to organize simply by writing on sticky notes, in notebooks, etc.

Pros

These methods are typically quick, easy, and inexpensive.

Cons

Everything is manual. There is no automation in any way.

You can easily lose the information.

There is no backup, so if you lose the information, it's gone for good.

Seriously? Sticky notes? Really?

MadCap Central

If you bind your projects to Central, you can take advantage of the following features for managing work. For more about each of these, see the Central online Help:

https://help.madcapsoftware.com/central

- **Tasks** These are high-level cards providing information about a particular task.
- **Checklists** These are just what they sound like—detailed checklists with rows and columns, where you can keep track of the progress for the minutiae of your work.

Pros
Within a task card, you can provide a name, description, priority level, associated project, relevant dates, assigned writers, and attachments.
Tasks can be managed in a Kanban format, as well as in a calendar view.
You can backlog tasks that need to be addressed in the future.
You archive tasks to remove them from the current workflow, but keep them for future reference if necessary.
Everything is in the cloud, so you don't need to worry about losing it.
You can create generic checklists and include any kind of information you want to track. Alternatively, you can create project file checklists, which automatically load Flare project files that you choose.
You can set multiple statuses (To Do, In Progress, Completed, N/A) in individual cells of a checklist.
Statistics for each checklist are automatically provided as you make changes.

Pros

You can create templates, which make it quicker to create new checklists that use the same format.

You can create widgets (graphical information objects) that provide snapshots of information about tasks and checklists.

Cons

You must be subscribed to Central to use these features.

What the MadCap Documentation Team Does

At one time, we used Excel spreadsheets and Word documents to keep track of all our work. However, we found that these methods were too manual and cumbersome for us. After tasks and checklists became available in Central, we began to use these features.

Tasks

For the most part, we create high-level tasks based on each new feature that is being developed by our programmers. Schedules can be somewhat unpredictable, so we don't focus much on entering start and due dates. But we do associate each task with a project and assign a writer.

Also, in the description for each task, we enter the following information:

- Feature (PBI) number
- Basic explanation of the feature
- Developer lead
- QA lead
- URL to the associated checklist

If it is determined that a particular feature will not be part of the upcoming release, but rather a later one, we move the associated task into the backlog.

Checklists

For each new product release cycle, there are three main types of checklists that we create:

- **Release To Do** This is a generic checklist with about 25 rows listing certain things that we do, usually in order, throughout each release cycle. The name of the checklist starts with the product name and version number, and it is in ALL CAPS to help it stand out from the many topic checklists that are also listed. We also have detailed notes for each row, which are especially helpful for a new writer. The columns are the names of the writers, who mark each item as complete (if applicable) throughout the course of the release.

- **Feature List** This is a generic checklist, where each row lists a feature that is part of the upcoming release. The name of the checklist starts with the product name and version number, and it is in ALL CAPS to help it stand out from the many topic checklists that are also listed. A note is included with each feature that provides the URL to the relevant topic checklist. The column are the names of the writers, who mark each item as complete (if applicable) as the feature is finished. There is another column called "Release" that we mark as complete when programmers have made enough progress on the feature to merge the code into their "release" branch (see "Source Control" on page 27). This tells us that we may safely merge the documentation for that feature into our "release" branch as well.
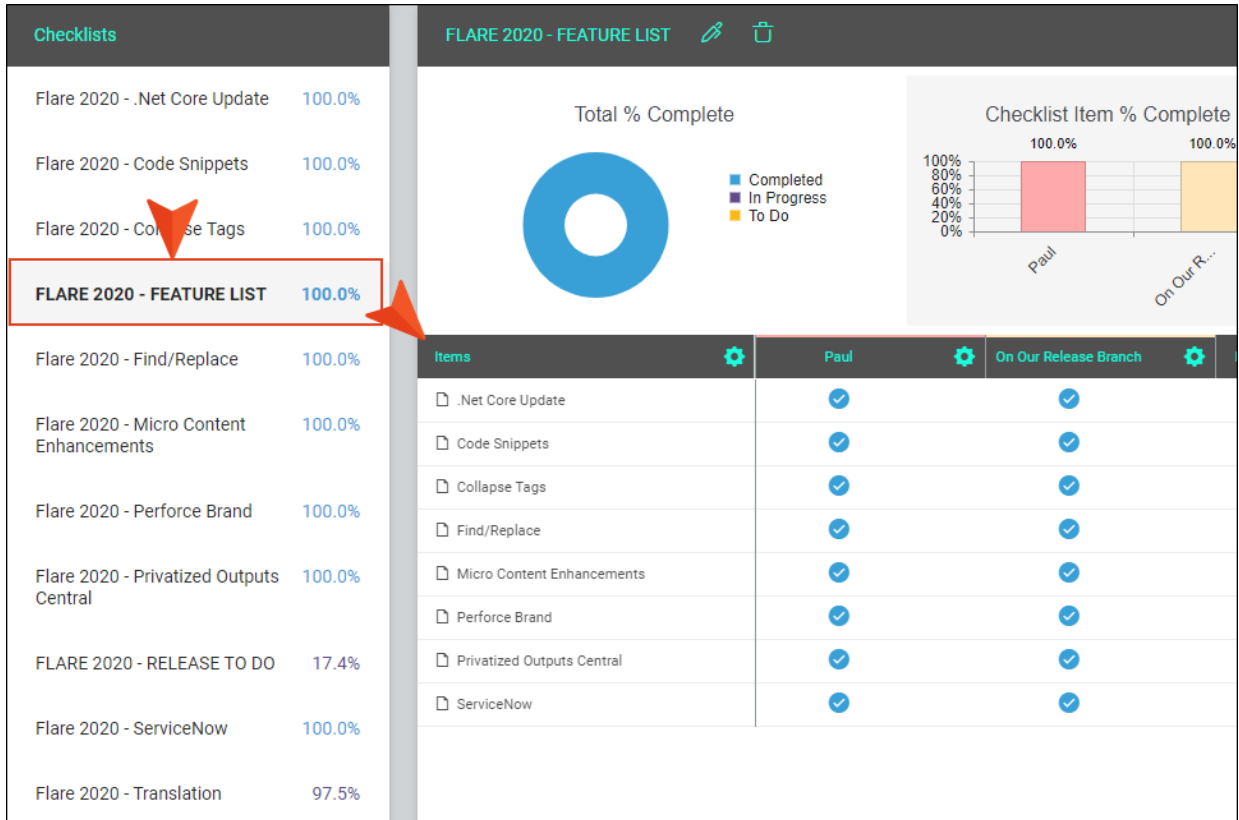
Essentially, this checklist is a consolidation of the tasks that were created for the upcoming release. We've found that is nice to have this "birds-eye" view of all the features intended for the next release. And if it turns out that a particular feature does not make it into the specified release, we edit the checklist, removing that row.

- **Topics** This is a project files checklist. It is the most detailed, and perhaps the most important, of our checklist types. For each feature, we create one of these topic checklists. The checklist name includes the product name and version number (or release date). Within the checklist, we choose the Flare topics that are new or affected by that feature. Columns indicate (in order, for the most part), the various things we want to ensure are completed for each topic (e.g., Edit, TOCs, Index, CSH ID, Reviewed). As needed, we enter notes for a particular topic to remind us of specific things that might need to be done (e.g., "Replace video link").


Because each of these types of checklists is used repeatedly over time, we have created checklist templates so that we don't need to manually re-enter all of the columns, rows, and notes each time we create a new checklist.

Importing External Files

As part of your overall architecture, you might import external files such as the following into your Flare projects:

- Microsoft Word
- Microsoft Excel
- HTML
- Atlassian Confluence
- Adobe FrameMaker
- DITA

This is necessary when you have legacy files that you want to leverage in your Flare projects. The legacy files are converted to topics and other Flare files. For more information see the online Help.
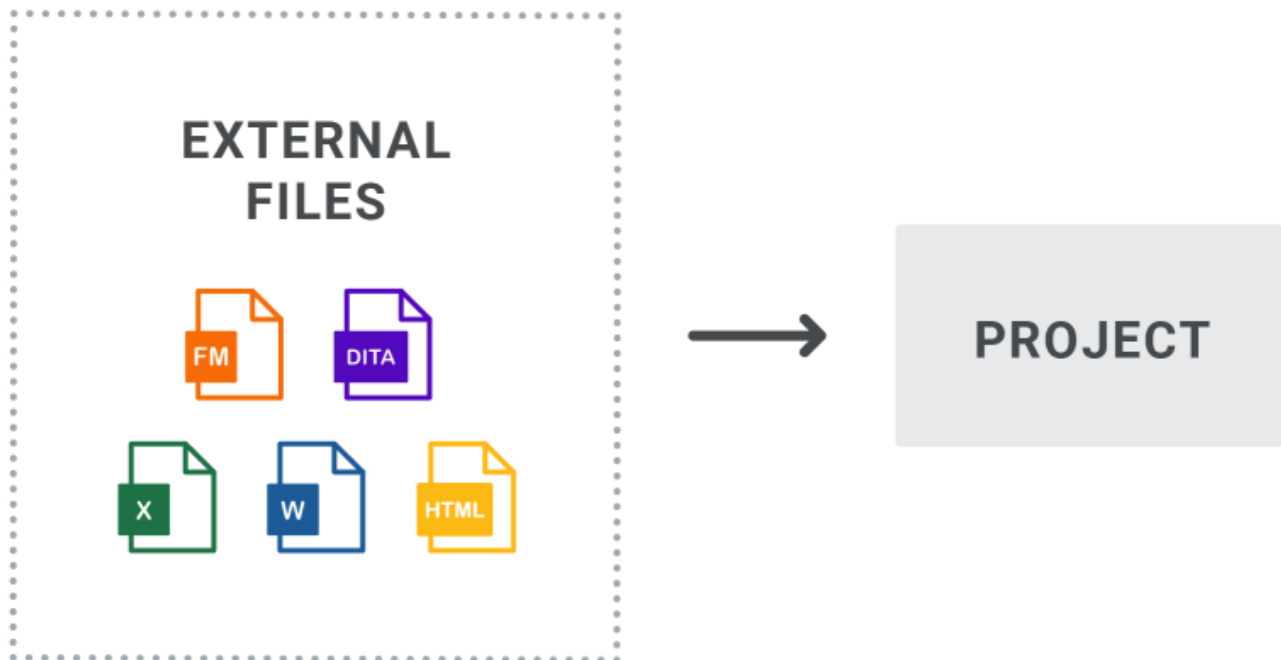
Basic Methods for Importing

When you import these types of files, you can use one of the following methods:

- **One-Time Import** After importing the files, you thereafter make any edits within Flare.

- **Easy Sync** You can maintain a connection between the legacy files and your Flare project. So after making future changes to the external files, you can reimport them back into Flare using this connection.

SharePoint and External Resources

Another way to leverage external files is to use the SharePoint or external resource integration features in Flare.

- **SharePoint** Flare supports integration with Microsoft SharePoint, including SharePoint Online (or SharePoint 365). From Flare you can access, edit, and synchronize SharePoint files. For more information see the online Help.

  SharePoint is like a scaled-down version of a source control tool, with the ability to check out and check in files. Many organizations use a source control tool to manage files that are part of products and will be shipped to customers, while using SharePoint mostly for internal files to be shared among colleagues.

- **External Resources** The External Resources window pane lets you select and maintain groups of external files that you want to share among Flare projects. The paths of these files are written to the registry so they will be available for all your Flare projects.For more information see the online Help.

  External resources can be virtually any local or network files to which you have access (e.g., images, PDF files, Flare project files). From the External Resources window pane, you can easily bring external files into a project (i.e., a copy of the file is added to your Flare project) and keep them synchronized with the source files through mappings.

  The external resources feature is ideal for shared files that you expect to change over time (e.g., logo images, PDFs, stylesheets), as opposed to, say, a template file that is simply copied into your project and changed only in that project.

SHAREPOINT FILES

EXTERNAL RESOURCE FILES

OUTPUT

PROJECT

OUTPUT
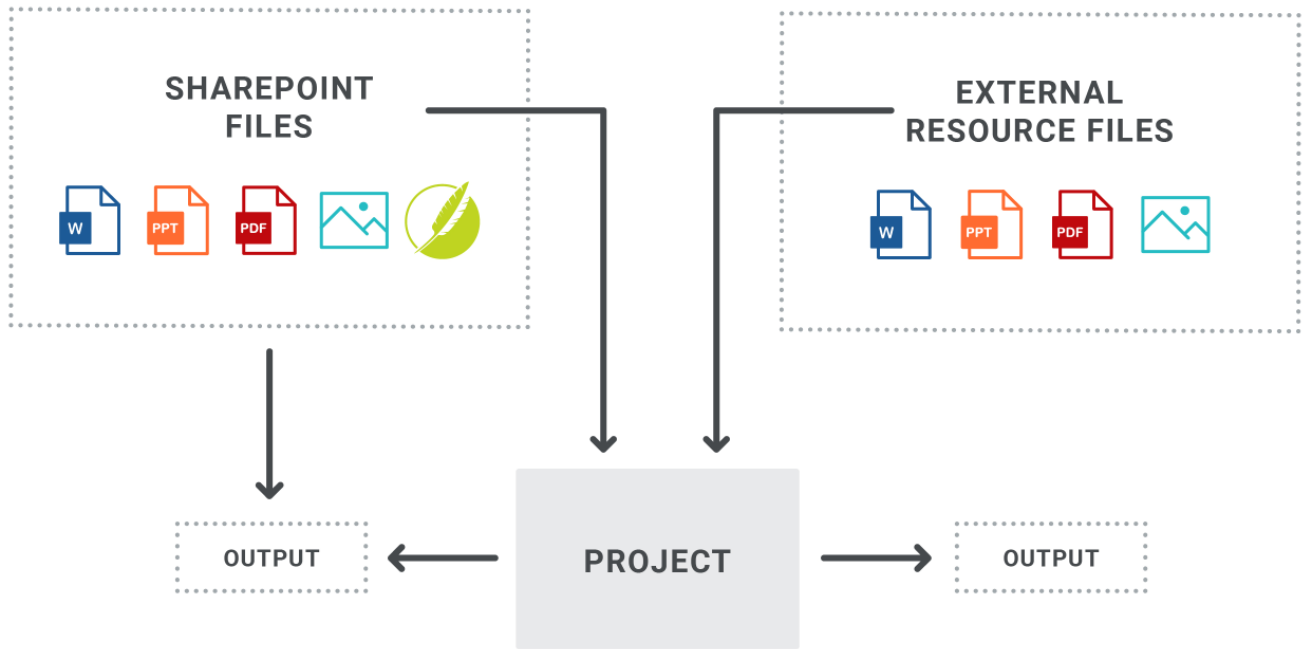
External Files Versus SharePoint/External Resources

Importing external files is different from using SharePoint or the external resources feature in the following ways.

Importing External Files
External files that are imported the regular way are converted to formats that can be edited in Flare.
The import file process requires you to reimport external files if they have been changed outside of Flare.

SharePoint or External Resources
The files that are brought into a Flare project via SharePoint or the external resources feature are not converted; they remain in their original format.
A synchronize feature lets you keep the SharePoint or external resource files up to date with their copies in the Flare project.

What the MadCap Documentation Team Does

All of our files are native to Flare, so we do not import external files. However, if we did import external files, we would use the following guidelines and tips, especially for Word or FrameMaker files:

- We would clean up the source files. For example, we would probably remove styling from those source files so that the import process is as smooth as possible.

- For Word imports, we likely would open the Word documents and apply "Private" field codes to the appropriate headings where new topics will be created after the import. This lets us control the file names that are produced as a result of the import. For more information see the online Help.

- For FrameMaker imports, we would follow the guidelines described in the topic "Moving from FrameMaker to Flare" in the online Help.

- We would do one-time imports instead of maintaining a connection between the source files and the Flare project. In the long run, editing the imported files within Flare is easier, better, and helps to avoid possible issues with reimports.

❙ Global Dictionaries

A global dictionary is a file that contains words and can be used for spell checking by any Flare project you open on your computer. This is a convenient way to ensure that all of your projects are using the same spellings for terms, rather than having each project rely on a different dictionary. For more information see the online Help.

You can store the global dictionary anywhere on your network and from the Options dialog (**File > Options**) in a project, you can point to that dictionary.

Not only can different authors use the global dictionary for spell checking, but they can also add new spellings to that dictionary.
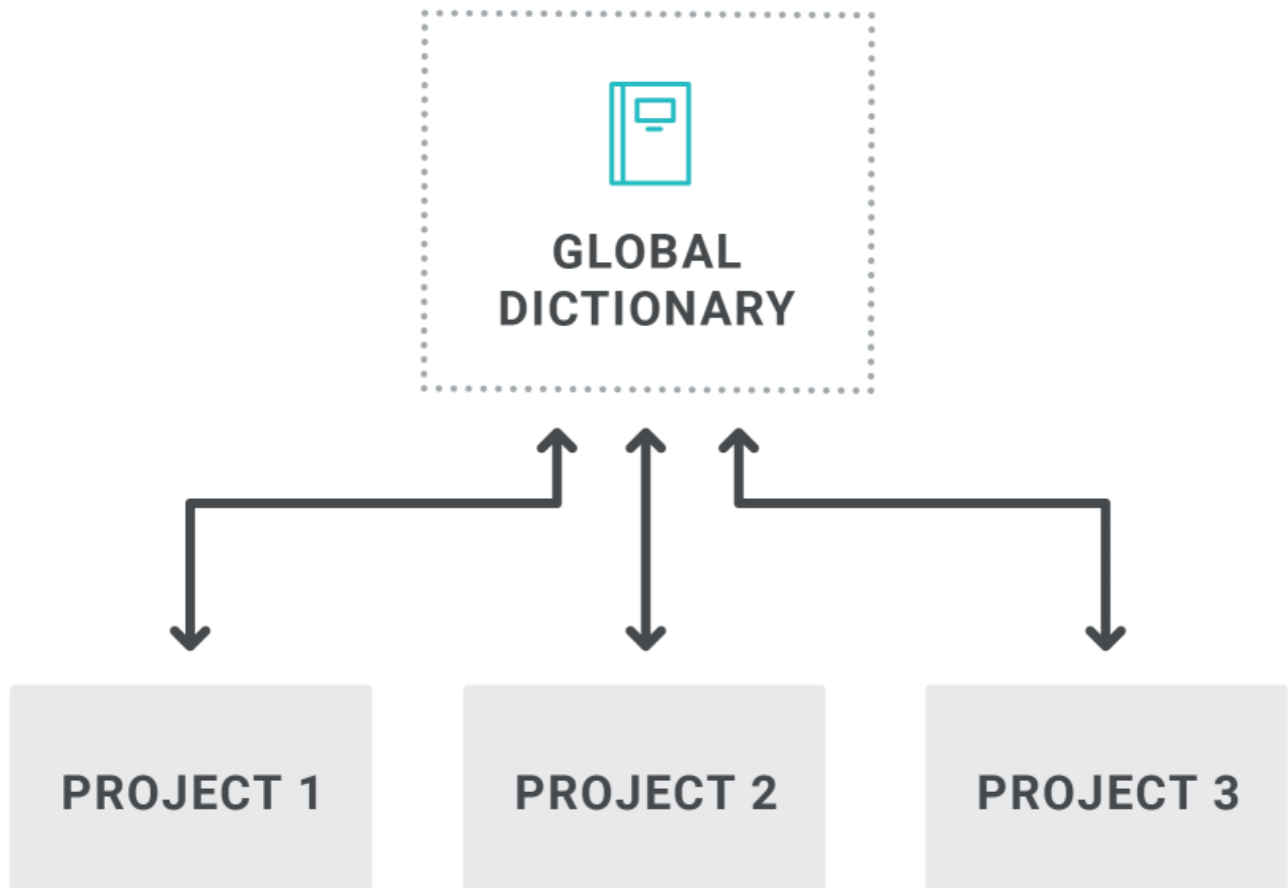
What the MadCap Documentation Team Does

We selected a network folder for a global dictionary, instead of using a local dictionary on each writer's computer. Each writer opens the Options dialog in Flare, selects the **Spelling** tab, and chooses that global dictionary in the custom location. If any writer adds a spelling to the global dictionary, that spelling becomes available to all writers.

Images and Videos

When it comes to image and video files, you will probably use a tool outside of Flare to create them (although there is some functionality in Flare's Insert ribbon for capturing images). Then you can insert those files into Flare projects or enter links to them.

Images

After creating image files, they are added to your Flare project and inserted by reference into content files. You can use MadCap Capture, or you can use a third-party image tool. Because of its tight integration with Flare, there are definite benefits to using Central.

For more information see the online Help.

MadCap Capture

Important Points

- Each image file created using Capture includes a .props file next to it. This lets you make your changes in Capture, with the image automatically being updated in Flare. There is no need to reimport the image.
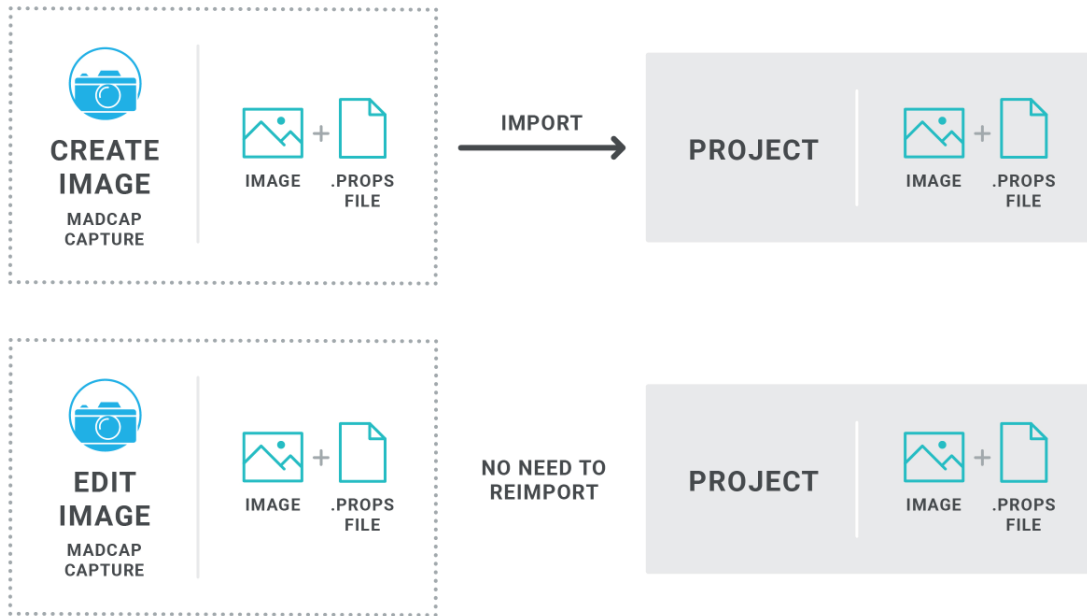
- You can take advantage of global profiles (settings for new images) and palettes (objects, such as callouts and arrows). This means that you can store and maintain these special files somewhere on your network, where anyone with Capture can use them. Also, this helps maintain consistency of images across Flare projects, especially when you have a team of writers.
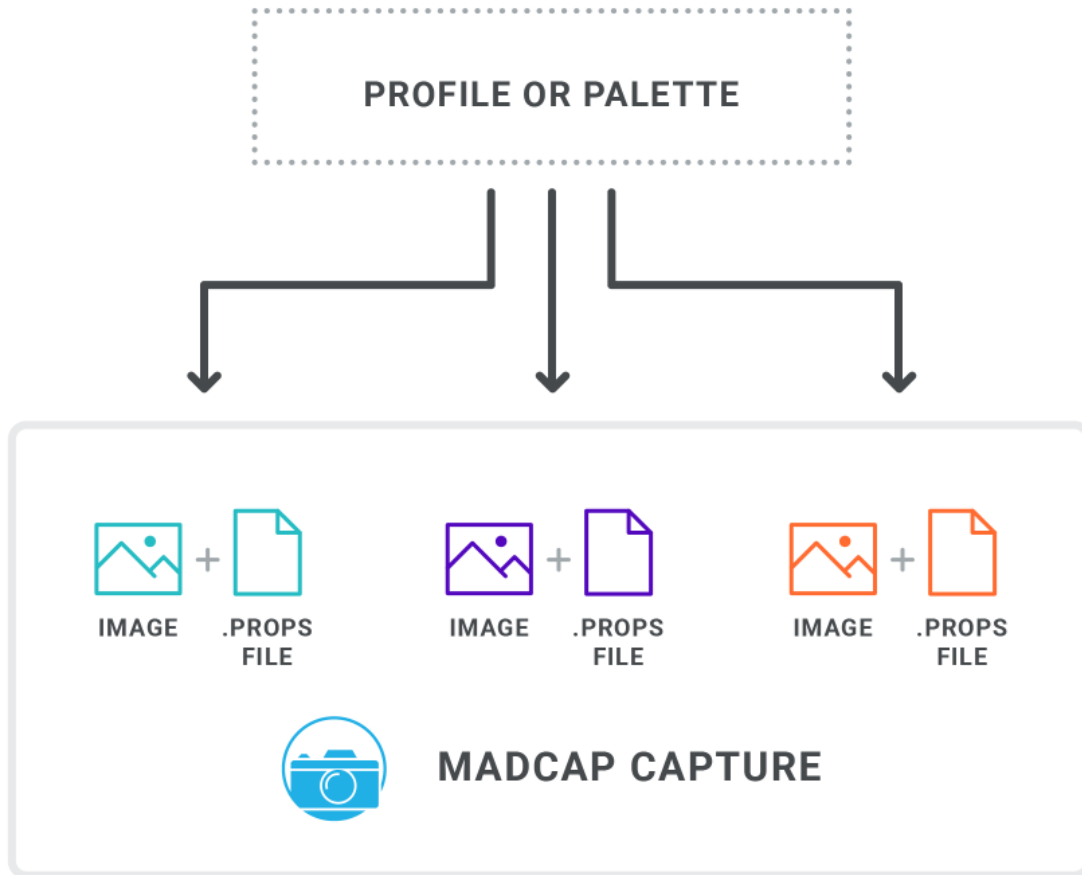
- There are probably fewer editing features in Capture than you will find in some established third-party tools. Although you can edit images in many ways within Capture, the primary purpose of Capture is not to provide advanced editing capabilities. Instead, its main purpose is to integrate tightly with Flare projects.

Third-Party Tools

- You will need to re-add each image file to your Flare project whenever you make changes to it.
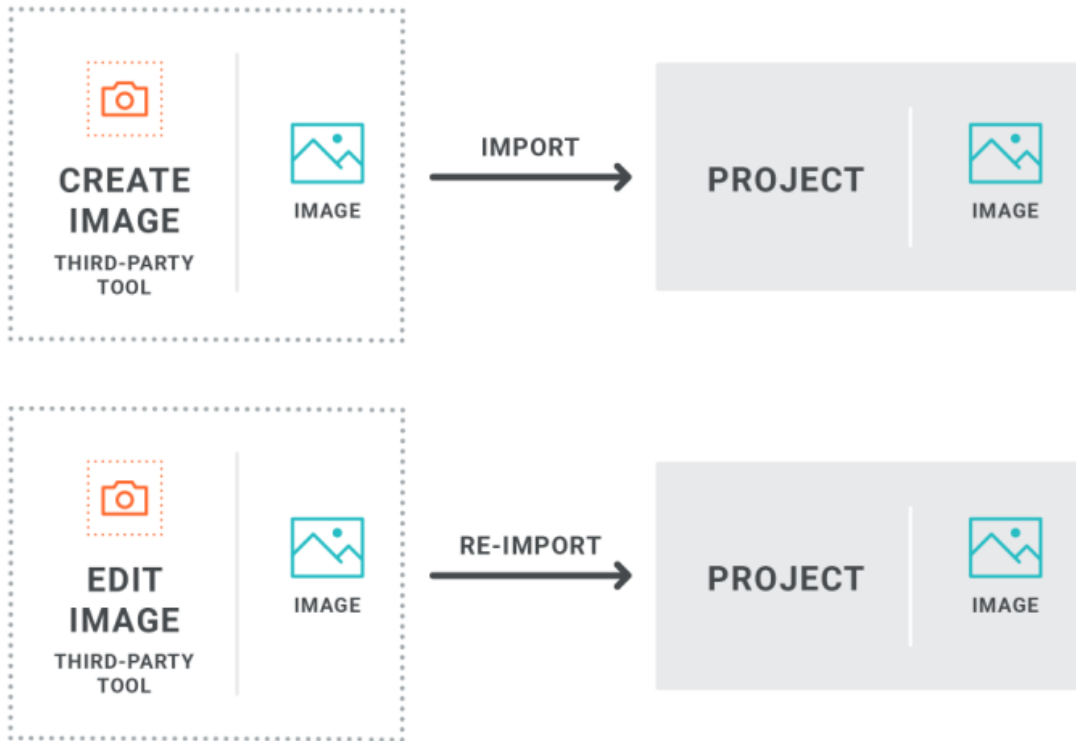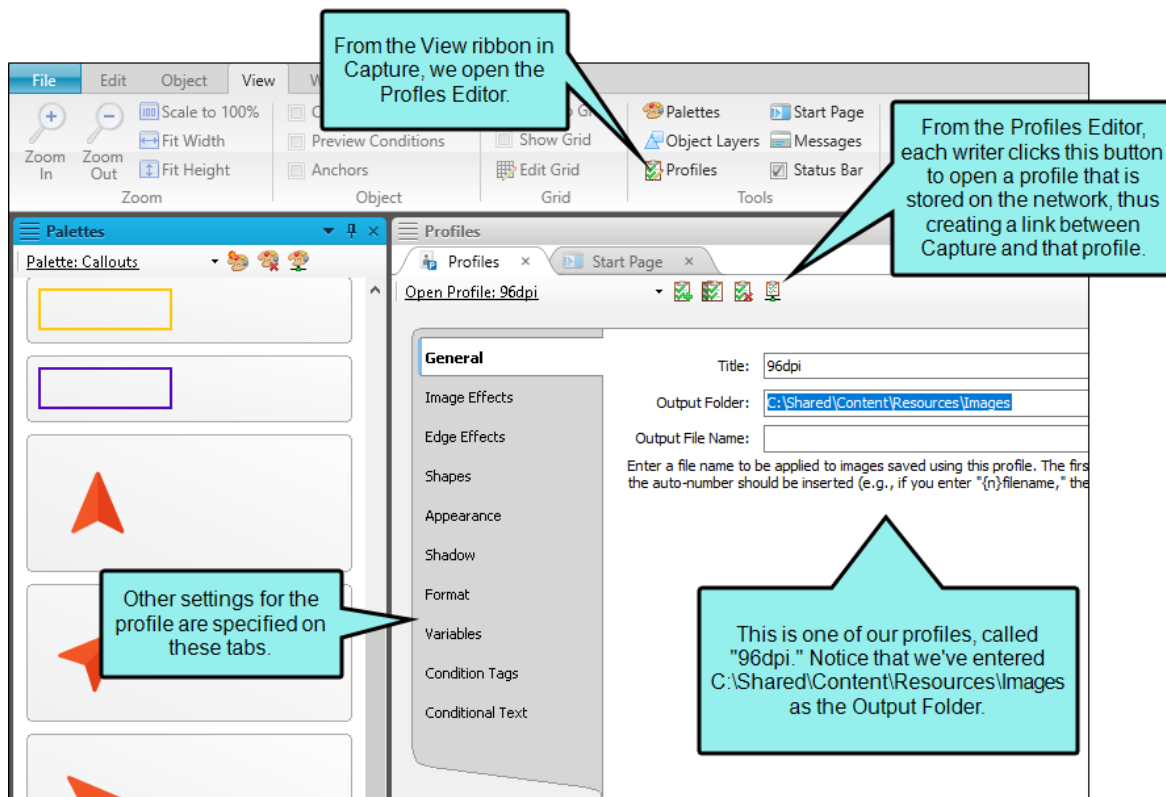

- Third-party tools may or may not have features that allow you to apply settings in advance (i.e., profiles) or maintain a library of custom objects to include in images (i.e., palettes).

- Many third-party tools have some very powerful editing features. If making slick, professional-looking images is a priority, you might want to use one of these applications.
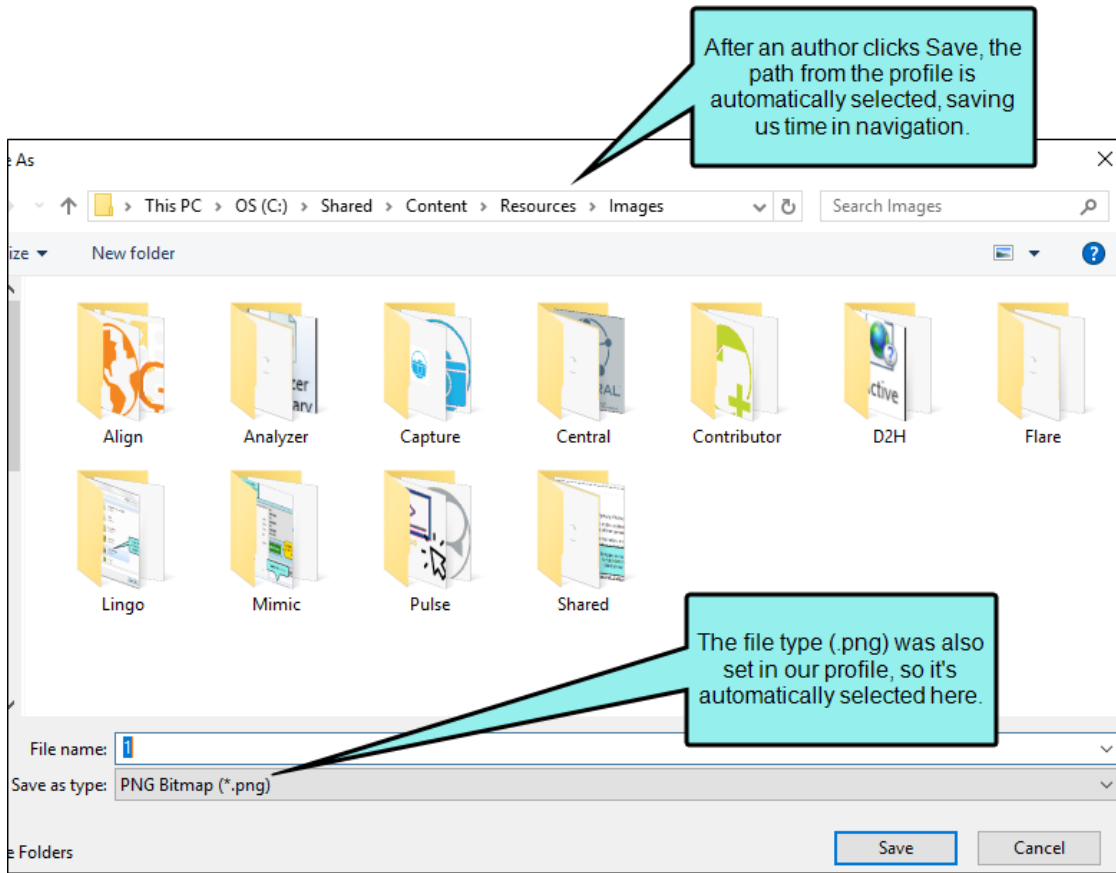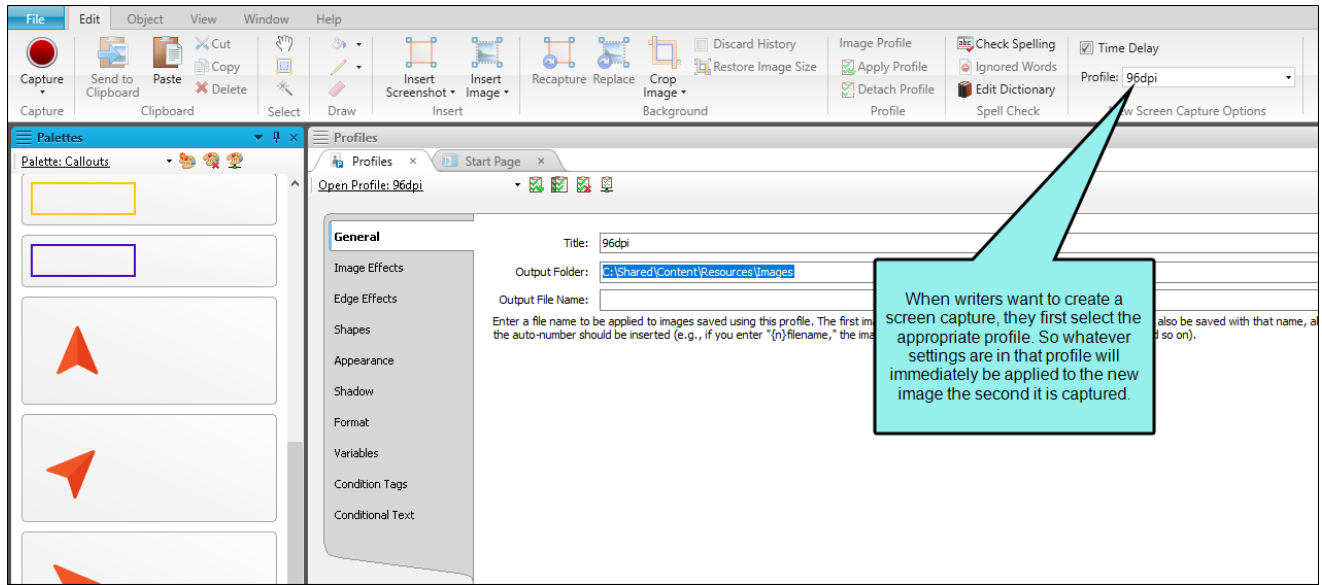
What the MadCap Documentation Team Does

We use MadCap Capture to create and edit images because of its single-sourcing capabilities and tight integration with Flare.
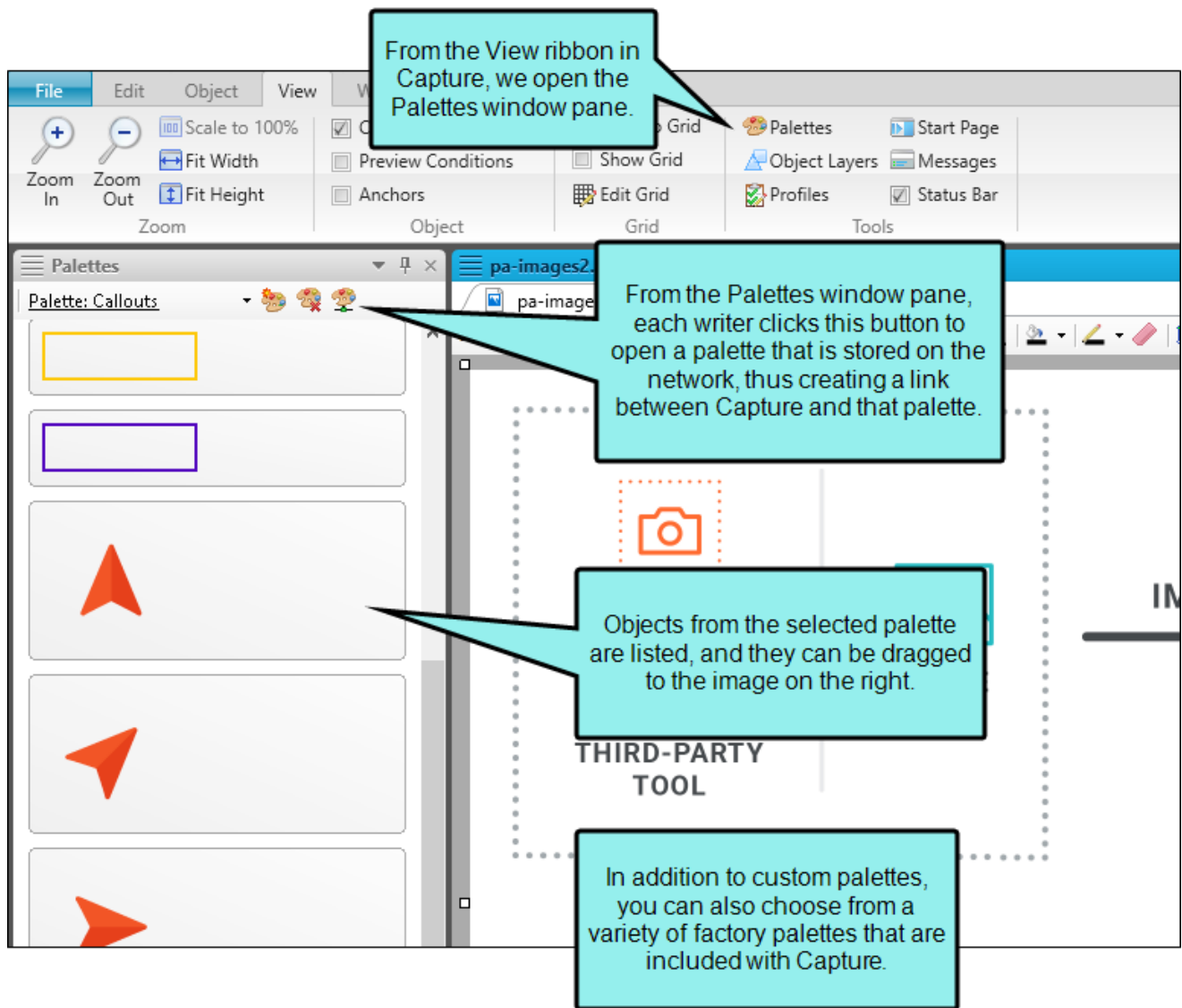
First, we created a couple of image profiles in Capture. Both profiles have identical settings (e.g., 96 DPI), except that one profile places a thin border around new screen captures and the other doesn't. Then we placed these profiles in a folder on our network. Each writer adds a link to these profiles in Capture.

One of the most important settings in the profiles is the output folder, which for us, is C:\Shared\Content\Resources\Images. This simply means that when a writer uses one of these profiles for a screenshot and clicks **Save**, the dialog immediately goes to this location. This saves time because we never have to navigate to the Images folder in our main Shared project (the most frequent place where we save images). This is also one reason that all writers store Flare projects in the root C:\ folder; the output folder setting in the profile is the same for everyone.

When writers want to create a screen capture, they first select the appropriate profile. So whatever settings are in that profile will immediately be applied to the new image the second it is captured.

After an author clicks Save, the path from the profile is automatically selected, saving us time in navigation.

The file type (.png) was also set in our profile, so it's automatically selected here.

We also created a palette in Capture that contains objects (e.g., callouts, arrows, rectangles) that we frequently add to images. By placing these objects in a palette and storing them in a network folder, we are assured that each writer is using the same objects and maintaining consistency.

From the View ribbon in Capture, we open the Palettes window pane.

From the Palettes window pane, each writer clicks this button to open a palette that is stored on the network, thus creating a link between Capture and that palette.

Objects from the selected palette are listed, and they can be dragged to the image on the right.

In addition to custom palettes, you can also choose from a variety of factory palettes that are included with Capture.

When profiles and palette objects are used in an image, that information is placed in the .props file next to the image file. This means that when we want to make changes to an image, we simply right-click it in Flare and select **Edit With MadCap Capture**.
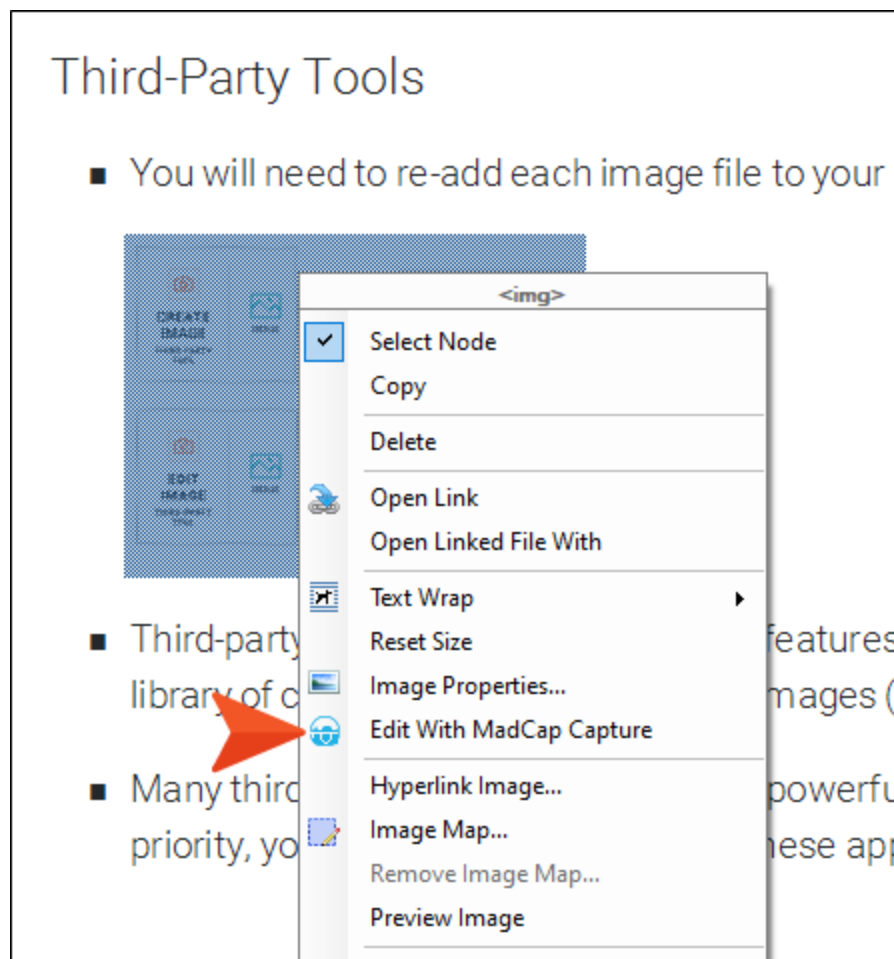

The image opens in Capture, and we can make any changes to it that we want, including adjusting the objects. As soon as we click **Save**, the image is updated in Flare. In other words, we don't need to waste time reimporting the image into the project.

Videos

After creating video files, you can add those files to your Flare project and then insert them into content files, such as topics or snippets. For more information see the online Help.

Alternatively, you can point to videos that you've uploaded to websites such as YouTube and Vimeo.
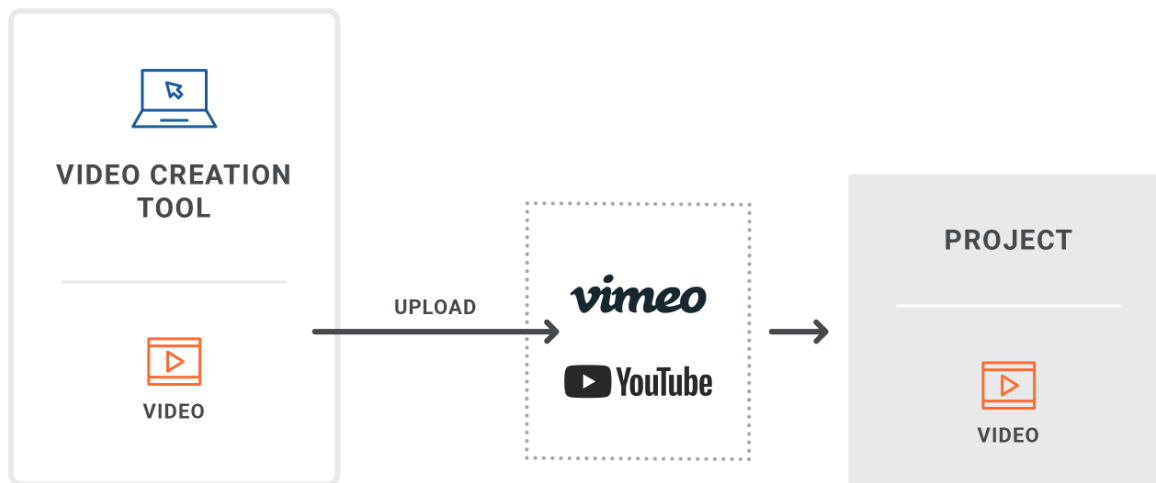
There are several video creation software tools on the market that you can use to produce your movies. If you use MadCap Mimic, you have the added benefit of being able to use the same palettes that are leveraged in Capture.
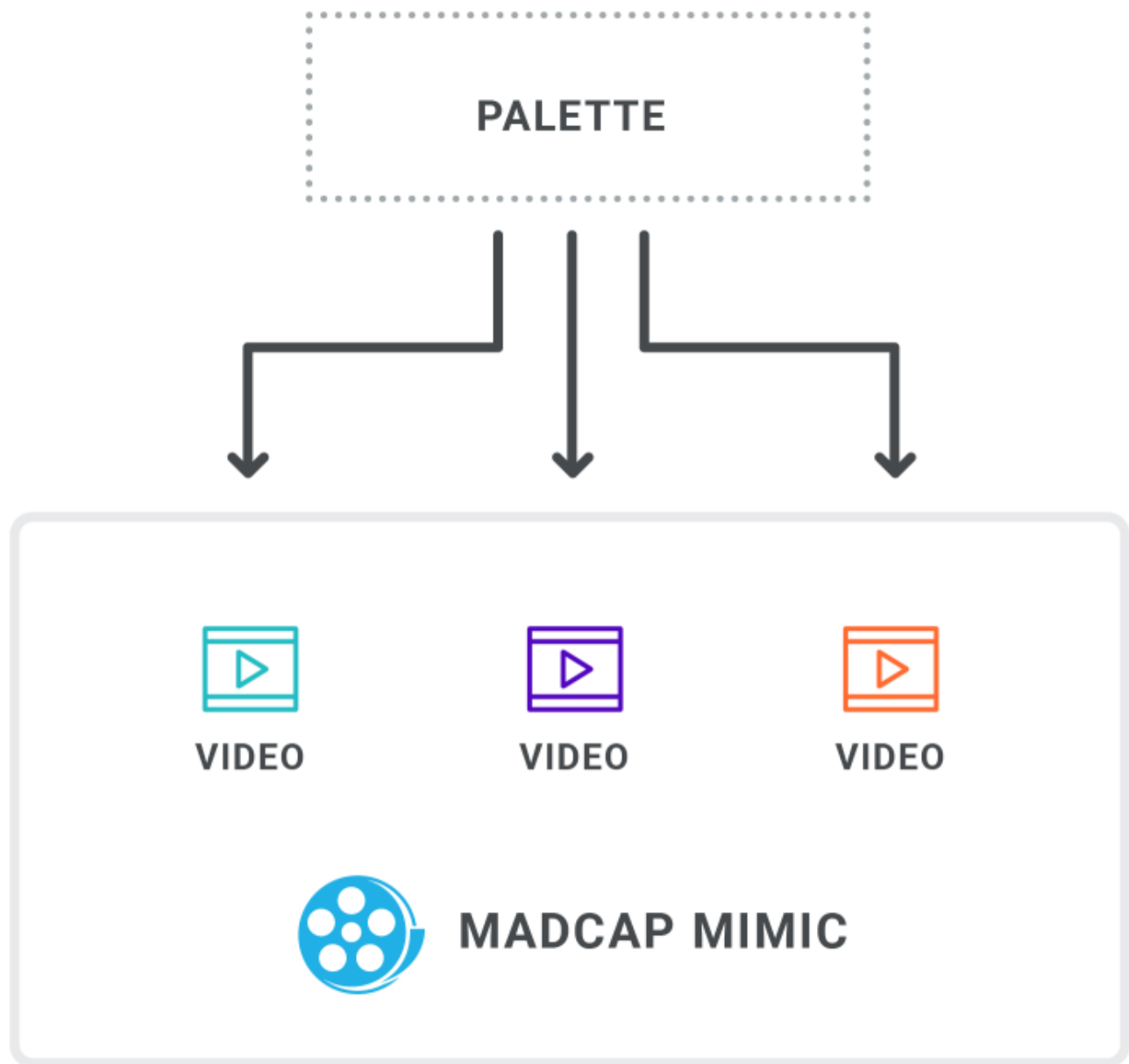
▎Analysis and Reporting

You can use various tools to analyze project source files for issues, and view user activity on output.

Analysis on Source Files

No setup is needed for analysis of source files. You simply run scans on your project.

Analysis Ribbon

From the Analysis ribbon in Flare, you can scan files and run reports to discover a wide variety of information. This includes broken links or bookmarks, files with changes, topics not in a table of contents (TOC), used meta tags, and more.
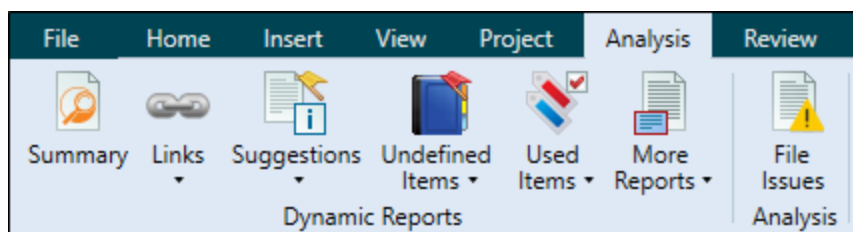

Reports

From the Reports folder in the Project Organizer, you can generate custom reports based on information contained in your project. In addition, you can design the look and feel of reports, save them for future access, and open them in a browser window (where you can print them).
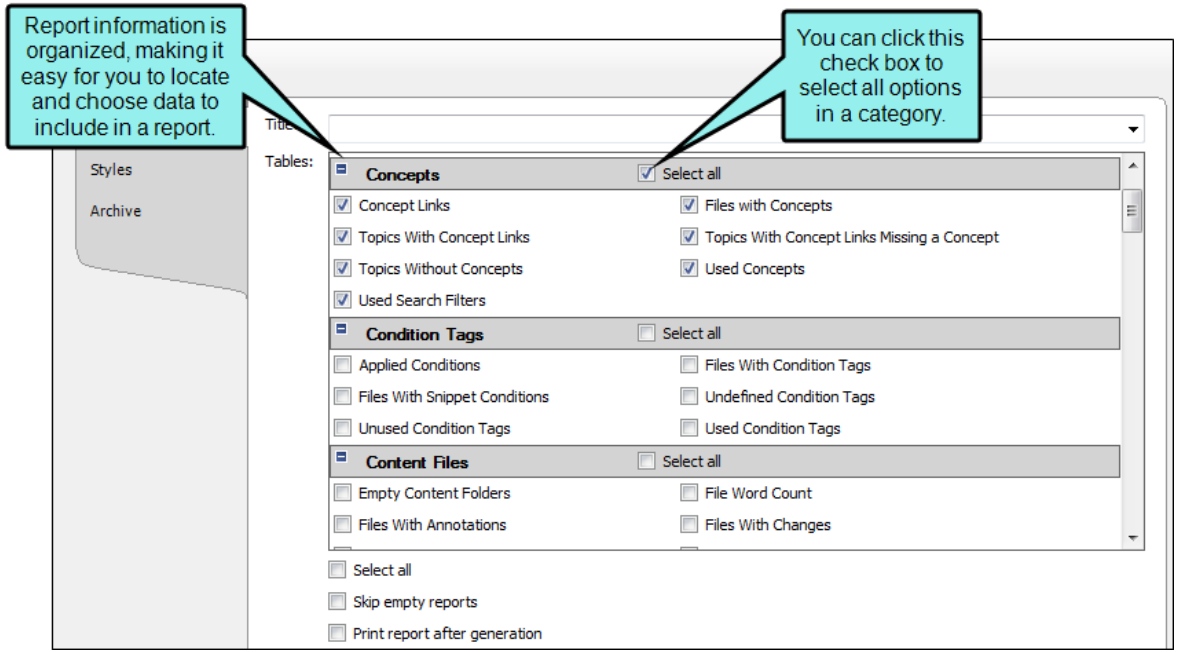
Text Analysis

From the Tools ribbon in Flare, you can run text analysis on content files for readability, average sentence length, and more. For more information see the online Help.
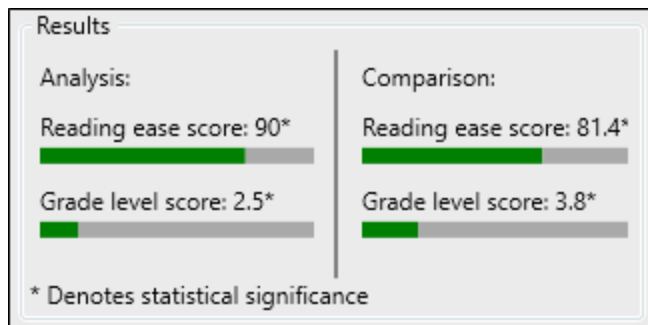
Analysis on Output Files

By collecting information about how people are using your output, you can more effectively make decisions to improve the content.

MadCap Central Analytics

If you have a MadCap Central license, you can view analytics on published Flare HTML5 output. This includes search phrases used, search phrases with no results, topics viewed, context-sensitive Help calls, and demographic statistics (browsers and operating systems). For more information see the online Help.
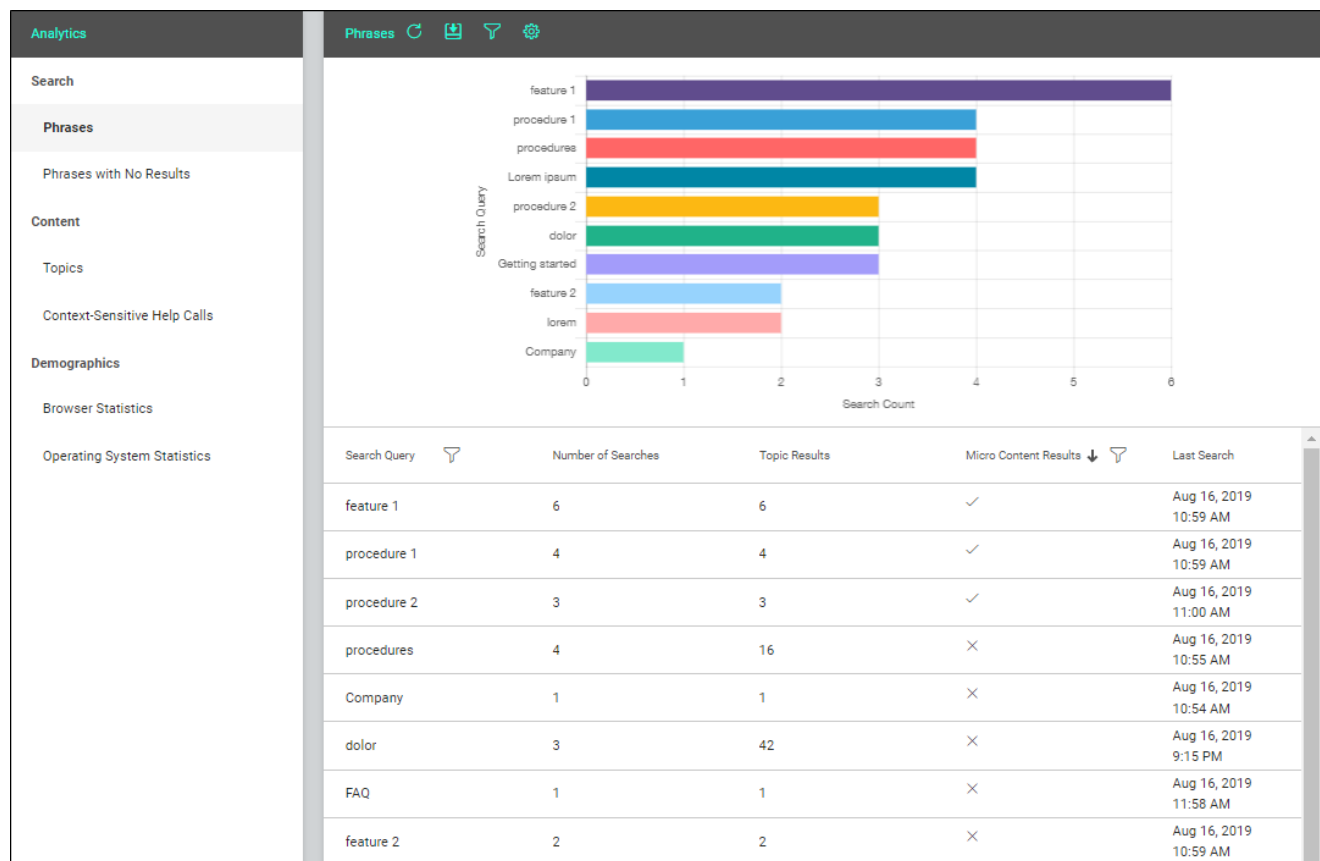

The process is quite simple. You create an analytics key in a couple of steps, associating it with a target. Then after building and publishing your output, you use the Analytics page on Central to view user activity on the output as it occurs.

Central analytics works on Flare projects uploaded to Central, or you can host output on your own servers. If you host the output outside of Central, you still need to use Central with a key to view the analytics data. Also, the server where the output is hosted must be able to communicate with Central (e.g., not be behind a firewall).

MadCap Content Services

The Content Services Portal is used to manage search indexes on your Elasticsearch output. You can review and manage search indexes by topic, micro content, user searches, and global searches. For more information see the online Help.
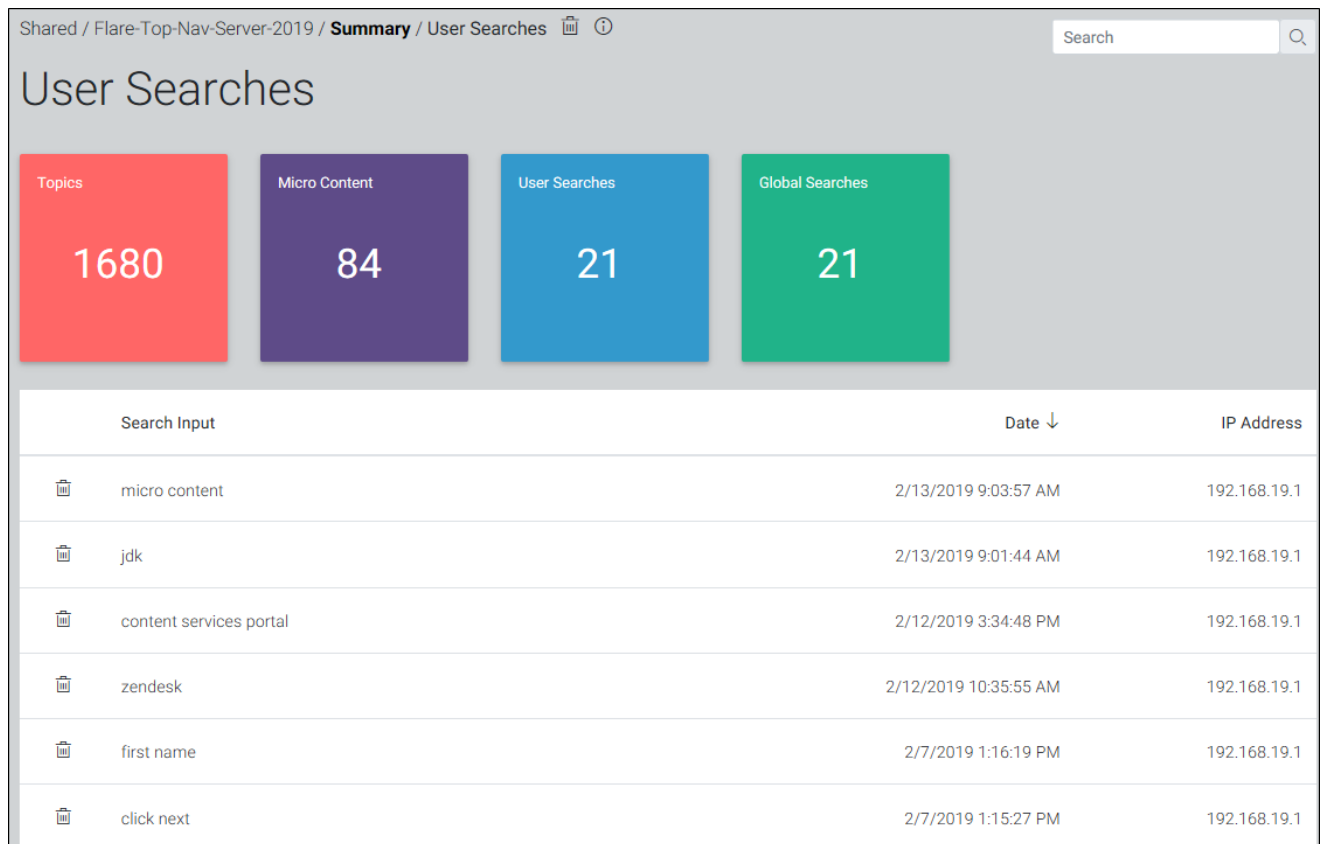
Google Analytics

You might decide to use Google Analytics (or another third-party tool) for reporting activity on your output. If you have a Google Analytics account, you can obtain a tracking code, which you can then add to your template page(s) in a Flare project. After publishing the output, you can use the Google Analytics website to see various reports about your output.

Syndicate Analytics

If you publish output files to a Xyleme Syndicate destination, you can use the analytics that are built-in to the application. Since Syndicate was designed primarily for learning and development (although it also hosts technical documentation), its analytics centers on tracking user interactions with content. It can track how users consume content such as individual topics or videos. For eLearning it can manage millions of records in its built-in learning record store (LRS). See the Syndicate documentation.

> 📝 **NOTE** If you are interested in learning more about Syndicate, contact MadCap Sales. They can provide information about the product, demos, review purchase options, and set up Syndicate accounts.

What the MadCap Documentation Team Does

We use analysis on both our source and output files.

Analysis on Source Files

When we get close to a product release, we scan the source files in our main "Shared" Flare project to discover various information about our source files, including:

- **Critical Issues** These are problems that must be fixed, such as broken links or bookmarks.

- **Files With Annotations** We occasionally insert annotations in topics and snippets as reminders to do something before a topic is considered complete. Scanning the project for files with annotations lets us quickly locate these instances and take action if necessary.

- **Topics Without Concepts** We use concept keywords in order to create search filter sets, as well as boost search results for certain topics. Therefore, we run this scan to make sure all necessary topics contain concept keywords.

- **Topics Not In Index** We do not use an index for online output, but we do for some PDF targets. Also, index keywords can help boost search results for topics. So we run this scan to make sure we have inserted index keywords where necessary.

- **Unused Images** We periodically run this scan to find obsolete images that can be removed from the project.

- **Used Meta Tags** We use meta tags extensively in our project, so we sometimes run this scan to make sure we've applied meta tags everywhere that is needed.

There are many other project analysis scans that you can run, but the ones listed above are those that we usually focus on.

Occasionally, we run text analysis on certain files, especially more complex ones, to make sure their readability level is acceptable.

Analysis on Output Files

Our outputs are integrated with MadCap Central. We log in to our Central license and use the Analytics page to view reports. These reports help us determine if we need to make changes.

We pay particular attention to the reports on search phrases. The "Phrases" report tells us which terms users are using the most to look for information. We can also easily see if any micro content yet exists for a phrase, or whether we should create it.

The "Phrases with No Results" report lets us know if we should create some new content or synonyms in Flare so that frequently searched phrases produce results in the future.

❙ Topic Reviews

Most authors require some kind of a process for others to review their Flare documentation, as well as to contribute new content when necessary. There are several ways you can do this, depending on your resources and knowledge of your reviewers and contributors.

Following are the most common methods, although you might use another solution not listed here.

Flare to Central—Peer and SME Reviews

You can use the connection between Flare projects and MadCap Central for topic reviews. Flare supports review packages (i.e., bundling files such as topics and snippets) and lets you send those packages to Central to be reviewed by others. After editing and annotating the files in Central's Review Editor, the reviewers submit the finished files, sending them back to your inbox in Flare. You can then accept or reject their changes and accept the file, replacing the original source file. For more information see the online Help.
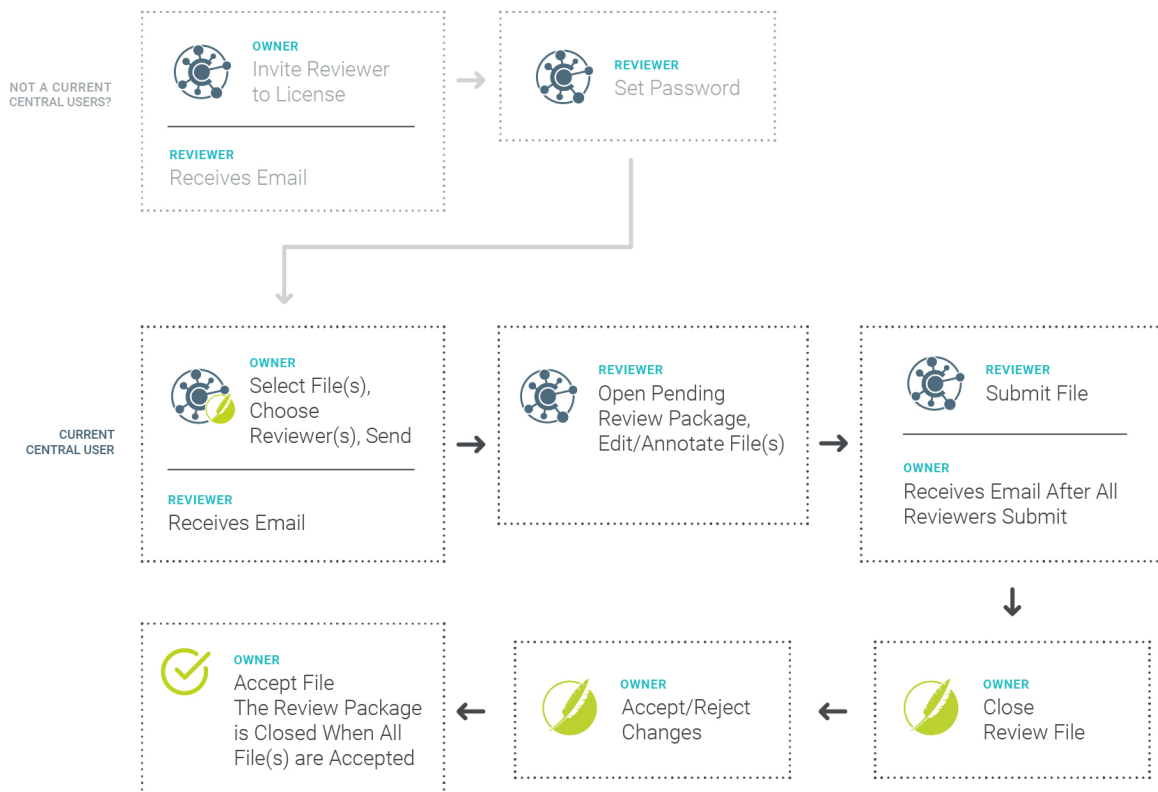
Pros

It is an easy workflow.

You can send review packages that contain files such as topics or snippets for review.

You can bundle files along with a TOC node. This provides more context for content while reviewing.

SMEs do not need to download and install any software. The review takes place in the cloud.

Multiple reviewers (e.g., SMEs, authors) can make changes and add comments to the same topic or snippet at the same time.

A lightweight version of the editor means a streamlined interface. You only see options and features that are relevant to the review process.

Changes in the editor are auto-saved as you work. In addition, all changes are automatically tracked so the owner can easily locate edits for approval or rejection.

Files can be sent for review from a specific Git branch. This lets you keep reviews limited to files that are still in a state of development, as opposed to finished and ready for publication.

Cons

Although snippets that are part of topics can be included in the review (or excluded from it), you will need to communicate with your reviewers about them. You can tell reviewers to open snippets separately in Central to provide edits and comments, or you can have reviewers insert comments above the read-only snippets in the topic itself. You might consider including larger and more substantial snippets as part of the review process, while excluding smaller snippets that you don't feel need to be edited by reviewers.

Flare to Flare—Peer Reviews

If you need a fellow Flare author to review content, a convenient solution is simply for both authors stay within the Flare project. There are a couple of ways to do this.

First, the original author can communicate (verbally, via email, etc.), asking another author to review the file(s). The reviewer opens the file(s), tracks his or her changes, and makes edits and/or annotations. The reviewer lets the original author know the files have been reviewed. The original author opens those files and accepts or rejects the changes.

A second way to use Flare for reviews between authors is to use the option in the Review ribbon to send files for review (in a review package).

After the reviewer finishes and returns the files, the original author can view the tracked changes, accepting or rejecting them as necessary. That author can then click a button to accept a reviewed file, replacing the original file with the reviewed one. For more information see the online Help.

Pros

By keeping everything within Flare, there is no need to transfer changes manually or convert file types. It is therefore more automated than some of other review solutions.

Using a Flare-to-Flare workflow can work quite well if only one reviewer at a time is looking at a file.

Cons

If you need multiple authors to review the same files, those files will need to be "daisy chained," reviewed by only one person at a time.

In order to see tracked edits, you need to click an option on the Review tab to show changes.

Flare to Contributor—SME Reviews and Contributions

If you need a SME—rather than another Flare author—to review content, MadCap Contributor is one solution.

For reviews, you send files to the person in a review package. That individual opens Contributor, which is like a scaled-down version of Flare. Changes are automatically tracked, and the reviewer can also add annotations, then send the review package back to you. After accepting or rejecting changes in those files, you can accept them back into the Flare project, replacing the original source files.

For new content, the person using Contributor can create the content and send it to you. You can then accept the file(s) into your Flare project.

For more information see the online Help.

Pros

Because Flare and Contributor read the same types of files, there is no need to transfer changes manually or convert files. It is more automated than some other review solutions.

This solution can work quite well when only one individual at a time is looking at a file.

Cons

A SME will need to have Contributor installed locally.

Although it works similarly to many other editors, there are some new tasks that a SME will need to learn. Also, because Contributor is used for both reviews and contributions, there may be many more features and options in the interface than a person will need to use.

The process can get complicated if you need to have multiple people review the same file. The files will need to be "daisy chained," reviewed by one person at a time.

PDF—Peer and SME Reviews

Another option for both peer and SME reviews is to generate PDF output and send it to whomever you need to review the content. The reviewer then makes comments in the PDF and sends it back to you. After this, you manually make changes to the Flare files based on the PDF comments.

Pros

PDF is a format that is familiar to most people.

Cons

Reviewers cannot edit the PDF. They can only add comments.

There is no way to automatically transfer feedback into your source Flare files. It must all be done manually.

You will need to set up the PDF output in your Flare project, if it is not done already. And even if it is already done, the entire output might be much bigger than the small section that you need the person to review.

Word—Peer and SME Reviews and Contributions

This option for peer and SME reviews is similar to the one for PDF. With this solution, you generate Word output and send it to whomever you need to review the content. The reviewer then makes changes or comments in the Word document and sends it back to you. After this, you manually make changes to the Flare files based on the Word document

Pros

Word is a format that is familiar to most people.

Reviewers can both make edits and add comments.

Word targets in Flare have an option that lets you enable the review mode for the output. This way, reviewers do not need to remember to enable the Track Changes feature in the Word document. For more information see the online Help.

Cons

There is no way to automatically transfer edits or comments into your source Flare files. It must all be done manually.

You will need to set up the Word output in your Flare project, if it is not done already. And even if it is already done, the entire output might be much bigger than the small section that you need the person to review.

What the MadCap Documentation Team Does

We switch to the relevant branch and send topics and snippets to reviewers (authors or SMEs) as necessary. In addition, we limit the snippets that are included with reviews. It is more likely that we include larger snippets in a topic, while excluding many of the smaller snippets. We feel that including too many snippets as part of the review process can confuse reviewers, especially SMEs, who might not be familiar with snippets. So we communicate with reviewers to open and edit the snippets that we do include, but to insert annotations as necessary next to the read-only snippets that are excluded from the process.

▎Publishing Output

Publishing output is the final piece of your external architecture. The way you publish output depends to a certain extent on the kind of output you are producing. There are lots of output types you can choose from in Flare, but we will focus here on the two most recommended formats—HTML5 and PDF. The same things discussed below might apply to the other types of output formats available in Flare.

Web Server

Many writers publish their output files to a web server so that they can be seen by end users. You might do this by setting up a destination file in your Flare project, which will automatically copy and upload the output files to the location you choose. Alternatively, you might use a third-party tool, such as FileZilla, to upload copies of your files.

Product Application

Some authors need to provide output files to developers in order to integrate a Help system into a software application. You might use a destination file to copy the output files to a location where developers can retrieve them, or you can manually copy and paste the output files. You might even place the output files into a source control repository for your developers. This can be done using a destination file in Flare, or by using a third-party tool such as Microsoft Visual Studio.
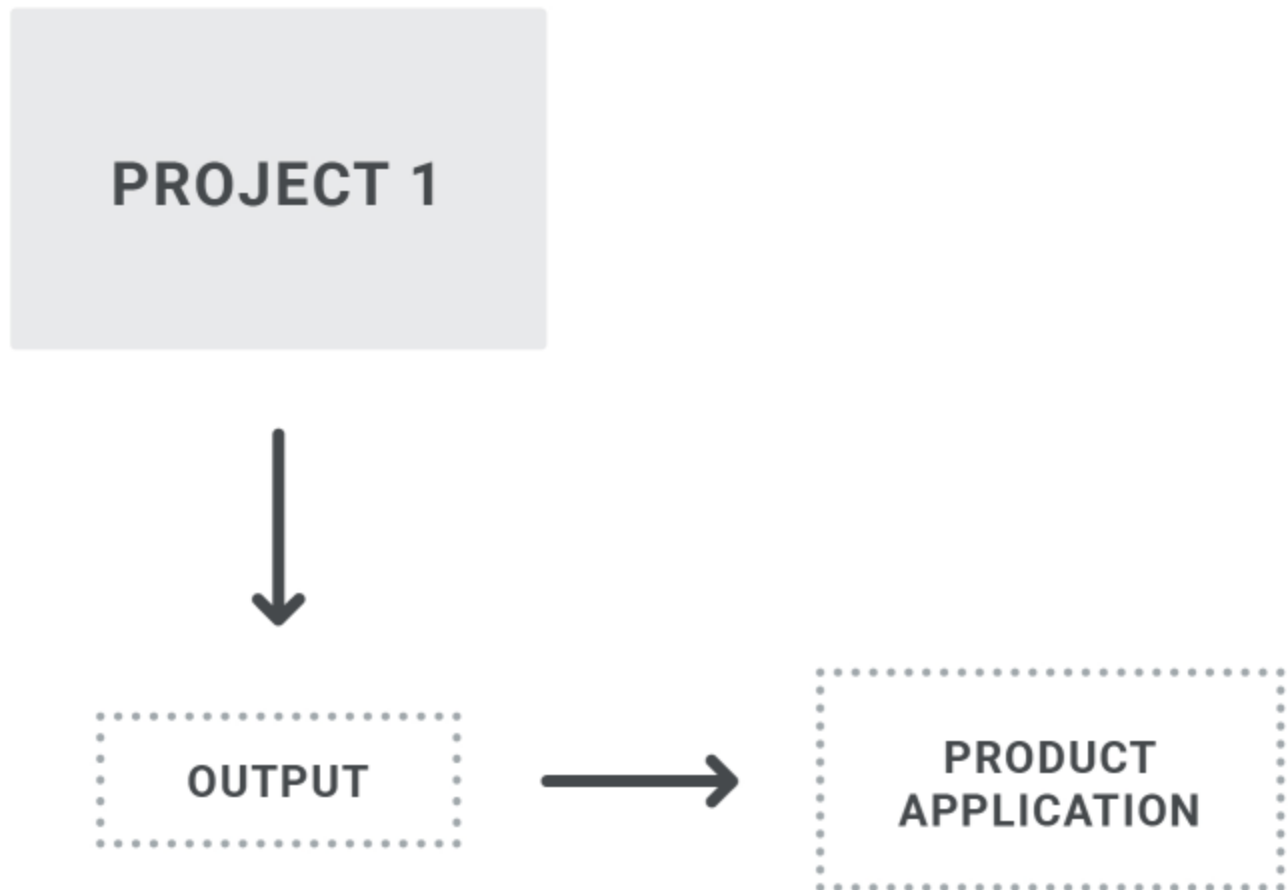
MadCap Central

If you subscribe to MadCap Central, there are certain benefits, including the following:

- You can quickly generate and publish your output with a few clicks. There is no need to move any files around. Any site set as "live" will be published and immediately available to your end users on Central's servers.

- You can quickly roll back the published output to an older build in an instant.

- You can customize the final URL for the output in various ways, including mapping to your own host domains.

- You can publish output directly from a Flare project to a Central license.

- You can publish private output so that end users must log in to see the documentation.

- You can select a specific branch for published output on a site (if you are using Git branching).

For detailed information about building and publishing on Central, see help.madcapsoftware.com/central.
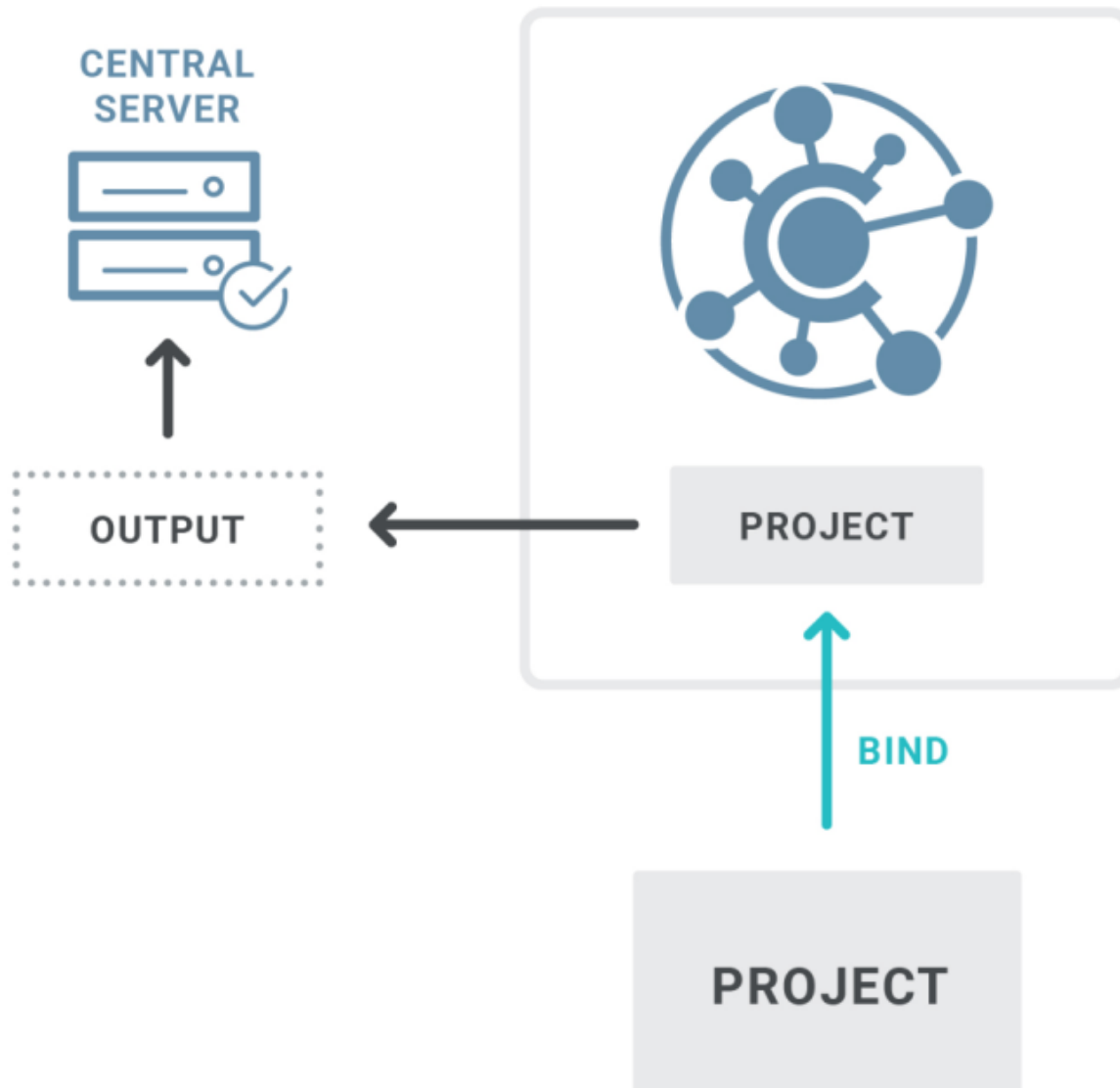
The image above illustrates the process if your Flare project is bound (uploaded) to Central. However, you also have the option of publishing output directly to Central from your Flare project. By "publish," we mean copying your output files to Central, not making that output "live," which would make it visible to the general public. You would still need to use Central to make that output live.

SharePoint

If you have a SharePoint connection in your Flare project in order to leverage those external files, you also have the option of publishing output to a SharePoint server. You can do this in Flare by setting up a destination file and pointing to that SharePoint server. For more information see the online Help.

PDF Inside Online Help

There might be times when you want to place one kind of Flare output inside another. The most common example is a print-based output such as PDF inside an online Help system.

You can do this by manually copying and pasting the PDF into your Content subfolder in Windows. Alternatively, you could use the External Resources feature. Once the PDF is in your project, you might want to link to it in your TOC. If you are generating the HTML5 server-based output, the PDF content will be included in online searches for text.

What the MadCap Documentation Team Does

We use multiple methods for publishing our output.

MadCap Central

Our main "Shared" project is bound to Central. From there, we build some output (for various branches) and publish it on the Central server. Also, we create vanity URLs for targets, which lets us control the final URL for our output. For details, see help.madcapsoftware.com/central.

Web Server

We also publish HTML5 and PDF output on a web server.

Product Application

Each MadCap Software application has both local Help and server Help. The server Help is HTML5 output that we can update any time, since it is separate from the application. The local Help is HTML5 output that is installed with each application (if the end user chooses to install it). This output is updated only when updates occur for the application itself. After generating the local Help, we check it in to Microsoft Team Foundation Server using Visual Studio. Our developers then point to this output, integrating it in to the application.

Internal Architecture

Internal project architecture has to do with the structure of files and use of features within a Flare project. Whatever approach and features you adopt, remember that consistency is important. Establishing consistent rules and guidelines will result in a cleaner project. It also makes it easier to locate files, which is especially important for teams of writers.

This chapter discusses the following:

❙ Naming Conventions

Put some thought into the way you name your folders and files, whether they are topics, snippets, images, targets, skins, etc. This is important not only for consistency, but also for potential effects on search results.

Word Separation

Many of your file and folder names are going to consist of more than one word. Here are some ways you can handle this:

- **Hyphens (Recommended)** Hyphens provide visible separation between words and are recommended by some search engine optimization (SEO) experts.

- **Spaces** Adding spaces between words is simple. However, blank spaces can cause issues for Web servers, particularly those running UNIX or Linux. Also, in the browser path field, the characters "%20" will be inserted between each word (e.g., Choosing%20a%20Cat.htm instead of, say, Choosing-a-Cat.htm).

- **No Spaces** For best results with keyword recognition, it is recommended that you do not run keywords together in file names (e.g., choosingacat.htm). When keywords run together, there is a greater risk of the keywords not being recognized, as well as not being parsed properly by search engines.

- **Underscores** While it is a common practice to use underscores in file names (e.g., Choosing_a_Cat.htm), many search engine providers and SEO experts no longer recommend it because the underscore is considered by some indexes to be its own word character. While it provides visible separation between words, one point to consider is that some search engines might take regular expressions such as "Choosing_a_Cat" and interpret it as "ChoosingaCat."

- **Periods** It is recommended that you do not use periods (.) to separate words in file names. Periods are used by some computer systems to indicate different file extension types.

Case

You might choose to uppercase or lowercase in your folder and file names. Because some major search engine providers run indexing systems on case-sensitive servers (e.g., Apache, Unix, and Linux servers), whether a file name is written in lowercase or uppercase can make a difference as to whether the file is indexed. For example, a page named "choosing-a-cat.htm" may be interpreted differently from "Choosing-A-Cat.htm." It can also lead to search engines creating duplicate page entries, which can result in error codes when users click links, as well as penalties for content ranking. While it is difficult to predict how each and every search engine will handle a file name, it is good to pick your file naming convention wisely and then use it consistently.

Length

Because file names are part of a URL, studies show that the general rule of thumb for page URLs is that shorter URLs are better. For example, a file named "choose-a-cat.htm" would be a more compelling link for an end user to click than a file named "how-to-go-about-choosing-a-cat-or-a-kitten.htm."

Key Words or Letters

You might consider adding key words or letters at the beginning or end of certain names to make them easily identifiable and organized. For example, perhaps you have created 50 snippets that are specifically intended for a feature called "transmission." Therefore, you might want to begin the name for each of those snippets with that word (e.g., transmission-getting-started.flsnp, transmission-initial-steps.flsnp, transmission-note-1.flsnp).

What the MadCap Documentation Team Does

Here is what we do for naming conventions:

- **Hyphens** For both file and folder names, we use hyphens between words.

- **Title Case** We use title case when naming our folders and files (e.g., Creating-Project.htm, rather than Creating-project or creating-project.htm).

- **Length** There are some topics that have somewhat long headings out of necessity. However, even in those situations, we try to limit the length of the actual file name, leaving out unnecessary or minor words, such as articles (e.g., Excluding-Javascript-CSH-Calls-HTML5.htm, rather than Excluding-Javascript-for-Context-Sensitive-Help-Calls-in-HTML5-Output.htm).

- **Keywords and Letters** For some file names, we use certain keywords or letters. For example:

  - For snippets that consist solely of an example, we begin the files with the word "Example" (e.g., "Example-Responsive-Content.flsnp," "Example-Maximum-Concurrent-Builds.flsnp"). It just makes it easier to find and identify them.

  - We have many troubleshooting topics that are different from most of our other topics. So we will start the name of each topic with "TS" (e.g., "TS-Condition-Tags.htm," "TS-PDF-Output.htm").

  - For small images of buttons in the user interface, we add the letters "btn" at the end of the image file name (e.g., "Save-btn.png," "Search-btn.png," "Font-btn.png"). This tells us at a glance exactly what kind of an image it is without having to open it.

❙ Folders

The more files you have in your project—whether they are in the Content Explorer or Project Organizer—the more necessary it will be for you to create folders and subfolders to organize them. For more information see the online Help.

Categories

Create intuitive categories for your folders based on the nature of your product. For example, maybe you are writing documentation for several company products, in which case you might choose to create the first level of folders based on those product names. Or maybe you decide it's best to create folder names according to features. Then again, there might be something else that dictates how you name folders, such as modules that your company uses for organization.

Conditions

Not only can you place conditions on content within topics, snippets, and template pages, but you can also set conditions on folders or files. If all of the files within a folder need to have the same condition on them, it is a good idea to place the condition on the folder instead. This can save a lot of time because whatever conditions you apply to a folder will be set automatically on the files within it. For example, if you have a folder that contains 500 files, you can put a condition on just that one folder instead of each of the 500 files within it.

Order

Whatever you name your folders and subfolders, they will be listed in the Content Explorer and Project Organizer alphabetically. If you want them to be listed in a particular order, you can begin folder names with certain numbers or characters to force the order that you want.

Resources

In the Content Explorer, we recommend that you use the Resources folder (and the subfolders under it) to hold files such as images, template pages, stylesheets, and more. Although it is possible to store such ancillary files anywhere else in the Content Explorer, using the Resources folder and its subfolders helps keep them files well organized.

What the MadCap Documentation Team Does

We do the following with our folders.

Categories

At one time, we organized all of our first-level folders according to the names of features in MadCap's products. But over time, we found that it was easier for us to organize by product, so most of the first-level folders have the name of a product. Within those product folders, we created more subfolders, usually named after features.
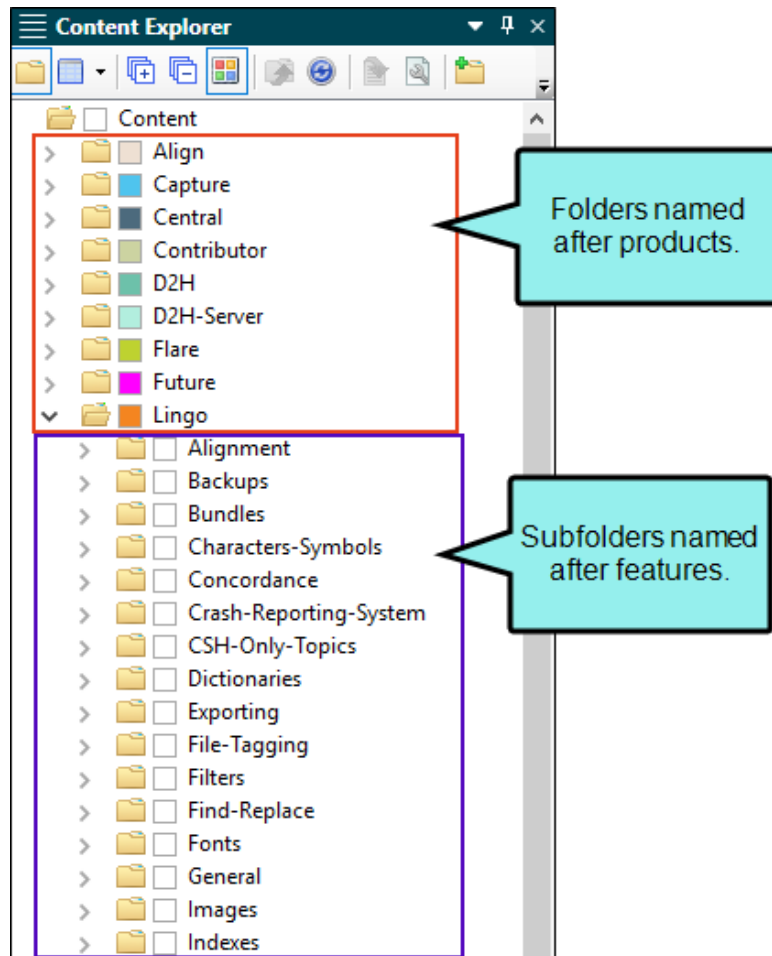
Conditions

We place conditions on all of the folders that are named after a product. Also, our products are associated with certain branding colors, so we use the same colors when creating those conditions.
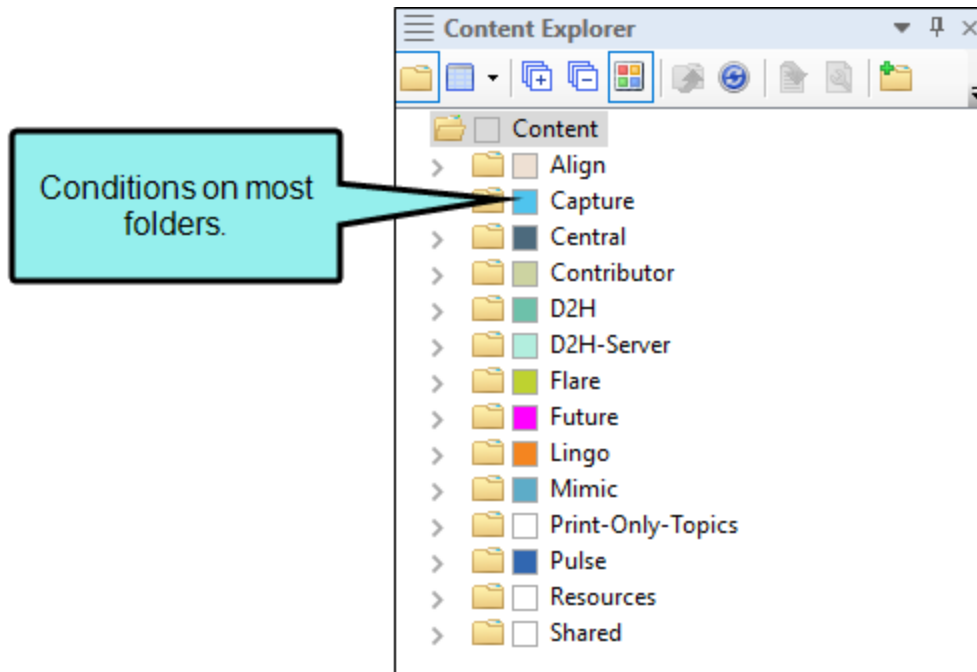
We have some files that are used in more than one product. This includes files such as snippets and images. For those files, we have created special folders named "Shared," and we set conditions on the subfolders holding those files.
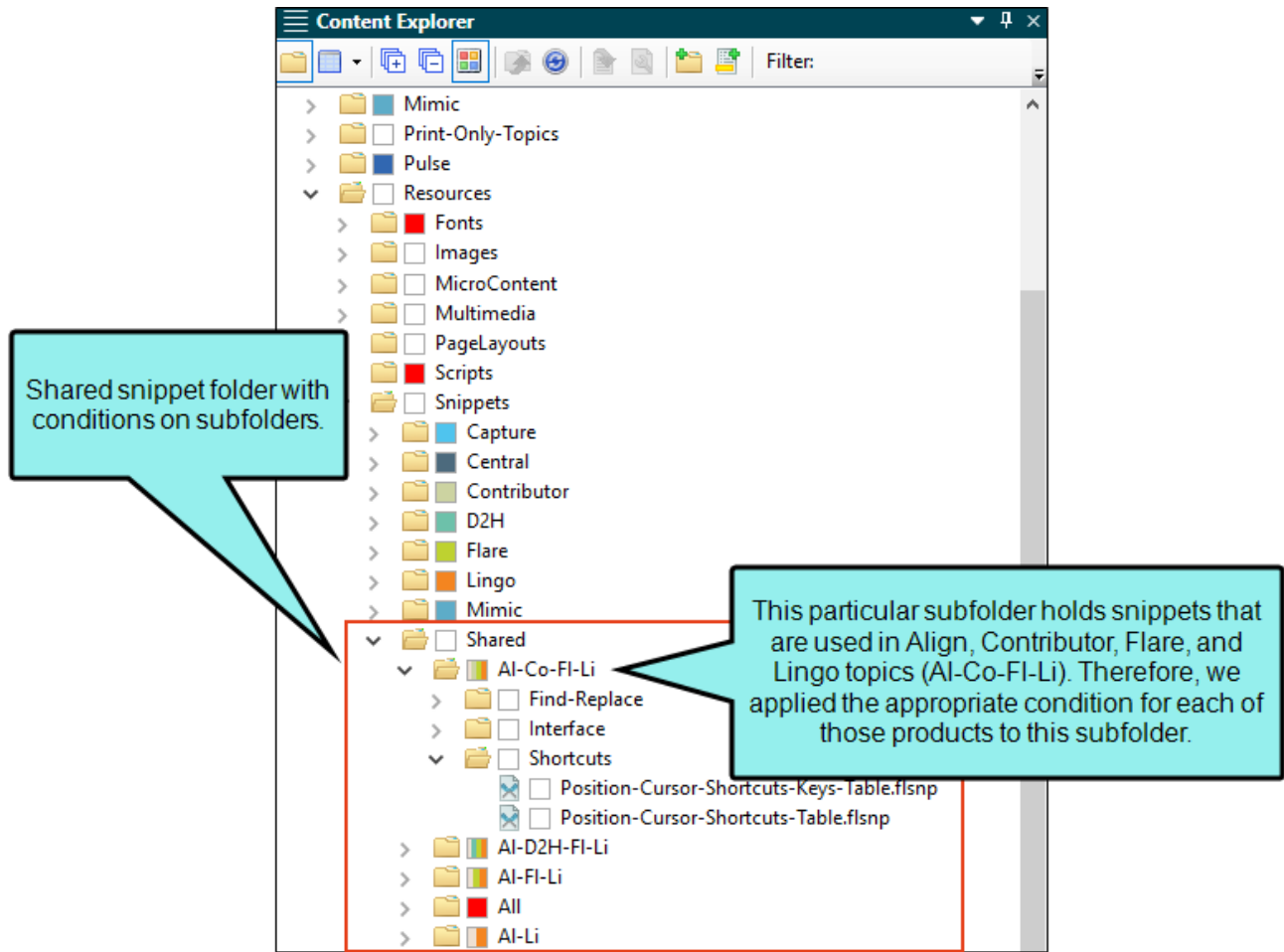
It is important that you try to make organizational decisions such as these early in your project's life. Doing so will help prevent the need to spend a lot of time changing folders and files later.

❚ Types of Topics

You can create different types of topics depending on your needs. For example, you might want to mimic DITA's model, creating the following kinds of topics, even creating your own sub-categories:

- **Concept** This is a topic that describes the meaning, background, and context for a subject. These are often "landing" topics, which usually provide links or navigation to related topics.

- **Reference** This is a topic that is designed to store detailed reference information, often in one or more tables.

- **Task** This is a topic that provides steps for completing procedures.

- **Troubleshooting** This is a topic that explains why a certain issue occurs and how to correct it.

Standards and Structure

It is a good idea to set internal standards and design a basic structure for each type of topic you need to use. For example, your team might determine that each task topic should include the following elements, in order and perhaps written in a particular way:

- Heading

- Summary Paragraph(s)

- Task Heading

- Steps

- Tips

- Notes

Using topic templates is a good way to ensure consistency when creating each of these types of topics. You can include a skeleton structure for each topic type, with dummy text that authors replace in new topics. See "Templates" on page 36.

You can also make a copy of an existing topic of the same type and then replace the content with new information. However, use caution when doing this because some information might be unique to the original topic (e.g., meta description, template page), and you need to adjust that information as well in new topics.

Naming

You might consider using certain standards when naming your topics to help organize and identify them. For example, you might begin topics with the words "Concept," "Reference," or "Task," depending on its type (e.g., Concept-Dog-Overview.htm, Reference-Dog-Breeds.htm, Task-Training-Dogs.htm).

What the MadCap Documentation Team Does

Most of our topics can be classified as one of the following:

- Concept
    - Major landing page
    - Minor landing page
    - General concept
- Task
- Reference
    - General reference
    - UI element
- Troubleshooting

We have created templates, which are skeleton versions of each of our topic types. When creating new topics, these templates can be used by any author.

▍Snippets and Conditions

Snippets and conditions are two of the most important features in Flare, especially when it comes to single-sourcing content. For more information see the online Help.

Organization

As is true with other kinds of files, the more of these you use in your project, the more important it will be to organize them. For snippets, that means creating subfolders in the Resources > Snippets folder of the Content Explorer. For conditions, it means using multiple condition tag sets in the Project Organizer.

Snippet-Focused Versus Condition-Focused Topics

Another important consideration when structuring your topics is whether to make them more snippet-focused or condition-focused. Although you are likely to use both snippets and conditions in your topics, you might give more weight (or focus) to one than the other.

☆ **EXAMPLE** You have a project where you are writing documentation for three company products—Product A, Product B, Product C.

You have a topic that is used in each of those products. The topic is made up of five paragraphs. The first two paragraphs are identical for all three products. The third paragraph is unique to Product A, the fourth paragraph is unique to Product B, and the fifth paragraph is unique to Product C.

TOPIC HEADING

PRODUCTS A, B, C

PRODUCTS A, B, C

PRODUCT A

PRODUCT B

PRODUCT C
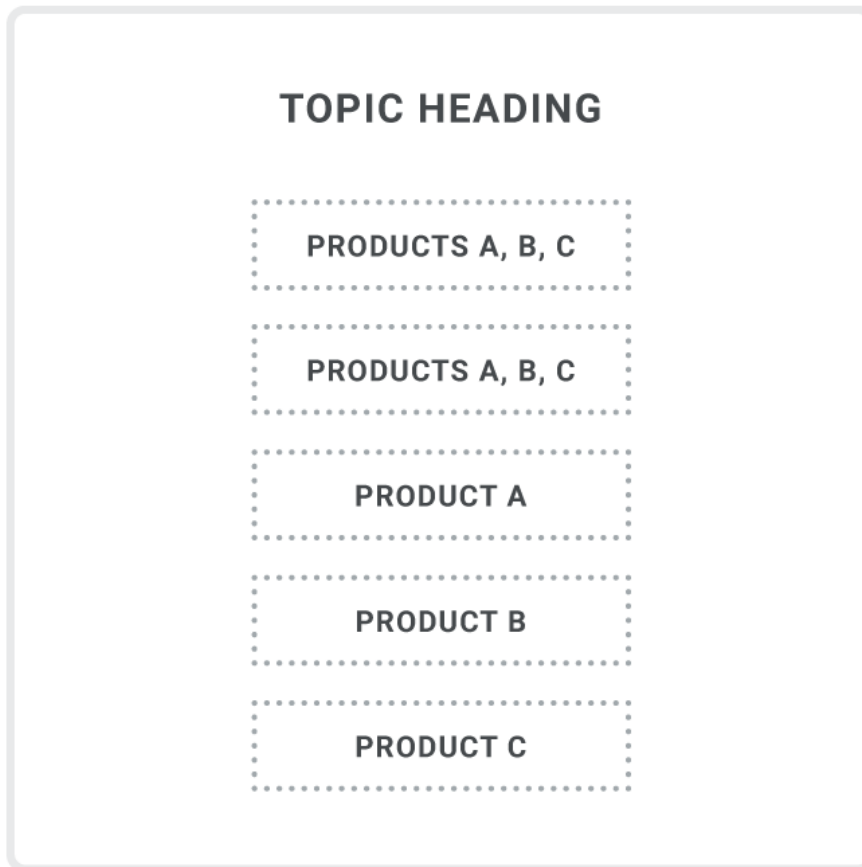
Should you have one topic or three? What is the best way to structure the topic? Which features should you use?

☆ You might choose to take a more condition-focused approach, or you might decide a snippet-focused approach is better. With either approach, this doesn't mean that you are limiting yourself only to conditions or only to snippets; instead, it means that one of those takes on a more prominent role. If you use a condition-focused approach, you're still likely to employ snippets in some ways, and likewise, if you use a snippet-focused approach, you will likely still have some conditions (e.g., online versus print).

CONDITION-FOCUSED APPROACH

With this approach, you would have just one topic, and you would use conditions to include or exclude paragraphs for a given target depending on the product.

Therefore, the first two paragraphs would not need a "product" condition since they are needed for all three outputs (i.e., the target for Product A, the target for Product B, and the target for Product C). In those first two paragraphs, you might have just regular content, nothing more.

However, on the third, fourth, and fifth paragraphs, you would place conditions that are unique to each product.

TOPIC HEADING

PRODUCTS A, B, C	REGULAR CONTENT
PRODUCTS A, B, C	REGULAR CONTENT
PRODUCT A	CONDITION ON REGULAR CONTENT
PRODUCT B	CONDITION ON REGULAR CONTENT
PRODUCT C	CONDITION ON REGULAR CONTENT
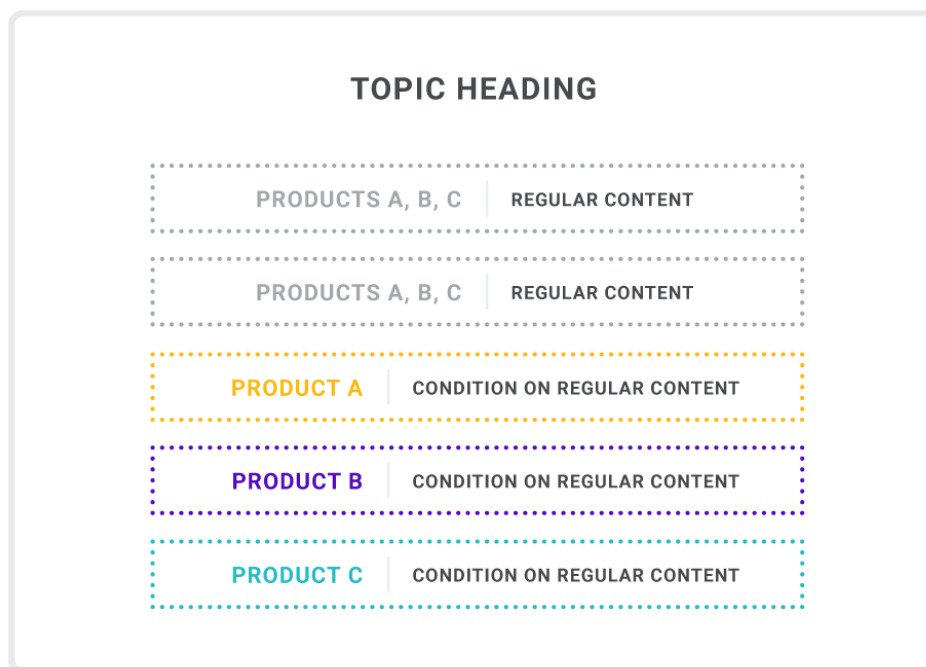
☆ SNIPPET-FOCUSED APPROACH

With this approach, you would have three topics instead of one. The first two paragraphs in each topic would be snippets, since the content is the same in all three topics. However, the third paragraph in each topic is simply regular content that is unique to each product; no conditions are necessary on those paragraphs.

TOPIC HEADING

PRODUCTS A, B, C | SNIPPET

PRODUCTS A, B, C | SNIPPET

PRODUCT A | REGULAR CONTENT

TOPIC HEADING

PRODUCTS A, B, C | SNIPPET

PRODUCTS A, B, C | SNIPPET

PRODUCT B | REGULAR CONTENT

TOPIC HEADING

PRODUCTS A, B, C | SNIPPET

PRODUCTS A, B, C | SNIPPET

PRODUCT C | REGULAR CONTENT

An obvious benefit of the condition-focused approach is that you only need to create one topic instead of three. On the other hand, a topic with a lot of content can get confusing, especially when much of it has conditions applied. Figuring out which content goes with what might take more time and effort than you want. This can be especially important if you have new writers or need to send the topic to subject matter experts for review.

As for the snippet-focused approach, it can be much more straightforward when you need to return to it for editing or send to a reviewer. However, this approach requires you to create multiple topics instead of one.

There isn't a right or wrong answer. Either approach can work nicely for your needs. It is recommended that you try each approach on a few topics to see which one you like best. In the end, the best choice is the one that is going to save you the most time and effort, and help you to avoid errors.

What the MadCap Documentation Team Does

At one time, we used a condition-focused approach. With this approach we created fewer topics, but over time we found that we were spending too much time on some of our shared topics, trying to decipher what all of the conditions meant and what needed to be edited.

Now we typically use snippet-focused topics. Even though we are creating more topics, we find them to be much easier to handle, especially since we have a very large number of conditions in the project. The more streamlined we can make the topics, the easier they are to handle, especially for a new author or a reviewer. The bottom line is that we found we were spending less time and effort on snippet-focused topics than we were on condition-focused topics.

Also, because we use a lot of snippets with this approach, we created a special subfolder next to our product subfolders, and we named it "Shared." Within this folder, we created even more subfolders to store snippets that are used for multiple products.

We also set the appropriate product conditions on each snippet file. These conditions have nothing to do with our snippet-focused approach. Rather, these conditions are used when we import files into our child "Develop" projects using Global Project Linking.

❙ Variables

Variables are extremely important in Flare projects. You might find that you use only basic variables, such as your company name, phone number, version numbers, or other general information. On the other hand, you might need to have a great many variables for different uses. In addition, you might need to use system variables (e.g., headings, date, time) in certain places. For more information see the online Help.

Variable Sets

Custom variables are stored in variable sets, and it's possible you might need only one variable set to hold everything you need. However, the more custom variables you have, the more likely it is that you will want to create multiple variables sets to keep them organized.

Conditions Versus Overriding Variable Definitions

You might be tempted to insert many variables of the same type (e.g., product names) in a topic and then place conditions on each variable in order to separate them at the time you build a target. This is not necessarily wrong, but you might find that it is more efficient to use just one variable and then override the definition as necessary for your various targets.

☆ **EXAMPLE** You have a project where you are writing documentation for five company products—Product A, Product B, Product C, Product D, and Product E. Instead of typing the name of each product manually throughout your project, you use variables for each one.

Now suppose you have a topic that is relevant to all five products. So you create five separate conditioned paragraphs (which is better for translation purposes than one paragraph) and insert the variables at the correct location.
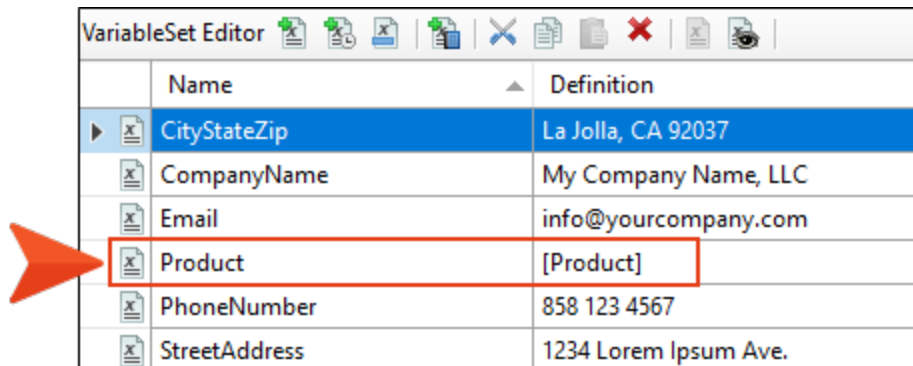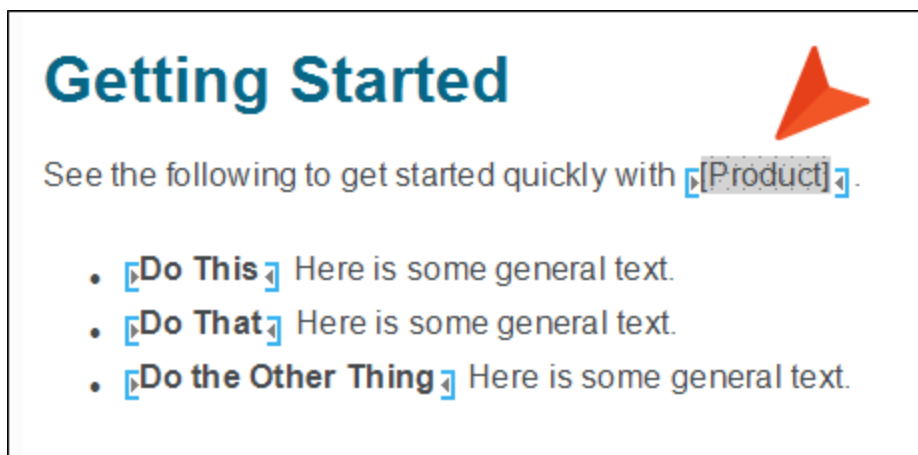
☆ This certainly will work, but there is a simpler way to do the same thing. Instead of five variables (and five paragraphs), you can create just one, perhaps naming it "Product."

Then in the topic where all five variables were inserted, you use this one variable instead.

☆ Then in each of the five targets, you override that variable definition with the name of each product.

So the target for Product A would look like this:
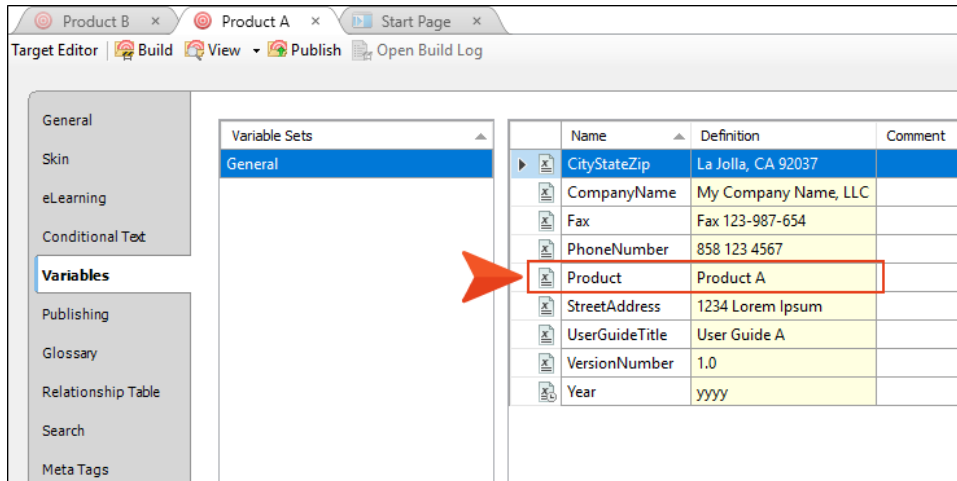
And the target for Product B would look like this:
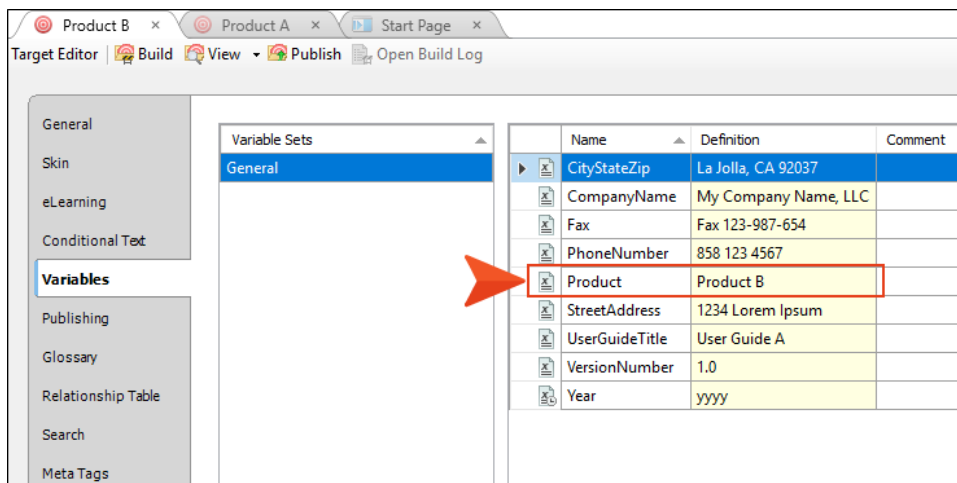
The targets for Products C, D, and E would likewise have the variable definition customized. This means that you always insert the same variable in topics and snippets, regardless of the product(s) affected, and the correct name is automatically shown when you build a certain output.

Multiple Variable Definitions

You can also add multiple definitions to a variable, using them as a way to override variables on a target. For more information see the online Help.

This is not necessarily better nor worse than the other method for overriding variable definitions described above. It is simply another alternative.

☆ **EXAMPLE** Your company might have multiple phone numbers. You can associate them all with the same variable and use the appropriate one wherever necessary.

When a variable has multiple definitions, cells of the non-default definitions are grayed out (except for the definition cell).

The default definition is determined by the order of creation. The first definition created is the default.

> ☆ In the target, you can then override the default definition with any of the others that were created.
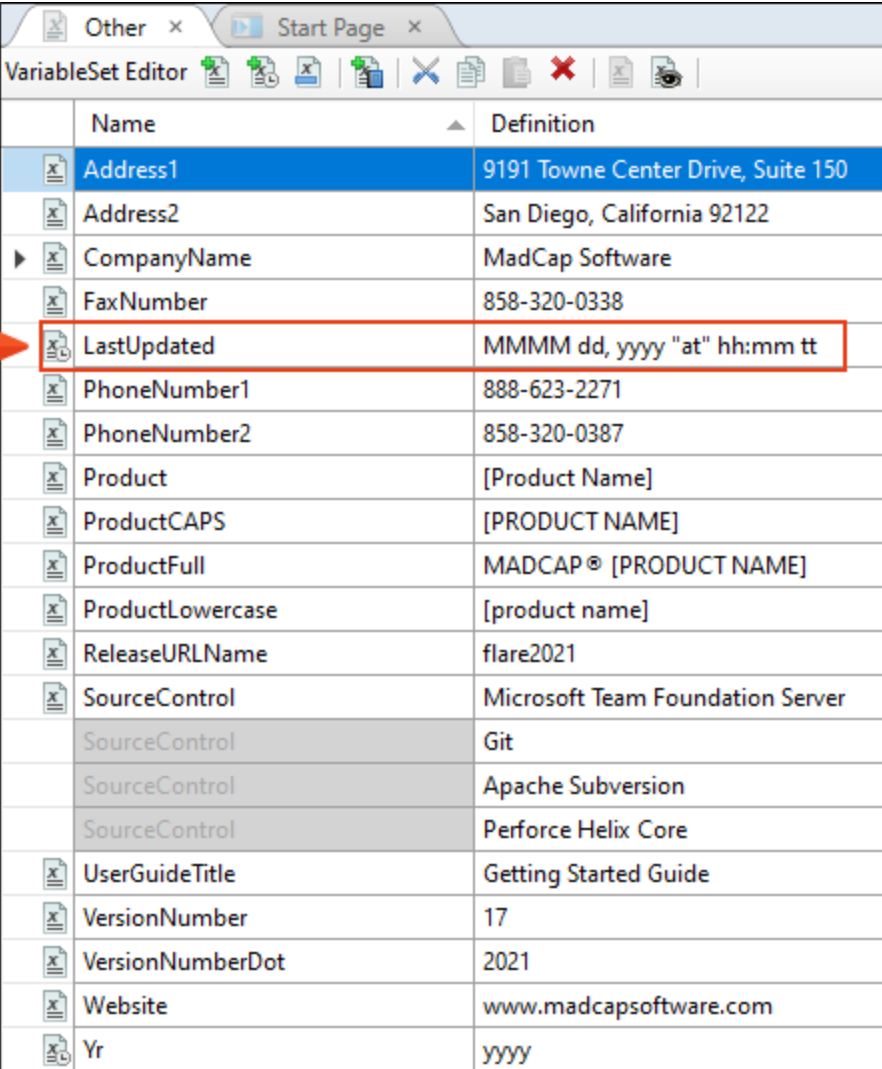
What the MadCap Documentation Team Does

Our variables are organized into the following sets:

- **Other** These are general variables, such as company information (address, phone), product names in various formats (e.g., regular, all caps, lowercase), user guide titles, version numbers, and the year.

- **Outputs** These are variables with names of all the output types available in Flare.

- **PDF-URLs** These are URLs to all the PDFs that we produce. They are typically used in template pages and some micro content snippets.

In the "Other" variable set, we have a couple of date/time variables. We have one variable called "LastUpdated," which uses the following format:

MMMM dd, yyyy "at" hh:mm tt

This variable is inserted at the bottom of a topic that automatically displays the day, year, and time when the output was last generated.

Company and Product Information

See the following for general information about MadCap Software and this release of Flare.

MadCap Software
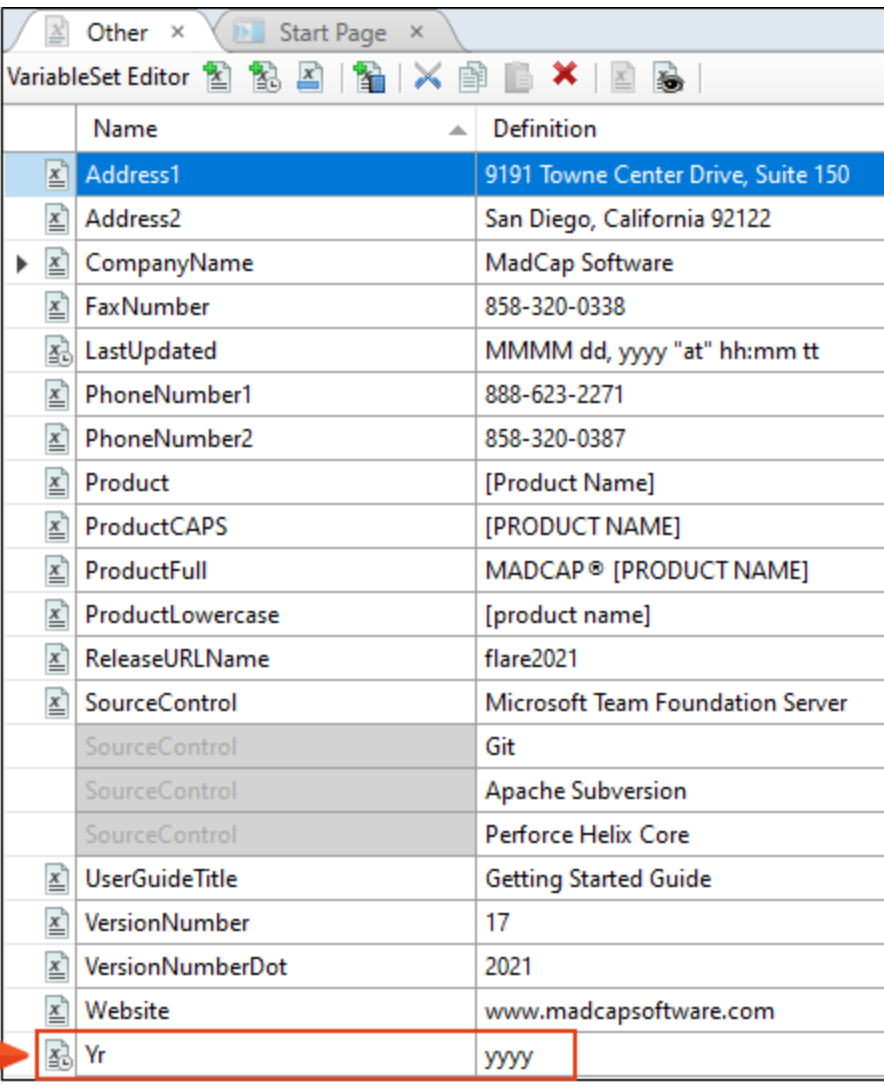
9171 Towne Center Drive, Suite 335
San Diego, California 92122
Toll Free: 888-623-2271
Tel: 858-320-0387
Fax: 858-320-0338
www.madcapsoftware.com

MadCap Flare 2023

Copyright © 2023 MadCap Software
Help System Last Updated on April 11, 2023 at 09:51 AM
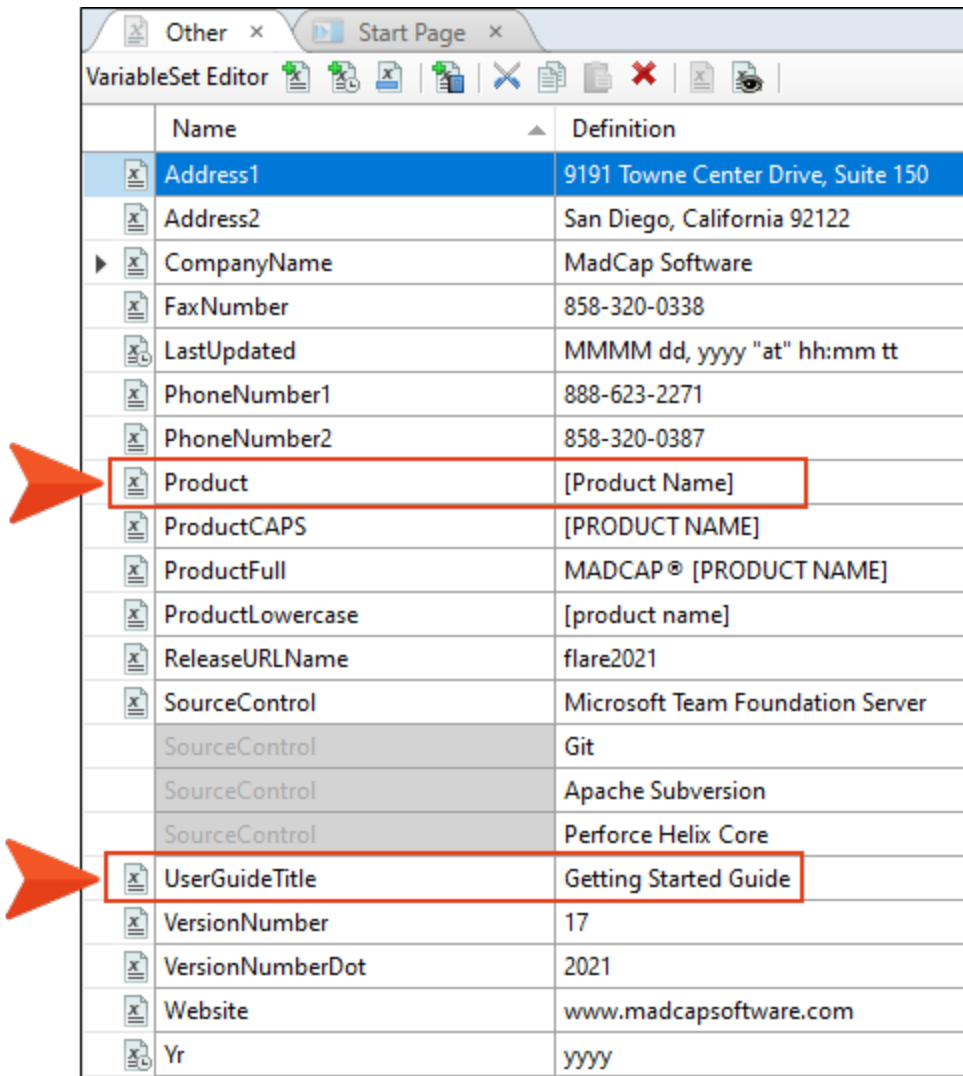
We also have a date/time variable called "Yr," which uses the yyyy format. This format simply displays the current year (e.g., 2024) automatically whenever output is generated.

We never need to update either of these date/time variable definitions because they are always generated automatically when we build the target.

There are other variables that are entered once in a variable set (e.g., Product, User Guide Title) with a random definition provided (e.g., Flare, Getting Started Guide).
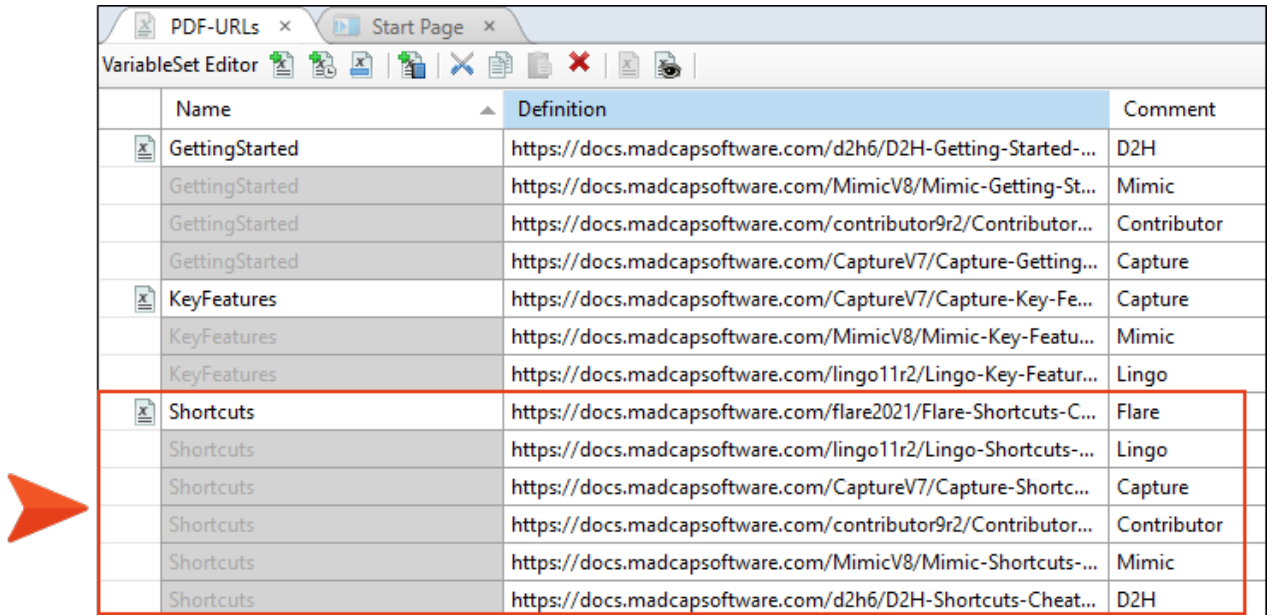
However, in each target we override that variable if necessary by simply renaming the definition by typing it.

And there are other variables for which we have provided multiple definitions. This is the case with some our variables that provide PDF URLs. For example, from many products we generate a "Shortcuts" PDF cheat sheet. The variable name is "Shortcuts," but we provide a different URL for the PDF of each product.

Then in each Shortcuts target, we select the appropriate product URL.

❚ Stylesheets

As with other facets of your documentation universe, you have choices as to how you can structure stylesheet(s) in Flare projects. For more information see the online Help.

How Many Stylesheets?

One of the first decisions you should make when it comes to stylesheets is how many of them you create for your projects. This is true for both regular stylesheets and table stylesheets. Here are the primary options before you:

- One Stylesheet for One Project
- One Stylesheet for Multiple Projects
- Multiple Stylesheets for One Project
- Multiple Stylesheets for Multiple Projects

One Stylesheet	Multiple Stylesheets
As a general rule, having fewer files is usually best when possible. If you are able to limit yourself to a single stylesheet, you have just one place to maintain styles. If you have multiple projects, the best way to stay limited to one stylesheet is to use Global Project Linking to import the stylesheet into child projects. For more information see the online Help.	Although a single stylesheet is usually preferred, creating multiple stylesheets can allow for a greater variety of looks.

One Stylesheet	Multiple Stylesheets
This solution can work quite well when only one individual at a time is looking at a file.	Although a single stylesheet is usually preferred, creating multiple stylesheets can allow for a greater variety of looks. This is not to say that you cannot achieve the same result in a single stylesheet. With enough effort and knowledge about styles, almost anything is possible. But for some authors, using multiple stylesheets in a situation such as this might be easier and make more sense.

Which Level?

Within a project, you can set a regular stylesheet at three levels—project, target, or content file. When possible, setting a stylesheet at the project level is preferred, because doing so makes the styles within the stylesheet available to all content files through the project. However, it might make more sense for some authors to set a freestyle at the target or file level, because it gives you more flexibility when using multiple stylesheets. For more information see the online Help.

Linking Stylesheets

Linking stylesheets together is another option some authors choose. This approach is sometimes used if you already have more than one stylesheet that you want to leverage, and you do not want to take the time to merge them into a single stylesheet. Just be aware that the settings from one stylesheet may override those from another when they contain the same styles. For more information see the online Help.

Inside a Stylesheet

Within a stylesheet, you have even more options for controlling how styles are structured. Following are some of the primary features that you might use.

Mediums

A medium is an alternative group of settings in a stylesheet and can be very useful when you are generating multiple kinds of outputs. Unless you tell Flare otherwise, default style settings will be used for the different outputs you generate. But there may be times when you want to override a default style setting for a particular output; that's why you would use a medium. *You need to explicitly tell Flare which medium you want a particular target to use.* This is done from the Advanced tab of the Stylesheet Editor. Also, keep in mind that mediums can be used not only with regular stylesheets, but with table stylesheets as well. For more information see the online Help.

Media Queries

A media query is an alternative group of settings in a stylesheet. These settings are automatically used under certain conditions, such as when a screen of a certain size is displaying the output. Media queries are able to do this because they are configured with specific criteria (e.g., maximum width of the screen, orientation, resolution). When the criteria are met, the style settings in the media query are used to display the output. *You do not tell a Flare target to use a media query; it just happens automatically.* For more information see the online Help.

Classes

In CSS there are primary styles that correspond to the different HTML elements (e.g., h1, h2, p, img). You can think of these as parent styles, because in a way, they can have children. A class is the most common type of child for a style. Some classes might already be included in your stylesheet when you first create a project. Classes can be quite important and useful in a stylesheet. In particular, you might want to use generic classes to streamline your stylesheet even more; generic classes can be used with any parent style in your stylesheet, and no just one specified style. For more information see the online Help.

Identifiers

In CSS, an identifier (ID) is similar to a class, except that IDs are unique. An element in your stylesheet can have only one ID on it, whereas it can have multiple classes. And each page of your output can have only one element with a particular ID. For many authors, using an ID may not be important, but for others—such as those making use of JavaScript—IDs can be very useful. For more information see the online Help.

What the MadCap Documentation Team Does

When it comes to stylesheets, we do the following.

Number of Stylesheets

We have three stylesheets that we use for our projects.

Our primary regular stylesheet is named "styles.css" and is controlled in our main "Shared" project. This stylesheet is periodically imported into our "Documentation Bible" project, as well as all of our "Develop" projects. Most of our work is done in this stylesheet.

In addition to the main stylesheet, we have a stylesheet called "Home-Page.css" that is used specifically for each product's Home topic, which has a look quite different from the rest of the topics. This stylesheet is rarely modified.

A third stylesheet in our project is named "icons.css" and is associated with some of our template pages. We almost never edit this stylesheet.
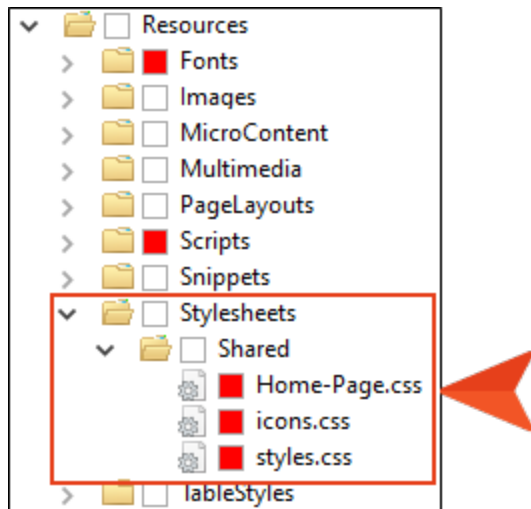
Level

Since almost all of the topics use styles from our main stylesheet, we set this stylesheet at the project level. And because our other stylesheet is used only for the Home topics, we enabled the option to allow local stylesheets.
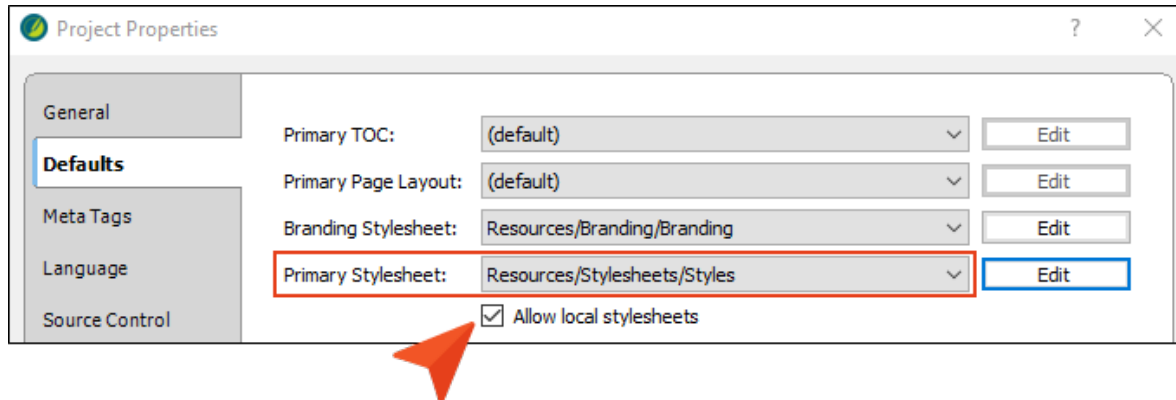

As for the Home topics for the various products, we associated the "Home-Page.css" stylesheet (as well as "styles.css") with each one. We also associated these stylesheets with the template page used by each of those Home topics.

Mediums

In addition to the "default" medium (where most of the styles are set) and the factory "print" medium, we created over a dozen mediums in our main stylesheet. For each product, we have an online and a print medium.

Since we have a separate stylesheet for each product's Home page topic—and because those topics are intended only for online output—we don't have a need for mediums in that stylesheet.

Media Queries

In our main stylesheet, we use the factory "tablet" and "mobile" media queries to change the look of some styles when content is displayed on smaller devices. We also created a couple of custom media queries to account for especially narrow views, such as in the Dynamic Help window pane.

Classes

We use classes a great deal in our stylesheets, especially in the main stylesheet. We also try to take special advantage of generic classes so that we can easily apply the formatting to many different parent styles.

> ⭐ **EXAMPLE** We have a generic class named "center." If we ever come across block content that we want to position horizontally in the center of the page, we can apply that class to it, regardless of the style used.

Tables of Contents

You have various options for structuring the tables of contents (TOCs) in your projects. For more information see the online Help.

How Many TOCs?

You might be working in a project that has just one TOC, or you might have dozens or even hundreds of TOCs. It largely depends on how many different outputs you need to produce.

It is possible to use a single TOC for different outputs. For example, you might be generating both HTML5 and PDF targets and want to use the same TOC. If there are any topics that you want to include in one of the outputs but not in the other, you can set conditions on the TOC entries to control this. However, you might find that this can get somewhat messy if there are many variations. In that case, it might be best to create separate TOCs for each output.

Linking TOCs

Linking one TOC to another is an option that some authors choose. By doing this, the TOCs are combined in the output and appear as a single TOC. For more information see the online Help.

This feature can be especially beneficial if you have multiple authors working in the same project. By assigning writers to different TOCs, you might be able to keep much of their work separate and reduce the chances of file conflicts.

Also, this allows multiple authors to work on the same TOC in a way, because in the end all of the individual TOCs will be merged together into one. Conversely, if you only had one big TOC, you would really be limited to one person working on it at a time.

What the MadCap Documentation Team Does

When Flare was first created, we tried to create a single TOC that was used for both online and print-based output. We needed to place conditions on many of the entries in the TOC, because there were some files that were relevant to only one output or the other. Eventually, we changed our plan, producing separate TOCs for each output. Although this has resulted in more files, it has also minimized any confusion and therefore saved much time and effort.

For each product being documented in our main "Shared" project, we create one TOC that is used for online output. That TOC is used for both our local and server HTML5 output.

Then we create a separate TOC for each PDF that we produce. In the case of Flare alone, this means dozens of TOCs.

We do not link any of our TOCs together. Most of our TOCs are quite established, and the amount of changes to a TOC for each release is usually not that extensive. Therefore, we rely mostly on communicating with others when one writer wants to work on the TOC.

APPENDIX

PDFs

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

▎Tutorials

Getting Started Tutorial

Autonumbers Tutorial

Back-to-Top Button Tutorial

Context-Sensitive Help Tutorial

Custom Toolbar Tutorial

eLearning Tutorial—Basic

eLearning Tutorial—Advanced

Image Tooltips Tutorial

Lists Tutorial

Meta Tags Tutorial

Micro Content Tutorial—Basic

Micro Content Tutorial—Advanced

Responsive Output Tutorial

Single-Sourcing Tutorial

Snippet Conditions Tutorial

Styles Tutorials

Tables Tutorial

Word Import Tutorial

Cheat Sheets

Context-Sensitive Help Cheat Sheet

Folders and Files Cheat Sheet

Learning & Development Cheat Sheet

Lists Cheat Sheet

Micro Content Cheat Sheet

Print-Based Output Cheat Sheet

Search Cheat Sheet

Shortcuts Cheat Sheet

Structure Bars Cheat Sheet

Styles Cheat Sheet

User Guides

Accessibility Guide

Analysis and Reports Guide

Architecture Guide

Autonumbers Guide

Branding Guide

Condition Tags Guide

Context-Sensitive Help Guide

Eclipse Help Guide

eLearning Guide

Getting Started Guide

Global Project Linking Guide

HTML5 Guide

Images Guide

Import Guide

Indexing Guide

Key Features Guide

Lists Guide

MadCap Central Integration Guide

Meta Tags Guide

Micro Content Guide

Navigation Links Guide

Plug-In API Guide

Print-Based Output Guide

Project Creation Guide

QR Codes Guide

Reviews & Contributions With Contributor Guide

Scripting Guide

Search Guide

SharePoint Guide

Skins Guide

Snippets Guide

Source Control Guide: Git

Source Control Guide: Perforce Helix Core

Source Control Guide: Subversion

Source Control Guide: Team Foundation Server

Styles Guide

Tables Guide

Tables of Contents Guide

Targets Guide

Template Pages Guide

Templates Guide

Topics Guide

Touring the Workspace Guide

Transition From FrameMaker Guide

Translation and Localization Guide

Variables Guide

Videos Guide

What's New Guide