# “Technically” Anyone Can Use This Documentation

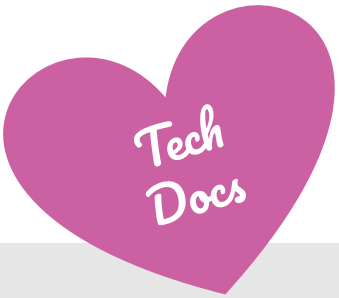
## How the Average Archivist Can Make Use of the ArchivesSpace Technical Documentation

Presented by Liz Caringola
on behalf of the Tech Docs Subteam

ArchivesSpace Virtual Member Forum - April 5, 2023

# Goals

1. Share basic information about the Tech Docs.
2. Demonstrate how the Tech Docs can support you in communicating with your IT folks.
3. Pique your interest so that you explore the Tech Docs on your own afterwards.




I have three goals for this presentation.

First I want share some basic information about the Tech Docs, what kind of information you can find in them, and where to find them.

I also want to demonstrate how the Tech Docs can support you when you need to communicate any changes you want to make to your ASpace instance. Maybe you'll even learn about some setting that you didn't realize you could change.

And most importantly, I hope this presentation piques your interest and gives you a reason to read through the Tech Docs on your own if you never have.

# What is the Technical Documentation?

- The Tech Docs contain information about ArchivesSpace as a software application.
- ASpace is open source, and the Tech Docs are open for anyone to use.
- You might turn to the Tech Docs when you are…
  - Evaluating the technical requirements and capabilities of ASpace
  - Installing, configuring, and/or maintaining your ASpace instance
  - Creating plug-ins or contributing to the ASpace core code

https://archivesspace.github.io/tech-docs/

The ArchivesSpace Technical Documentation contains information about ArchivesSpace as a software application. This is different from the User Manual, which provides information on how you as an archivist can use ASpace once it's set up and running.

ArchivesSpace is an open source application, which means that anyone can view and use the code to set up ArchivesSpace and make contributions to the core code. In order to facilitate anyone being able to use ArchivesSpace and improve upon the existing code, the Technical Documentation is available to everyone, meaning you do *not* have to be an ArchivesSpace member to get access to the Tech Docs.

In our Tech Docs GitHub, there are three main reasons listed for how one would use the Tech Docs:

1. When you are evaluating the technical requirements and capabilities of ASpace;
2. Installing, configuring, and/or maintaining your ASpace instance; or
3. Creating plug-ins or contributing to the ASpace core code.

However, even if you will never do any of those things, you can still use the Tech Docs! The Tech Docs can be a great tool in bridging the gap between the archivists who need to use the ArchivesSpace and the people with the technical expertise to set up and maintain your ASpace instance. So let's talk more about that…

# Talking to IT

You need to:

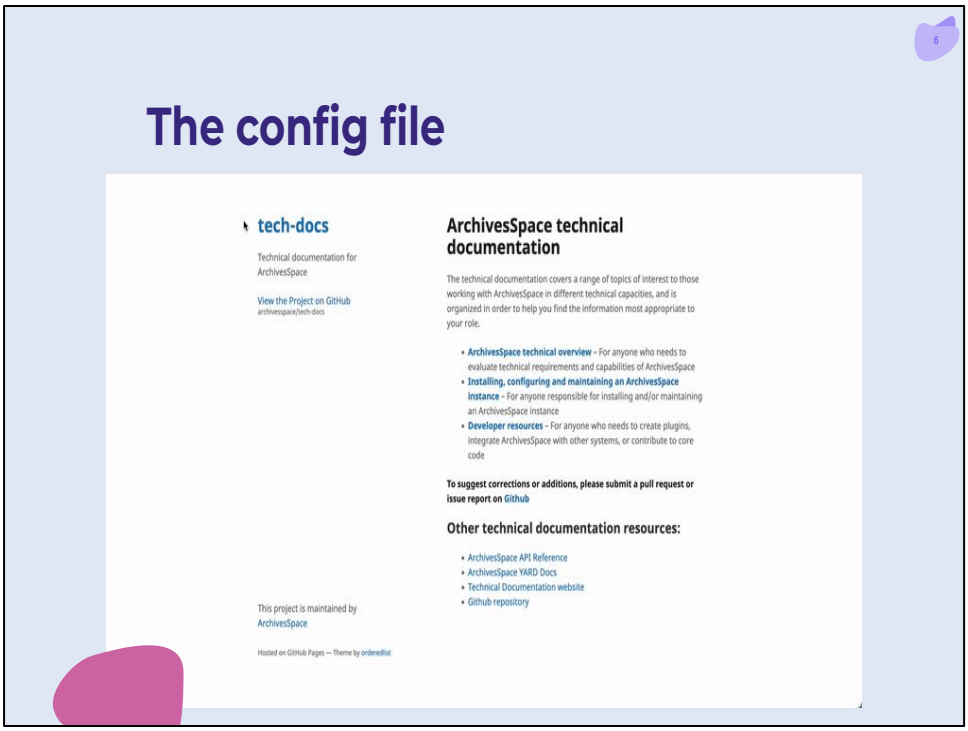
- Have some foundational knowledge of ArchivesSpace's technical requirements, architecture, and configuration
- Have the language to effectively communicate your requests
- Be able to refer others to the Tech Docs

If you are the main point of contact between your archives and your IT department, sysadmin, developers, and/or hosting provider, then you need to be able to effectively communicate your needs to them. This communication piece can be difficult if you're speaking in "archivist" and the other party is speaking in "tech." To an extent, you'll need to educate yourself about the more technical aspects of the ASpace software. Reading through the Tech Docs is a great way to give yourself that foundation and language to better communicate.

You don't necessarily need to read the Tech Docs in their entirety or memorize everything that's contained in the Tech Docs; you just need to be familiar with the kind of information that the Tech Docs contain so that you can reference it, or point others to it, as needed. And keep in mind that when you're communicating with whoever can do this technical work for you that it can be helpful to meet them where they are and use the terminology that they understand, as opposed to expecting them to do that work of translating concepts.

# Let’s look at some examples...

Let’s walk through a few examples of the kinds of information contained in the Tech Docs and how you might be able to use that information to request minor customizations to your ASpace instance.




For our first two examples we're going to talk about the "config" file (short for configuration file). You might have heard about the config file in forums like this one or on the member listserv where people talk about how they've made certain customizations to their ASpace instance. This file contains a lot of ASpace's default settings and allows you to to change them. Information about the config file can be found in the Tech Docs on the page called "Configuring ArchivesSpace." To get to this page from the Tech Docs home page, click on "Installing, configuring and maintaining an ArchivesSpace instance" and then "Configuring ArchivesSpace." (How to navigate to this page is shown in the GIF on the slide.)

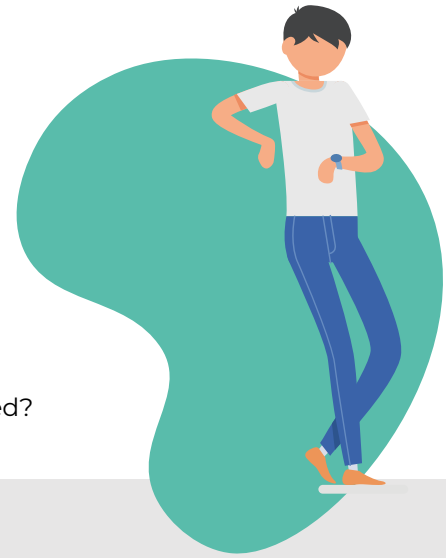
Reading through this page will give you a sense of what kinds of default settings and behavior can be changed. There's a lot of information on this page, and yes some of it is super technical in nature and goes over my head a bit, but as I continue scrolling down the page, I'm also finding that that I can understand a lot of it. For example, I see that it's possible to hide tabs on the public interface's main horizontal menu ("pui_hide"), change the amount of time before your ASpace session times out ("Session expiration"), change the URL of the "feedback" link in the footer ("feedback"), specify the number of characters that make up a valid top container barcode ("Container management configuration fields"), and more.

Now that we know a little bit about the config file thanks to the Tech Docs, let's see how we could use this info to request a change to our ASpace instance.

# Example: record inheritance

**Behavior you've noticed**

When viewing your finding aids in your ASpace public interface, folders that shouldn't have a Conditions Governing Access note (because you didn't enter one in the staff interface) are displaying a Conditions Governing Access note. You realize that there's a pattern to this behavior and that the folders are displaying the note that you entered for their corresponding series.



**You're wondering...**

- Can this behavior be changed?
- How do I ask?

In this scenario, you, the archivist, have noticed that folders (a.k.a. archival objects) that don't have a Conditions Governing Access note in the staff interface are displaying a Conditions Governing Access in your public interface. It seems like it's the same note that you did enter at the series level (a.k.a. an archival object that is above your folder-level archival objects).

This type of behavior is called "inheritance," meaning that if a certain field is left blank, it will "inherit" it from an archival object that is above it in the resource record hierarchy and display in the public interface by default. However, this setting can be turned off.

# Example: record inheritance

**Record inheritance in public interface**

**AppConfig[:record_inheritance]**

Define the fields for a record type that are inherited from ancestors if they don't have a value in the record itself. This is used in common/record_inheritance.rb and was developed to support the new public UI application. Note - any changes to record_inheritance config will require a reindex of pui records to take affect. To do this remove files from indexer_pui_state

```
AppConfig[:record_inheritance] = {
  :archival_object => {
    :inherited_fields => [
      {
        :property => 'title',
        :inherit_directly => true
      },
      {
        :property => 'notes',
        :inherit_if => proc {|json| json.select {|j| j['type'] == 'accessrestrict'} },
        :inherit_directly => true
      },
      {
        :property => 'notes',
        :inherit_if => proc {|json| json.select {|j| j['type'] == 'scopecontent'} },
        :inherit_directly => false
      },
      {
        :property => 'notes',
        :inherit_if => proc {|json| json.select {|j| j['type'] == 'langmaterial'} },
        :inherit_directly => false
      },
    ]
  }
}
```

**Example support ticket**

For archival objects that do not have a Conditions Governing Access note, please turn off record inheritance in the public interface.

In config/config.rb:

:property => 'notes',
:inherit_if => proc {|json| json.select {|j| j['type'] == 'accessrestrict'} },
:inherit_directly => true

Update inherit_directly to false.

For more information, see “Record inheritance in public interface” on https://archivesspace.github.io/tech-docs/customization/configuration.html.

If you go back to the Tech Docs and the "Configuring ArchivesSpace" page, you’ll see all of the default settings for ASpace that appear in the config file. Scroll down or use ctrl/command + F to search for "inheritance." You'll find a section called "Record inheritance in public interface," which is screenshotted on the left, and it shows you what the default inheritance settings are for archival objects. By default, the Access Note is configured to be inherited (circled in yellow).

Using this information, I formulated my example support ticket on the right in 3 parts. First, I’m stating my request, and I’m using terminology that is specific to ASpace (resource record, archival objects, etc.) and that I’m seeing in the documentation. In the second part, I’m stating the file and the line(s) of code that need to be edited to implement my request, which I’ve just copied and pasted from the Tech Docs. And lastly, I’m pointing my IT folks to the place in the documentation where I got this information and linking to the exact page so that they can easily find more information if they need it.

Hopefully can see how this request would be pretty easy for your developer to act on, compared to how we described this issue in the previous slide. To be clear, I’m **not** saying that the previous slide would have been a “bad” support ticket. If you can’t figure out what’s causing your ASpace instance to behave a certain way, then you have to just explain what you’re observing. My main point here is that if you can’t determine the root cause or how to change a behavior, then whomever you submit the ticket to will need to do some research first and may have additional questions for you before they can change the behavior that you want to change.

# Example: define barcode length


**Desired behavior**

Your archives uses barcodes that are 10 characters in length. You want ArchivesSpace to recognize this and prevent the saving of top container records with barcodes that aren't 10 characters.

**You're wondering "Is this possible?" Yes!**

- An exact length or range can be specified.
- It can also be customized for different repositories.
- If the barcode is blank (0 characters), it will also validate.

In this next scenario, you would like for ASpace to "know" how long your archives' barcodes are and prevent top container records from saving if a barcode with the wrong number of characters is entered. You think this will help to prevent typos if someone is manually keying-in barcodes.

This is possible and is a setting that can be configured in the config file. On the same "Configuring ArchivesSpace" page in the Tech Docs, there is a section called "Container management configuration fields." This section explains that it's possible to configure an exact character length for barcodes, or a range of acceptable character lengths. It's also possible to have different barcode lengths for different repositories within your ASpace instance. Finally, the Tech Docs also say that if the barcode field is left blank (i.e., is 0 characters), ASpace will still consider it to be valid.

Now that you know more about what is possible in terms of barcode validation, you can decide if this is something you want to have configured in your ASpace instance.

# Example: define barcode length

**Container management configuration fields**

**AppConfig[:container_management_barcode_length]**

Defines global and repo-level barcode validations (validating on length only). Barcodes that have either no value, or a value between :min and :max, will validate on save. Set global constraints via :system_default, and use the repo_code value for repository-level constraints. Note that :system_default will always inherit down its values when possible.

```
AppConfig[:container_management_barcode_length] =
{:system_default => {:min => 5, :max => 10},
'repo' => {:min => 9, :max => 12}, 'other_repo' =>
{:min => 9, :max => 9} }
```

**Example support ticket**

I want ArchivesSpace to validate the number of characters in top container barcodes. Barcodes for all ArchivesSpace repositories should be 10 characters in length.

In config/config.rb, add the following:

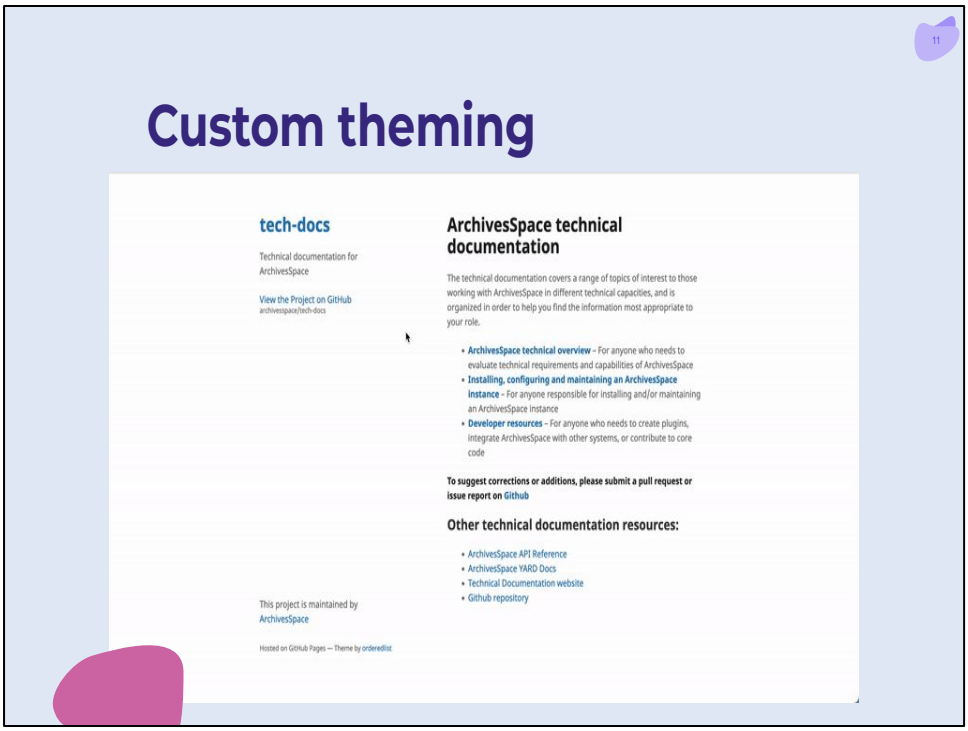
AppConfig[:container_management_barcode_length] = {:system_default => 10 }

For more information, see “Container management configuration fields” on https://archivesspace.github.io/tech-docs/customization/configuration.html.

For the sake of this example, let’s say that we want all barcodes across all repositories to be exactly 10 characters. By asking for this configuration, ASpace will only save a barcode if it is either 10 or 0 characters in length.

In my support ticket, I’ve again included an explanation of my request that uses the language we see in the Tech Docs. I’ve also included what (I think and hope) is a correct line of code to turn on barcode validation. However, this wasn't just a copy/paste job; I had to slightly modify what was in the documentation (screenshotted on the left) since I don’t want a range of character lengths and I don’t need to have different barcode lengths for different repositories.

If you wouldn’t feel comfortable trying to figure out how to edit this line of code, then skip putting that in your ticket. Again, the more info you can provide to your IT folks, they will probably be able to respond more quickly, but don’t stress about it if you’ve hit your limit with what you think you can figure out, or spend too long trying to get it perfect. I felt pretty confident attempting to edit this code and it took me just a minute. Your IT people can definitely use their expertise and the documentation to figure it out.




For our last example, we'll talk about how custom theming can be added to your ASpace instance.

From the Tech Docs home page, click on "Installing, configuring and maintaining an ArchivesSpace instance" and then "Theming ArchivesSpace." (How to navigate to this page is shown in the GIF on the slide.)

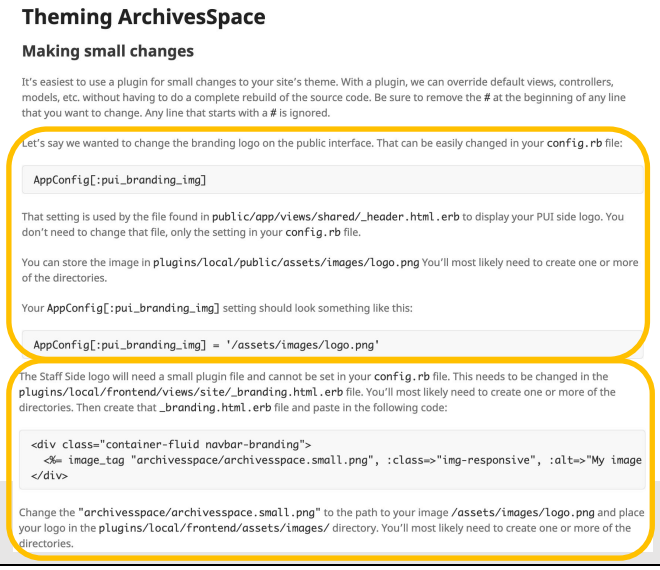
This page tells you how to change the logo on your public and staff interfaces ("Making small changes"), how to add custom stylesheets that can change the look and feel of your ASpace instance so that it can more closely match the look and feel of your institution's website ("Adding CSS rules"), and how to build a new theme from scratch ("Heavy re-theming").

We'll stick with a relatively simple example of how to switch out the ArchivesSpace logo with another image, such as your institution's logo.

# Example: replace logo

**Desired customization**

You want to replace the ArchivesSpace logo with your institution's logo in the staff and public interfaces.



### Theming ArchivesSpace

#### Making small changes

It's easiest to use a plugin for small changes to your site's theme. With a plugin, we can override default views, controllers, models, etc. without having to do a complete rebuild of the source code. Be sure to remove the # at the beginning of any line that you want to change. Any line that starts with a # is ignored.

Let's say we wanted to change the branding logo on the public interface. That can be easily changed in your `config.rb` file:

```
AppConfig[:pui_branding_img]
```

That setting is used by the file found in `public/app/views/shared/_header.html.erb` to display your PUI side logo. You don't need to change that file, only the setting in your `config.rb` file.

You can store the image in `plugins/local/public/assets/images/logo.png` You'll most likely need to create one or more of the directories.

Your `AppConfig[:pui_branding_img]` setting should look something like this:

```
AppConfig[:pui_branding_img] = '/assets/images/logo.png'
```

The Staff Side logo will need a small plugin file and cannot be set in your `config.rb` file. This needs to be changed in the `plugins/local/frontend/views/site/_branding.html.erb` file. You'll most likely need to create one or more of the directories. Then create that `_branding.html.erb` file and paste in the following code:

```
<div class="container-fluid navbar-branding">
  <%= image_tag "archivesspace/archivesspace.small.png", :class=>"img-responsive", :alt=>"My image
</div>
```

Change the `"archivesspace/archivesspace.small.png"` to the path to your image `/assets/images/logo.png` and place your logo in the `plugins/local/frontend/assets/images/` directory. You'll most likely need to create one or more of the directories.

What's important to note about this customization is that it's a different process to replace the logo on the staff interface and the public interface, so want to include directions on how to do both (circled yellow in the screenshot).

# Example: replace logo

**Example support ticket**

Please update the logos on the ArchivesSpace public and staff interfaces from the default ASpace logo to the UMD Libraries logo (logo.png - file attached).

To update the public interface logo, put the logo file in the plugins/local/public/assets/images/ directory. In the config.rb file, update the location/file name of the logo here:

```
AppConfig[:pui_branding_img] = '/assets/images/logo.png'
```

To update the staff interface logo, put the logo file in the plugins/local/frontend/assets/images/ directory. In the plugins/local/frontend/views/site/_branding.html.erb file, add the following:

```
<div class="container-fluid navbar-branding">
  <%= image_tag "/assets/images/logo.png", :class=>"img-responsive", :alt=>"My image alt text" %>
</div>
```

For more information, see “Making small changes” on https://archivesspace.github.io/tech-docs/customization/theming.html.

In my example ticket, I included both sets of directions, one for changing the logo on the public interface and one for the staff interface. I also included where a copy of my logo image should be saved in ASpace’s file directory, since changing the code won’t do anything if the file isn’t saved where ASpace needs it to be saved.

# Links and contact

Tech Docs home page

https://archivesspace.github.io/tech-docs/

Tech Docs GitHub

Go to this version of the docs if you want to submit an issue or pull request.

https://github.com/archivesspace/tech-docs

# Tech Docs Subteam, 2022-2023

Rachel Searcy, Lead
Jenna Silver, Vice Lead
Austin Munsell
Liz Caringola
Blake Carver
Mark Cooper

That brings us to the end of the presentation. The Tech Docs Subteam and I hope that I've inspired you to take a look at the Tech Docs, especially if you never have before. This first link will take you to the Tech Docs web pages that you've seen in the screenshots of this presentation.

If you have any feedback on the Tech Docs, want to suggest an area for improvement, found a broken link, etc., please go to the second link on the slide, which is the Tech Docs GitHub, and submit an issue. You don't need to know how to use GitHub to submit an issue; it's basically like submitting a comment, so please don't be afraid to do that. If you are comfortable with GitHub, you could submit a pull request to directly edit the documentation, which the Tech Docs Subteam will review and probably approve.


Presentation template/graphics from:
https://slidecoretemplates.com/producto/pointer-seo-strategy-plan-free-presentation-template/