# Table of Contents

**Welcome to Ponydocs**
- What is Ponydocs? ................................................................. 1

**Create content in Ponydocs**
- Overview of objects in Ponydocs ............................................ 2
- Create a product ........................................................................ 4
- Create a version .......................................................................... 5
- Create a manual ......................................................................... 7
- Create a table of contents ............................................................. 7
- Create and edit a topic ................................................................. 10
- Link to other topics .................................................................... 11

**Manage content in Ponydocs**
- About branching and inheriting .................................................. 13
- Branch a topic ........................................................................... 14
- Branch multiple topics at once .................................................... 14
- Inherit content into a new version .............................................. 15
- Permissions in Ponydocs ............................................................. 16
Welcome to Ponydocs

What is Ponydocs?

Ponydocs is Splunk's MediaWiki-powered documentation platform. We build and deliver our official product documentation on this platform (you're looking at it right now), and have made it available to the community at large as an open source project.

Check out this blog post about the Ponydocs beta release by our lead developer and Web Dev manager, Ashley for an overview.

You can find the public source for Ponydocs at http://splunk.github.com/ponydocs/

Note: Installation and configuration information is currently packaged with the code.

Why open source?

We are making Ponydocs available to the open source community because we think our docs platform is awesome, and also because we want to take advantage of the open source community's powers of bug detection and resolution (through issue tracking), testing (through others eating our cooking) and potential future hire determination (through excellent individuals providing quality patches to the project).
Create content in Ponydocs

Overview of objects in Ponydocs

Ponydocs objects are the containers and content in the documentation system. This topic provides a high-level overview of what they are and do. Procedures for creating and working with these objects are in later topics.

Products

Ponydocs supports multiple products, each with their own versioning scheme. A product contains a version scheme, manuals (each with a table of contents), and topics. You must define a product before you can create a version scheme for it and add content (manuals, topics, etc). A product has a short name and a display name. The short name is used in all URLs for that product and cannot be changed once the product is created. The display name is the full name of the product, and can be changed as desired.

For example, you might have a product called "App for Sparkly Unicorn Data". You probably don’t want all the URLs to all the topics in all the manuals about that app to have that full string in them, so you might choose "UnicornData" as the short name, resulting in URLs like:

http://<yourponydocsserver.com>/UnicornData/Install/InstallUnicornData

Versions

A product must have a version scheme before you can create manuals within it. A version scheme just means it has to have one or more versions available to which content can belong, or apply. Version schemes can contain alphanumeric text, so you can have things like "2.0beta" "v.098" and so on. Typically, the versions defined for a product in Ponydocs map to (or match exactly) the versions of the product or system being documented.

Version tags

Ponydocs uses version tags to denote what topics, manuals, and tables of contents belong to which version of the product. Version tags are a type of MediaWiki Category: tags, except that there’s no hierarchy of versions, and the notation adds a V into the middle. In Ponydocs, we say that something (an
object) is 'tagged' with a version. An object can (conceptually) have any number of version tags. A version tag (in this case for version 1.0Beta2) looks like:

```
[[Category:V:1.0Beta2]]
```

**Version state**

To control access to certain versions of your documentation, Ponydocs provides the concept of "version state". In Ponydocs, a given version can have three possible states: released, unreleased, or preview:

- **Released**: objects tagged with a version that is in the "released" state are visible to anyone who visits the site. Users do not have to be logged in to see this content.
- **Unreleased**: objects tagged with a version of "unreleased" are only visible to logged-in users belonging to the 'employees' group in the underlying MediaWiki deployment. The Splunk docs team uses this state when we are working on documentation for an upcoming release.
- **Preview**: objects tagged with a version of "preview" are visible to logged-in members of the "employees" group in the underlying MediaWiki deployment, as well as to logged-in members of the "preview" group. The Splunk docs team uses this state to provide beta documentation access to designated customers without allowing other users to see this preview content.

The state of a version can be changed. For example, over the course of a development/release cycle of the product you are documenting, you may want to transition the version of the documentation from unreleased, to preview, and then to released when the product ships.

**The concept of "latest" version**

URLs in Ponydocs include a version string so that users can link directly to a specific version of a product's documentation. URLs that include version information look like:

```
http://yourponydocsserver/UnicornData/2.5/Install/SupportedPlatforms
```

However, you usually want users to go to the latest (released) version of the documentation, and you don't want to keep having to update a URL you've posted somewhere, so Ponydocs also supports the concept of a "latest" version. You can replace the version number in any URL with "latest" and that URL will always go to whatever the latest version in "released" status of that object is. Like so:
Additionally, the first time a user visits a given product on your Ponydocs site, that user will be placed into the "latest" version as well (as well as any time that user's cookie expires). Similarly, if a user selects a version other than the latest one, that user will be returned to that version the next time he or she visits the site and product, unless their cookie expires.

**Manuals**

A product can have one or more manuals associated with it. Like products, a manual has a short name and a display name, to help keep URLs short. In the example URL above, the manual's short name is "Install", and the display name might be something like "Installing the App for Sparkly Unicorn Data". The manuals for a given product are listed on that product's landing page.

**Note:** Manuals don't technically have version tags applied to them; the version tags instead apply to instances of the tables of contents for that manual (because the same manual can change a lot between versions).

**Tables of contents (TOC)**

A manual is built from a TOC, and represents the structure of that manual for a given version or range of versions. TOCs are tagged with versions, but not all topics within a TOC have to apply to all the versions to which the TOC applies.

**Topics**

A manual can have one or more topics within it. Topics are tagged with versions. Topics contain the actual documentation/content that visitors to the site see.

**Create a product**

Before any other objects can be created, at least one product must exist in the Ponydocs system.

**Note:** In addition to defining the product and its description as explained below, the administrator of your Ponydocs implementation must also must configure the value of $ponyDocsProductsList to match this list of products exactly.
To access the Products page, use the following URL, substituting the server name for your own Ponydocs server in place of `<yourponydocsserver>`.

http://<yourponydocsserver>/Documentation:Products

This page drives the listing of available products on the Documentation landing page.

**Things to know when creating a product**

- The products appear on the Documentation landing page in the order in which they are defined on this page.
- A product can designate another product as its parent, so you can have suites or collections of products.

**Syntax**

Create a product using the following syntax:

```
{{#product:productShortName|displayName|description|parent}}
```

- `<productShortName>` is what gets used in the URL for that product.
- `<displayName>` is what is shown in the list of products the end user sees on the Documentation landing page.

**Note**: Do not use URL-unsafe characters such as : (colon), ? (question mark), / (forward slash), or spaces in the short name.

The `description` and `parent` fields are optional.

When you save the Products page, the products you add are not displayed on the Documentation landing page until you create at least one version, manual, and topic for each product.

**Create a version**

Once you've created a product, the next step is to create one or more versions that apply to this product.

To access the Versions management page, use the following URL, substituting the server name for your own Ponydocs server in place of
This page drives the dropdown list of versions available for the defined product in addition to the state of that version, which designates whether the end user can or cannot see content tagged with that version. For more information about version state, refer to this overview of versions.

Note: If there are no manuals for the selected version the version will NOT appear in the drop-down list.

**Things to know when creating a version**

- A version must be defined here before any topics may be tagged with that version.
- The versions can be any text or numeric string, such as 1.1.4 or Beta or 1.0beta.
- The versions should be listed in release order; that is, the newest/latest versions should be closer to the bottom of this page as this is where you would typically add them.
- Change the status of a version by editing this file (from unreleased to released, for example).
- To edit the versions page for a given product, you must have "docteam" permissions for the specific product for which you're adding a version. For example, if you want to edit the versions page for the Unicorns product, you must have the checkbox checked for "unicorns-docteam" on the permissions page. If you don't see a link to create/edit the versions page, you probably don't have that permission.

**Syntax**

Create a version using the following syntax:

```
{{#version:<name>|[released|unreleased|preview]}}
```

For example, the entry for versions 1.0Beta1 and 1.0Beta2, both in preview status, would look like this when you have the Versions management topic open for editing:

```
{{#version:1.0Beta1|preview}}
```
When you save the Versions management page, the version you added is created in Ponydocs.

Create a manual

Once you’ve created a product and a version for the product, you can create manuals.

To access the Manuals management page, use the following URL, substituting the server name for your own Ponydocs server in place of <yourponydocsserver>:

http://<yourponydocsserver>/Documentation:<product short name>:Manuals

The order in which the manuals are defined in the Manuals management page defines the order in which they are displayed in the list of manuals in the left hand navigation.

Note: A given manual will only appear in this list if there is a table of contents page defined for the manual for the currently selected version of documentation.

Syntax

Use the following syntax to define a new manual:

{{#manual:<manualshortname>|<manual display name>}}

- <manualshort name> is what gets used in the URL for that manual.
- <manual display name> is what is shown in the list of manuals the end user sees.

IMPORTANT: Do not use URL-unsafe characters such as : (colon), ? (question mark), / (forward slash), or spaces in the short name.

Create a table of contents

Once you’ve created a product, a version for the product, and a manual for that product, you can create and build a table of contents (TOC). The TOC file for a
manual defines what topics belong to that manual.

To create a TOC for a new manual (one that doesn't exist yet in a given version), navigate to the Manual management page. To access the Manuals management page, use the following URL, substituting the server name for your own Ponydocs server in place of <yourponydocsserver>:

http://<yourponydocsserver>/Documentation:<product short name>:Manuals

This page displays a list of all the currently-defined manuals available for this product:

- If a manual has a TOC in the version you're currently in, you'll see that manual's name as a link.
- Any manuals that don't have a TOC in the version you're currently in will have a link beneath the manual name that says "Click manual to create TOC for current version".

Make sure that you have selected the version for which you want to create the TOC from the drop-down menu, and then click on the manual link from the Manuals management page. This creates an empty TOC definition page for the version you have selected from the drop-down.

This TOC will include the tag(s) the version(s) that this TOC applies to, they look like this:

[[Category:V:<short product name>:<version>]]

Don't edit these version tags by hand unless you know what you're doing. Keep the version tags at the bottom of the page and add your topics above them.

**What's in a TOC?**

TOC files contain three things (besides version tags): a manual description, chapter names, and topic names:

- Manual description: This is the text that is shown with the manual title on the product's landing page.
- Chapter name: This is an arbitrary text string. The chapter name is shown in the left-hand navigation between the groups of topics in each chapter. Chapter names can contain any ASCII character, since they are not included in any URL. You must have at least one chapter name in a TOC.
• Topic name: This is the string that will appear in the URL path for that topic (in addition to the product short name, manual short name, and version). Because the topic name is used in the URL, it can only contain alphanumeric characters, and any spaces you include will be removed when the topic is created. The topic name is what is used as the initial title and H1 of the topic (and is displayed in the left-hand navigation when you view the topic), but once the topic is created, you can edit the H1 and change it. At that point, you can use any characters you want in the H1, and your changes will be reflected in the left-hand navigation.

Things to know when creating and editing a TOC

• Put the manual description at the top of the file.
• Add chapter name as free text, they can contain any ASCII characters you like.
• You must have at least one chapter name for the manual to display.
• Add topic names under their respective chapter names.
• Topic names can only contain alphanumeric characters, and spaces are removed when the TOC is saved and the file is created. You can't use question marks, colons, hash signs, or slashes in filenames, but you can add them back in in the H1 within the topic if needed.
• The same topic cannot appear more than once in a single TOC page.
• Remember to use * in front of each topic you add--this is different from the versions and manuals list pages, and from the manual description.
• Topics you add to the TOC file are created when you save the TOC file. Ponydocs will automatically create them and tag them with the correct version(s).

Syntax

The syntax for adding the manual description is:

```
{{#manualDescription:Manual Description}}
```

where "Manual Description" is the string of text you want displayed.

Note: You must add this to each version of the TOC for a given manual.

The syntax for topic names is:

```
* {{#topic:TopicName}}
```
As an example, the TOC for the Sparkly Unicorn Data User Manual might look like this:

{{#manualDescription:This manual introduces Sparkly Unicorn Data, a sparkle-based version of ponies. It explains what you need to do to get started, add ponies, hug, and share unicorns.}}

**Introduction**
* {{#topic:About Sparkly Unicorn Data}}
* {{#topic:Sparkly Unicorn Data FAQ}}
* {{#topic:Learn more and get help}}

**Splunk concepts and how-tos**
* {{#topic:Inputs and projects}}
* {{#topic:Sources and source types}}
* {{#topic:What Sparkly Unicorn Data does with your data}}

**Before you start**
* {{#topic:About ponies}}
* {{#topic:Ways to get data into Sparkly Unicorn Data}}
* {{#topic:Sparkly Unicorn Data performance and sizing}}
* {{#topic:Information to have ready before you start}}

**Get started**
* {{#topic:Create and activate an account}}

**Add data**
* {{#topic:Add ponies over TCP or UDP}}
* {{#topic:Upload a unicorn}}
* {{#topic:Add ponies using the REST API}}
* {{#topic:Add ponies from a Webhook}}
* {{#topic:Pick a mane type}}
* {{#topic:Set the colors and sparkles}}

**Manage, and share a unicorn**
* {{#topic:Share your unicorn}}
* {{#topic:Troubleshoot a unicorn}}
* {{#topic:Delete ponies from a unicorn}}

### Create and edit a topic

Once you've **created a product**, a **version** for the product, a **manual** and a **table of contents** for that product, you can create **topics**.

When you add a topic name to a **TOC file** and then save that file, the topic you named is created. To access it for initial editing, click on the topic name from the TOC file.
Topics and version tags

By default, when a topic is created by saving a TOC file, it is automatically tagged with all the versions that the TOC file was tagged with. If a topic only applies to a subset of the versions supported by the TOC, remove the other version tags, being careful not to remove or leave any extra brackets, and then re-save the topic before proceeding.

Things to know when creating or editing a topic

• The H1 is by default the original topic name you used when you created it in the TOC file.
• You can’t use question marks, colons, hash signs, or slashes in filenames, but you can add them back in in the H1 within the topic if needed.
• You can change the H1 to be something else (for example, to re-include : or ? symbols if they’re needed) and this will also be reflected in the left-hand navigation shown to the user.
• A topic can only have one H1 in it.

Markup in Ponydocs

Ponydocs is built on top of a standard MediaWiki installation, and uses the standard MediaWiki markup.

Adding graphics

To include a graphic in a topic, use the Upload file link in the Toolbox, then link to that graphic using the MediaWiki linking syntax.

Link to other topics

Ponydocs has augmented the standard MediaWiki linking syntax to support the concept of products, manuals, and versions.

Link to a topic in the same manual

To link to a topic that is in the same manual as the one you’re editing, use the standard MediaWiki syntax for linking:

[[TopicName|<text you want to appear as the link>]]
Link to a topic in another manual

To link to a topic that is not in the same manual, use the following syntax:

```
[[Documentation:<ShortProductName>:<ShortManualName>:<Topic>|<text you want to appear as the link>]]
```

Things to know about linking to other topics

- The links are case-sensitive. You must use the exact capitalization for the short product name, short manual name, and topic name.
Manage content in Ponydocs

About branching and inheriting

When you create a new version for a product in Ponydocs, you can choose to inherit all the content from the previous version into the new version, or you can choose to branch some or all of the content instead.

- If you inherit content from one version to the next, the version tag for the new version is simply added to all topics that belong to the product you're versioning. Any changes you make to that content going forward applies to the new version and the previous version.
- If you branch content from one version to the next, a new copy of the topic(s) is created and tagged with the new version tag. Any changes you make to the topic tagged with the new version are not applied to the previous versions.

What you choose to do depends a lot on the situation:

A maintenance release

Let's say you are producing documentation for Fflanda, a software product, currently shipping version 2.7.4, and version 2.7.5 is about to come out. It's a maintenance release--just a few things are changing in the software, so the documentation isn't changing a whole lot for the release. When you add version 2.7.5 to the Fflanda product in Ponydocs, you decide to inherit all the content from version 2.7.4 into version 2.7.5, and then just go into the few topics that changed, and branch them individually so you can make changes to the content that apply only to versions 2.7.5 and later.

A minor release

It's a few months later at Fflandacorp, and version 2.8 is about to come out. There are a few feature changes, so two of your manuals are changing a lot, although the rest of the manuals only have a few small updates. When you add version 2.8 to the Fflanda product, you decide to branch both of the manuals that are changing in their entirety, but inherit the content for the rest of the manuals.
A major release

Fflanda is about to ship version 3.0, a major release with a lot of new features and changes. The documentation has also changed; almost every topic has been updated, and you’ve restructured and reorganized most of the manuals. When you create version 3.0, you decide to branch all the content.

Inserting a version

At some point you may have to insert a version in between two existing, already-populated versions. This might occur because you're maintaining and supporting many versions of a product simultaneously, and a major bug might get fixed on an older codeline (resulting in a change that applies to subsequent versions). You can insert a version into a range of inherited versions and then split/branch at that point.

Branch a topic

Before you branch a topic, make sure the version you want to branch to exists.

1. Select the version you want to branch from and navigate to the topic you want to branch.

2. Click Branch. The Branch and Inheritance Console is displayed, with the topic you’re in and the version you’re branching from preselected.

3. Choose the Target version, the version you are branching to, from the drop-down menu and click Continue to Manuals.

4. Make sure the Branch option is selected for the topic you’re branching and click Process Request.

The topic is branched.

Branch multiple topics at once

Before you branch, make sure the version you want to branch to exists.

To branch multiple topics, or entire manuals in a single operation:
1. Navigate to the Special:BranchInherit page on your Ponydocs instance.

2. Select the product and versions you're branching from and to, and click Continue to Manuals.

3. Select the manual or manuals that contain topics you want to branch. You'll be able to either select an entire manual or a subset of its topics on the next page.

4. If you're planning to branch the majority of the topics in a manual, select Branch as the default action. You'll be able to change the action on individual topics on the next page. If you're branching most topics in one manual and a minority in another, you might want to accomplish this in two separate operations.

5. Click Continue to Topics.

6. Review the list of topics and ensure the desired action is selected for each topic, then click Process Request.

The topics are branched.

Inherit content into a new version

In most cases, when you create a new version of an existing product, the easiest thing to do is inherit all the content from the previous version into the new version, and then branch what needs branching in subsequent operations.

Before you inherit content into the new version, make sure the version you want to inherit to exists.

1. Navigate to the Special:BranchInherit page on your Ponydocs instance.

2. Select the product and versions you're inheriting from and to, and click Continue to Manuals.

3. Select the manual or manuals that contain topics you want to inherit. Typically when you're inheriting content into a new version, you'll want to select all the manuals from the previous version.

4. Click Continue to Topics.
5. Review the list of topics and ensure the desired action is selected for each topic, then click **Process Request**.

The topics are branched.

### Permissions in Ponydocs

Ponydocs uses a combination of the standard MediaWiki user rights system and version management to support per-product and per-version access to your content.

### User access to content and editorial tools

As discussed in the overview of objects in Ponydocs, a given version can be in one of three states: released, unreleased, or preview. The Splunk documentation uses these states in the following way:

- All Splunk employees are members of an 'employees' group in the MediaWiki user rights system, and can view and edit any product's documentation content in any version state.
- Docteam members are additionally members of groups granting per-product editorial privileges allowing them to branch and inherit content, as well as create and edit TOC files and manuals.
- (Non-employee) users can see any content that is tagged with at least one version that is in 'released' state. These users can be logged in to the site or not.
- Logged-in users can be made members of per-product 'preview' groups, granting them additional permissions to see content tagged with at least one version that is in 'preview' state.

You can create user groups in the MediaWiki user rights system to meet your organization’s product- and user-specific requirements, and map them to access to the three version states.