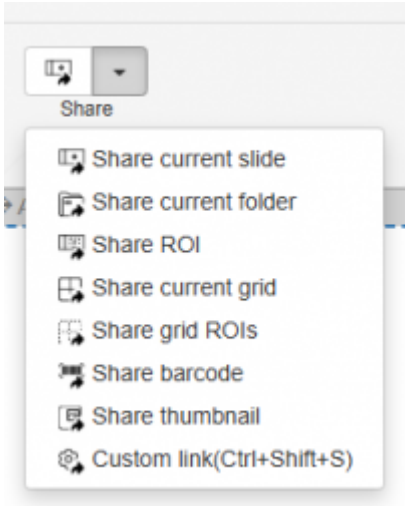
Linking and sharing (automation)

Sharing content is arguably one of the most important applications of digital pathology, if not for the Web in general.

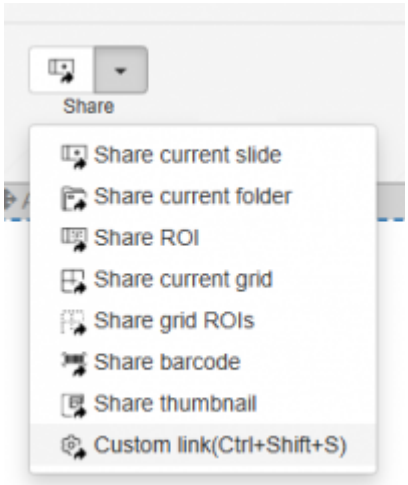


PIMS LS allows you to share content in a variety of ways. There is a dedicated group for sharing content on the ribbon:

When you just want to share what you're currently looking at, chances are that you can get by with one of the quick share buttons:

- If you want to share the current folder you're navigating, click on the "Share folder" button
- If you want to share the current slide that you're looking at, click on the "Share slide" button
- If you want to share the current grid that you're looking at, click on the "Share grid" button
- Etc.

If you want more control over what and how you're sharing content, you can click on the final "Share" button of the group. You could say that that's our "universal" share button.



It allows for further customization of your share link, including:

- Password-protect your link
- Expire the link (e.g. students can only access it for the duration of a test)
- Include or exclude annotations from the shared link

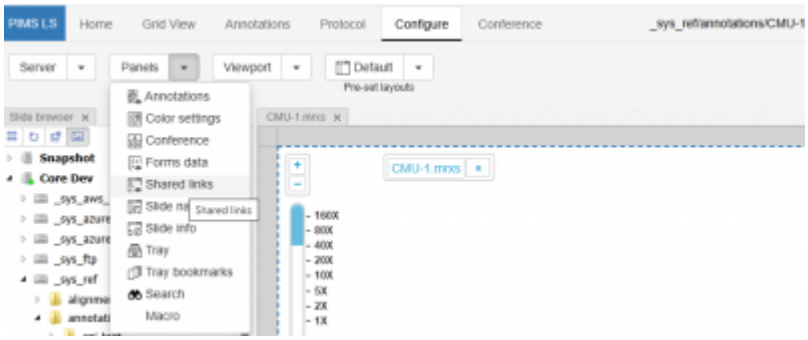
- Use a QR code instead of a plain text link
- Etc.

Share administration

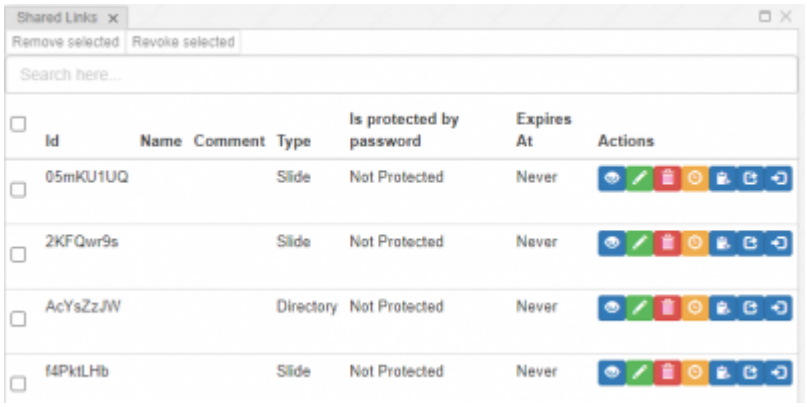
We've worked hard on making the sharing concept in PIMS LS broadly applicable to a variety of content. We've also worked on making it easy to share content.

So with all this sharing going on then, it's only natural to be asking after a while "wait, what am I actually sharing?".

On the configure tab, in the "Panels" group, you can active the "Shared links panel"

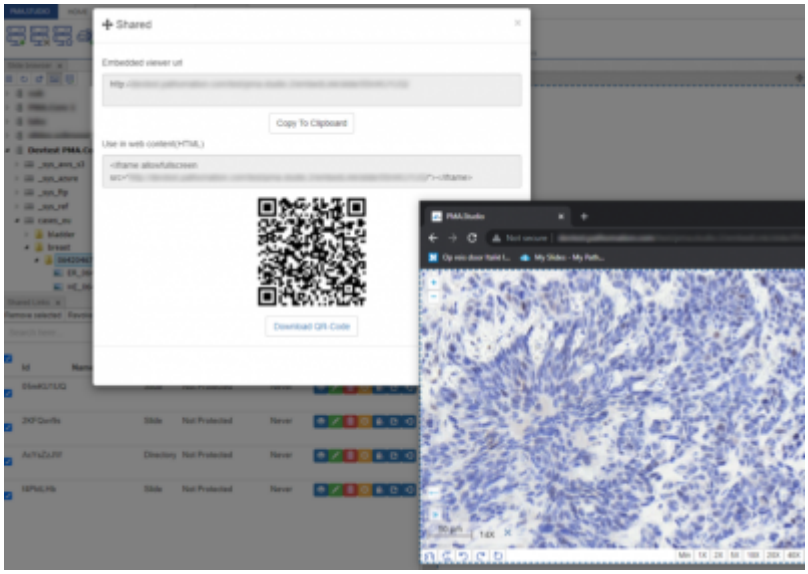


Once clicked, you get a new panel with an overview of everything you've shared so far.



The buttons behind each link allow various operations.

One application of this is to recycle a share link and re-use as you see fit.

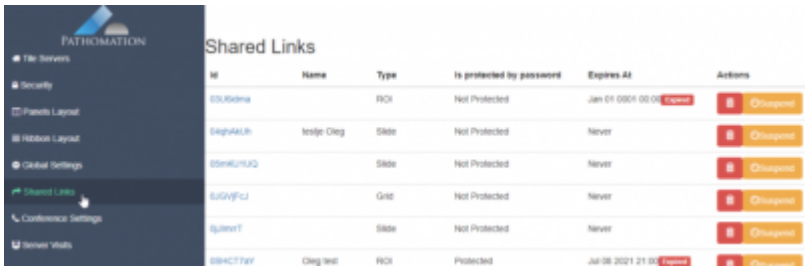


You can also (temporarily) invalidate links, or delete them altogether.

The history is linked to your PMA.core login, so if at first you don’t see anything, make sure that you’re connected to the PMA.core instance for which you expect to see share links.

Monitoring

On the back-end of PIMS LS, administrators can get an overview of all created shares across all users. They can also use this view to temporarily suspend or even delete shares.



Automation

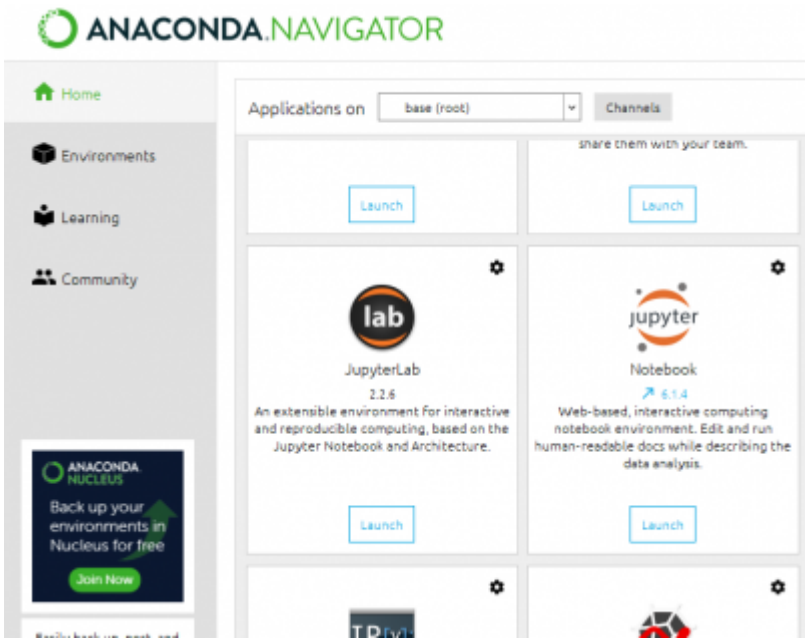
While we highly advocate the implementation of the PMA.UI framework in third-party software like (AP)L(I)MS, PACS, VNA, and other digital pathology consumers, we realize that this is not trivial. In a proof-of-concept phase, all you may want to do is show a button in your own user interface that then subsequently just pops up a viewport to the content that you want to launch. Easy-peasy, as they say.

Let’s say that you have an existing synoptic reporting environment that looks like this:

In order to convince your administration that adding digital pathology to it is a really good idea, you want to upgrade the interface to this:

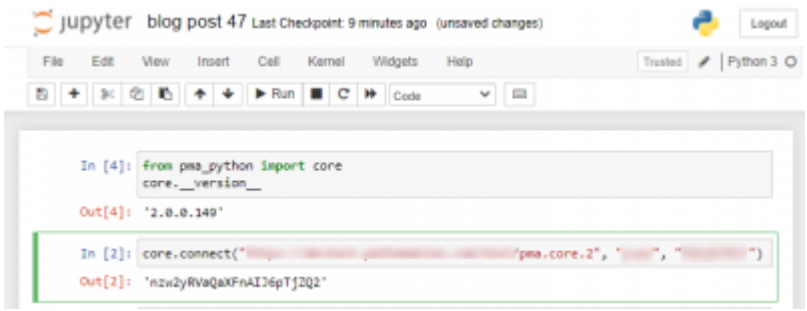
With PIMS LS, you can now get exactly this effect.

Let’s switch to Jupiter to see how this works:



First some homekeeping. We import the pma_python core module, and connect to the PMA.core instance that holds our slide.

Our slide is stored at “cases_eu/breast/06420637F/HE_06420637F0001S.mrxs”. Let’s make sure that the slide exists in that location by requesting its SlideInfo dictionary:

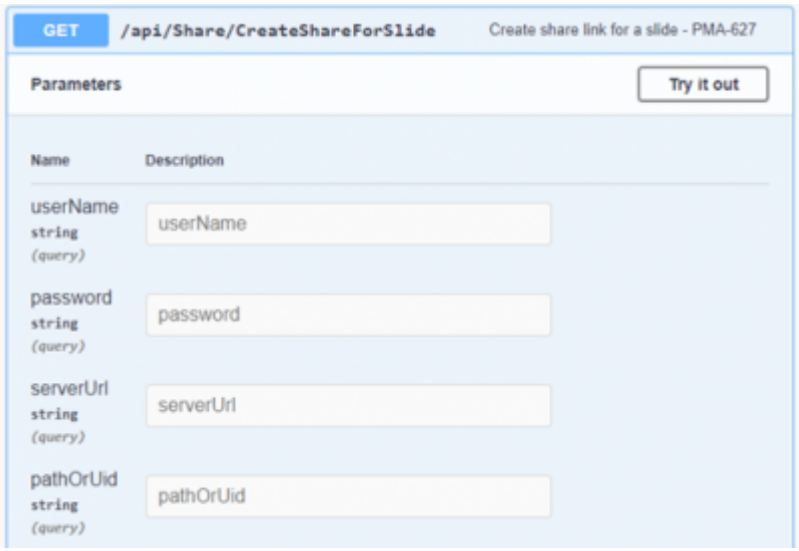


Alternatively, we can also write some exploratory code to get to the right location:

The PIMS LS API

Ok, we’ve identified our slide. Now let’s go to the PIMS LS. Unfortunately, PMA.python doesn’t have a PIMS LS module yet, so we’ll have to interface with the API directly for the time being.

The back-end call that we need is /API/Share/CreateLinkForSlide and takes on the following parameters:



We create the URL that invokes the API by hand first. We can do this accordingly:

```
In [3]: from pma_python import pma

        server = "https://server/"
        baseUrl = server + "pma.studio.2"
        apiCall = "/api/Share/CreateLinkForSlide"
        userName = "user1"
        password = "user1"
        serverUrl = server + "pma.core.2/"
        slideref = "cases_eu/breast/06420637F/HE_06420637F0001S.mrxs"
        pathOrUid = core.get_uid(slideref)

        finalUrl = (baseUrl + apiCall +
            "?userName=" + userName +
            "&password=" + password +
            "&serverUrl=" + pma._pma_q(serverUrl) +
            "&pathOrUid=" + pma._pma_q(slideref))

        print(finalUrl)
```

Never mind that pma._pma_q() method that we use. It's a fast and easy way for ourselves to properly encode HTTP querystring arguments. You're free to piggy-back on ours, or use your own preferred method.

After execution of the code, you get a URL that looks like this:

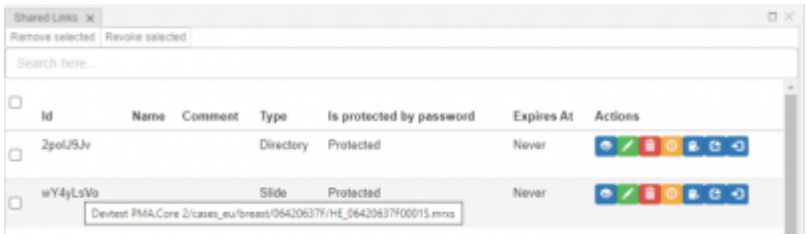
https://yourserver/pma.studio.2/api/Share/CreateLinkForSlide?userName=user1&password=user1&serve.rUrl=https%3A%2F%2Fyourserver%2Fpma.core.2&pathOrUid=IRVGLPWBFT

The URL by itself doesn't do anything, but create the share link. So you still need to invoke it. You can do this by either copying the URL to a webbrowser, or by invoking it from Python as well:

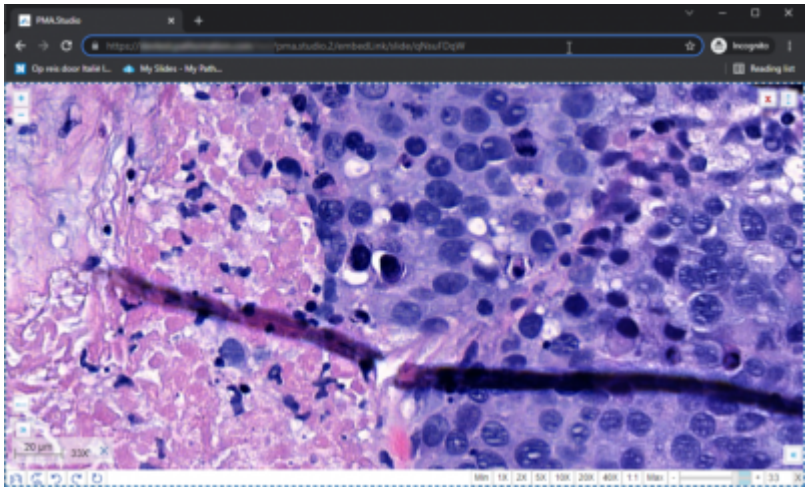
```
In [5]: from urllib.request import urlopen

        shareLink = urlopen(finalUrl).read()
        print(shareLink)

        b'"https://[redacted]/test/pma.studio.2/embedLink/slide/qNsuFDqW"'
```

Again: it's the return result from the URL that you want to distribute to others and not the initial URL.

To confirm that it worked, you go back to PIMS LS and check your panel with the share link overview:



But you can also just pull up the resulting URL in a new browser window:



Yay, it worked!

Automating folders

You can also create links that point to folders.

That slide that we just referenced? It turns out to be an H&E slide, and along with the other slides in the folder, actually comprises a single patient case.

So you can emulate cases by organizing their slides per folder, with each folder representing a case. Your hierarchy can then be:

[patient]

[case]

[slides]

Say that we want to offer a case-representation of breast cancer patient 06420637F. We use Share/CreateLinkForFolder and point to a folder instead of a slide:

```
In [8]: from pma_python import pma

server = "https://server/test/"
baseUrl = server + "pma.studio.2"
apiCall = "/api/Share/CreateLinkForFolder"
userName = "b_user1"
password = "user1"
serverUrl = server + "pma.core.2/"
folderref = "cases_eu/breast/06420637F"

finalUrl = (baseUrl + apiCall +
    "?userName=" + userName +
    "&password=" + password +
    "&serverUrl=" + pma._pma_q(serverUrl) +
    "&path=" + pma._pma_q(slideref))

print(finalUrl)
```

The result again appears on the PIMS LS side. And clicking on it results in a mini-browser interface:

What’s next

After PMA.core, we’re starting to provide back-end API calls into PIMS LS as well. Even though as we prefer developers to integrate with PMA.UI directly, there are scenarios where automation through PMA.UI makes sense. When you’re in one of the following scenarios when:

- PIMS LS is your main cockpit interface to work with slide content, but there are a few other routes (like an intranet) through which you want to provide quick access to content, too.
- You have an (AP)LI(M)S, PACS, VNA, or other system and you’re in a PoC phase to add digital pathology capabilities to your own platform, PIMS LS automation may be a quicker route to go than adapting our SDKs.

Do keep in mind however that we’re providing the PIMS LS back-end mostly for convenience, at least for the time being. There are any number of ways in which you may want to integrate digital pathology in your infrastructure and workflows. For a high level of customization, you’re really going to have to move up to PMA.UI, as well as a back-end counterpart like PMA.python, PMA.php, or PMA.java.