Document-athon 2023 writeup

Introduction

Writing good documentation is an art. Like any art, getting started to produce something needs a source of inspiration. Also, the work of art, in our case a set of documentation about our set of tools: EPICS and associated software, is created to convey something; and in that sense, serve a purpose.
In our case, documentation should serve the purposes of helping the work of both EPICS users and software developers, and helping new users and developers to get started. For structuring the documentation work towards this purpose, a good source of inspiration is the Diátaxis framework (https://diataxis.fr/).

At the moment, EPICS documentation is scattered over many places, resembling the situation where we were with the software before modern distributed version control systems, in particular git, became available. Documentation is written in different formats and styles, available from different places which one can find using a search engine of choice – if one knows what to search.
The upcoming Documentathon, as well as later similar events are intended to (gradually) improve the state of documentation; this time we want to focus on the process of writing and publishing. An important part of these efforts is to enable and encourage participation in documenting our tools and bringing the documents to the audience - and do it in a way that is well understood and the process is fun to work with.
Improvement is a long-term process; in a single event we can only go so far.

Present situation:

The EPICS website epics-controls.org was created a few years ago. In the first stage, a number of documentation items were ported from the APS website (epics.anl.gov) to the new site. However, we soon noticed that the main website, built with WordPress, is not the most flexible platform for writing and managing the typical documentation items. So, the idea of having an attached site for documentation, hosted at readthedocs.org, was born.
At the moment, the site at readthedocs (RTD), accessible via the URL docs.epics-controls.org, is organized as follows:

The source text is contained in two github projects:

- epics-docs (@github epics-docs)
- epics-howto (@github epics-docs)
- plus beginnings of Application Developer's Guide of EPICS 7, appdev (same place.) Not completed – at least yet.

The documentation from these two is processed by the readthedocs build system and published on the docs website.

Some documentation that was converted from the APS site is still hosted on the epics-controls.org site (e.g., under “support”). The articles are mostly how-to kind of short texts which should be reviewed and decided if they are still of interest, do they need updates or have to be rewritten.

In addition, there is a lot of other documentation scattered around the web. The other documents are about components or applications that belong to the general EPICS infrastructure but are not included in the EPICS core distribution.

Goal (for the long term):

In short, the goal would be to have most (if not all) EPICS-related documentation accessible from a single site. This does not mean that all documentation has to be hosted on the same site. On the contrary, I believe that documentation of a software component, application or library should be kept together with the source code. As most of the EPICS-related software is now hosted or mirrored on publicly available (git) repositories, often together with associated documentation, it is possible to provide links to these from the central docs.epics-controls.org site. Sphinx, the documentation tool (one of them) supported by readthedocs project and build toolchain, provides helper tools to make inter-linking easier and more robust against changes in the remote sites. It also allows at least a prospect of a common look and feel, and consistent navigation.

What should be hosted on the central epics-docs repository:

- User documentation
    - General introductory material (already there, but can be improved.)
    - Specifications; these usually benefit from explicit versioning.
    - Conceptual descriptions that rarely change
    - How-to’s that are separate from specific applications (e.g., “getting started”)

What should probably be hosted elsewhere but integrated to the docs site with links:

- User documentation that changes with the software
    - Record Reference Manual; auto-generated from the sources in core
    - Device support,
    - EPICS “extensions” documentation
        - E.g., Phoebus. These are often large and independent documentation projects on their own. But having links to them will benefit users, especially new ones.
- Core developer documentation
    - AppDevGuide. Parts that explain concepts can, but do not need to be in the epics-docs repository.
    - Related to AppDevGuide, API doc autogenerated from sources in EPICS core with doxygen, versioned per release.
    - Client application developer’s documentation
    - C++, Java, Python, etc., bindings

Related to this:

- Deprecate and hide or remove obsolete pages on epics-controls.org
    - They tend to get unmaintained.
    - Ideally, make docs.epics-controls and epics-controls look-and-feel similar.
- Make it simpler and easier to contribute documentation.
    - At the moment, it is rather unclear what to put where, and in what format.

What has been tested so far:

- Versioned API doc from sources works; a few issues to clarify remain (related to “breathe” and “exhale” (Sphinx extensions); some CPP signatures are not recognized and fail to render.)
    - Plain html works though…
- Sphinx inter-links can be used to restructure the existing docs, for much better integration
    - Interlinks provide an “auto-discovery” of tags, making referring a lot easier
    - “hoverxref” is a nice addition. Also “sphinx-tippy”.
    - Asyn, etc., documentation can now be directly and easily referred to.
- To be considered: move from rst to commonmark. This is now even recommended by RTD, and it works together with Sphinx
    - I noticed some (quite minor but still annoying) formatting issues when mixing rtd and md.
- I managed to solve the build issues on RTD, and am quite happy with the service. However, I have no objections if we want to move to GH.
    - I am not sure how the versioned build works over there, though.
    - I would also need help from somebody to set things up on GH.

Tools to be used:

Writing documentation that is not auto-generated:

- CommonMark (markdown), with MyST enhancements where appropriate.
- reStructuredText (aka rst). Used to be the primary format for readthedocs, but now rst and md can be mixed, and the trend is towards CommonMark, because of wider applicability.
    - However, there are still useful features in rst that are not (yet?) supported in md.
- Readthedocs.org (aka RTD). RTD provides a lot of useful tools. There is a proposal to move to Github Actions; I personally do not mind, but I am not yet convinced if GH provides all the possibilities and tools that RTD does. In any case, I would need somebody else to do the move.
- Git, especially but not only, GitHub

Work plan:

- Spend some time (half a day?) to get familiar with the RTD process.
    - Get an understanding of:

        - Setting up a documentation project with webhooks. Good reading at: Read the Docs: documentation simplified
        - Configuration of the project: conf.py, etc.
        - MyST parser (https://myst-parser.readthedocs.io/en/latest/intro.html) and its relation to CommonMark (https://commonmark.org/)
        - We try to organize some training on the tools (RTD, Sphinx, readthedocs, etc.)
- Discuss and decide the layout for docs.epics-controls.org (on RTD)
- Decide on projects to work on:
    - Add documentation build targets and commands to EPICS base (on Github?)
        - Start and test with a fork.
            - Goal 1: build Record Reference Manual
            - Goal 2: use Doxygen & co. to create API documentation on RTD.
        - This means using Sphinx-multiproject: https://sphinx-multiproject.readthedocs.io/en/latest/
    - Add configuration and (perl) code to EPICS-base that processes dbd.pod and other .pod files to Markdown. Perl scripts exist but need review (they were created by a Perl n00b, namely me, by copying and modifying Andrew's code.)
    - Configure the doc site menus so that they make sense for a reader.
    - Review document items on epics-controls.org
        - Convert to Markdown (or RST) if the content is still useful. Add to how-to's, docs or some other (to be decided) appropriate place.
        - Deprecate (=hide) obsolete documents
        - Restructure the documentation page so that it appropriately points to the docs.epics-controls page, or to items within.
    - Assemble Record Reference Manual on RTD.
        - Some inspiration on how it could look can be found here.
    - Assemble Application Developer's Guide on RTD.
        - Some parts have already been converted to reStructuredText; revision and integration needed.
    - Assemble the API documentation on RTD.
        - A proof-of-concept exists. Discussion on how to best proceed with this is necessary. In principle this can be done by adding some build targets and some configuration files in the EPICS core git repo.
    - Link other related documentation to the RTD site. Use Sphinx interlinks and the site's object inventory when possible.

        - Investigate the possibility of using the same theme (if it makes sense.)
        - Some examples:
            - pva2pva, asyn, sequencer
            - CS-Studio (Phoebus)
            - Archiver Appliance (non-Sphinx)
            - ChannelFinder (https://channelfinder.readthedocs.io/en/latest/)
    - Create a (hopefully short and easy) **EPICS Document Writer's Guide**
    - Add some life to the pages with hoverxref or sphinx-tippy.
- When the infrastructure is in place, start to think how to update the documents with EPICS 7-specifics so that the reader gets a consistent picture.
    - Write, or modify the documentation accordingly.