Documentation Home

- Software Developer Documentation
- Tools for Creating Documentation in RST
  - Editing Tools
    - Ubuntu
    - All Platforms
    - Web-based Editors
  - Screen Capture and Edit
    - Ubuntu
    - Mac
    - Windows
- Start your repo documentation
  - Documentation directory Structure
    - Required Files
    - Optional Files
    - Templates
  - Configure Doc Builds
  - Writing Guidelines
- Having Questions

Getting Started

Please refer here for the step by step tutorials to get you started with installing and running the recent release features.

Software Developer Documentation

O-RAN-SC project documentation comes in two types. wiki (which you are in now) and the documentation that is updated by the developers in https://docs.o-ran-sc.org/en/latest/

Documentation Contributor Guide

This guide describes how to create documentation for the O-RAN Software Community (O-RAN-SC) component. platform. O-RAN -SC repositories create a variety of content depending on the nature of the project. For example, projects delivering a component may have different types of content from a project that creates libraries for a software development kit. The content from each project may be used together as a reference for that project and/or be used in documents are tailored to a specific user audience and task they are performing.

Much of the content in this document is derived from similar documentation processes used in other Linux Foundation Projects including ONAP, OPNFV and Open Daylight.

O-RAN-SC documentation is stored in git repositories, changes are managed with gerrit reviews, and published documents generated when there is a change in any source used to build the documentation.

Authors create source for documents in reStructured Text (RST) that is and published on https://docs.o-ran-sc.org/. The O-RAN-OSC wiki and other web sites can reference these rendered documents directly, allowing projects to easily maintain current release documentation.

Tools for Creating Documentation in RST

Editing Tools

Ubuntu

1. ReText: Ubuntu Software Store or follow the instructions on the project's Github page; ReText has a live preview feature
2. Atom with the RST-Preview Package
3. Notepadqq
4. gedit

All Platforms

1. ReST Editor for Eclipse
2. Visual Studio Code with the vscode-restructuredtext extension

VIM

> *reStructuredText syntax highlighting mode*
>
> *VST (Vim reStructuredText) plugin for Vim7 with folding.*
>
> *VOoM plugin for Vim that emulates two-pane outliner with support for reStructuredText (since version 4.0b2).*
>
> *Riv: Take notes in rst Vim plugin to take notes in reStructured text.*

1. Emacs with rst-mode turned on
2. PyCharm
3. Atom with RST plugins

Web-based Editors

1. NoTex - it can handle complete projects. You can upload your files and edit stuff.
2. Online reStructuredText Editor - It does not support all reST constructs (like the `.. codeblock::` directive),

Project Documentation Guide

Documentation is an important aspect to any software project. LF-Releng provides some recommended tools for projects to get setup with their own documentation and we will attempt to describe them in this guide.

Screen Capture and Edit

Ubuntu

Shutter: Ubuntu Software Store or from PPA instructions on the project's websiteMac

Mac

Greenshot: Mac App Store ($1.99)

Windows

Greenshot (free)

Component Docs Guide

In the past, standard documentation methods included ad-hoc Word documents, PDFs, poorly organized wikis, and other, often closed, tools like Adobe FrameMaker. The rise of DevOps, Agile, and Continuous Integration, however, created a paradigm shift for those who care about documentation because:

1. Documentation must be tightly coupled with code/product releases. In many cases, particularly with open-source products, many different versions of the same code can be installed in various production environments. DevOps personnel must have access to the correct version of documentation.

2. Resources are often tight, volunteers scarce. With a large software base like OSC, a small team of technical writers, even if they are also developers, cannot keep up with a constantly changing, large code base. Therefore, those closest to the code should document it as best they can, and let professional writers edit for style, grammar, and consistency.

Plain-text formatting syntaxes, such as reStructuredText (RST), are a good choice for documentation because:

1. They are editor agnostic
2. The source is nearly as easy to read as the rendered text
3. Documentation can be treated exactly as source code is treated
4. Shallow learning curve

The Documentation team chose reStructuredText largely because of Sphinx, a Python-based documentation build system, which uses reStructuredText natively. In a code base as large as OSC', cross-referencing between component documentation was deemed critical. Sphinx and reStructuredText have built-in functionality that makes collating and cross-referencing component documentation easier.

The Sphinx version is defined in `` `documentation/etc/requirements.txt` ``.

RST vs the Wiki - Which Docs Go Where

Frequently, developers ask where documentation should be created. Should they always use reStructuredText/Sphinx? Not necessarily. Is the wiki appropriate for anything at all? Yes.

It's really up to the development team. Here is a simple rule:

The more tightly coupled the documentation is to a particular version of the code, the more likely it is that it should be stored with the code in reStructuredText.

Two examples on opposite ends of the spectrum:

Example 1: API documentation is often stored literally as code in the form of formatted comment sections. This would be an ideal choice for reStructuredText stored in a docs repo.

Example 2: Meeting notes, release plans – the wiki would be a better choice for this.

The Docs team encourages component teams to store as much documentation as reStructuredText as possible because:

- The Docs team can more easily edit component documentation for grammar, spelling, clarity, and consistency
- A consistent formatting syntax across components will allow the doc team more flexibility in producing different kinds of output
- The documentation can easily be reorganized
- Wiki articles tend to grow stale over time as the people who write them change positions or projects

RST Tools and Resources

For detailed information ReStructuredText and how to best use the format, see:

1. RST Primer
2. RST Quick Reference
3. RST Directives
4. RST Cheatsheet

A list of RST tools is available on the Tools for Creating Documentation in RST page. Sphinx is also a good tool on Linux system.

Start your repo documentation

Documentation directory Structure

In O-RAN SC, before you starting document your repo documentation, it is suggested to prepare a set of rst files (both required files and optional files) under your-repo/docs (e.g. doc/docs/).

Required Files

Documentation team suggests to contain following files in your-repo/docs to help build up O-RAN SC documentation.

1. `index.rst` the entrance of a repo documentation, includes what documentation files your repo has. You will need to list what optional files rst you have in your index.rst.
2. `overview.rst` state the overview of the your project

Optional Files

Optional files in the following are depends on what a specific repo needs. Your may need all of them or part of them.

1. `installation-guides.rst` contains how to install O-RAN SC component
2. `release-notes.rst` contains the release notes for the component
3. `developer-guide.rst` contains information that a developer needs to know in order to work on the component
    a. this should be very technical, aimed at people who want to help develop the components
    b. this should be how the component does what it does, not a requirements document of what the component should do
    c. this should contain what language(s) and frameworks are used, with versions
    d. this should contain how to obtain the code, where to look at work items (Jira tickets), how to get started developing
4. `api-docs.rst` contains details on the component's API. There are some good tools to help generate API document.
    a. Doxygen: it is the defacto standard tool for generating documentation from annotated C++ sources, but it also supports other popular programming languages such as C, Objective-C, C#, PHP, Java, Python, IDL (Corba, Microsoft, and UNO/OpenOffice flavors), Fortran, VHDL, Tcl, and to some extent D. http://www.doxygen.nl/
    b. Swagger: Simplify API development for users, teams, and enterprises with the Swagger open source and professional toolset. Find out how Swagger can help you design and document your APIs at scale. Good for converting yaml and restful API format files to doucment. https://swagger.io/

1. `user-guide.rst` contains information on how to use and configure the component; most components will not have a user guide
    a. if the guide contains sections on third-party tools, is it clearly stated why the OSC platform is using those tools? are there instructions on how to install and configure each tool/toolset?
    b. does the guide state who the target users are? for example, modeler/data scientist, OSC platform admin, marketplace user, design studio end user, etc
    c. if there are instructions, they are clear, correct, and fit for purpose
    d. does the guide contain information more suited for a different guide?
    e. a user guide should be how to use the component or system; it should not be a requirements document
    f. a user guide should contain configuration, administration, management, using, and troubleshooting sections for the feature

Note: It is not mandatory to have all files in your repository, please select the files you need.

Templates

Templates are available in the documentation project under doc/doc-templates. You can clone the documentation project or download from here. People who wish to contribute OSC docs please free free to download templates. Feedback are welcome.

Current list of templates:

index.rst

overview.rst

installation-guide.rst

release-notes.rst

developer-guide.rst

api-docs.rst

user-guide.rst

The templates themselves also contain guidance on what topics to include in the contents. **Please read the contents of the templates!**

Configure Doc Builds

This section provides instructions for configuring your component's documentation area so the O-RAN-SC Jenkins will automatically build and deploy the material to https://o-ran-sc.readthedocs.io/en/latest/ upon change of any file within your docs/ directory. There are many options you can choose to configure your repo documentation in O-RAN SC.

A detailed guide for the setup process is here: Configure Repo for Documentation

Another option is git clone doc repo, in doc/doc-templates/ directory, your can have what your need. Please start with README.txt

The automated setup process that is documented at https://docs.releng.linuxfoundation.org/projects/global-jjb/en/latest/jjb/lf-rtdv3-jobs.html

If you have questions about this process or need help, please contact the O-RAN-SC Documentation Project Technical Lead, weichen ni

Writing Guidelines

Documentation writing guidelines can be found at Documentation Writing Guidelines

Having Questions

If you have any questions on documentation, please send e-mail to Weichen Ni <niweichen@chinamobile.com>

Software Developer Documentation

O-RAN-SC project documentation comes in two types. wiki (which you are in now) and the documentation that is updated by the developers in https://docs.o-ran-sc.org/en/latest/

Documentation Contributor Guide

This guide describes how to create documentation for the O-RAN Software Community (O-RAN-SC) component. platform. O-RAN -SC repositories create a variety of content depending on the nature of the project. For example, projects delivering a component may have different types of content from a project that creates libraries for a software development kit. The content from each project may be used together as a reference for that project and/or be used in documents are tailored to a specific user audience and task they are performing.

Much of the content in this document is derived from similar documentation processes used in other Linux Foundation Projects including ONAP, OPNFV and Open Daylight.

O-RAN-SC documentation is stored in git repositories, changes are managed with gerrit reviews, and published documents generated when there is a change in any source used to build the documentation.

Authors create source for documents in reStructured Text (RST) that is and published on https://docs.o-ran-sc.org/. The O-RAN-OSC wiki and other web sites can reference these rendered documents directly, allowing projects to easily maintain current release documentation.

Tools for Creating Documentation in RST

Editing Tools

Ubuntu

1. ReText: Ubuntu Software Store or follow the instructions on the project's Github page; ReText has a live preview feature
2. Atom with the RST-Preview Package
3. Notepadqq
4. gedit

All Platforms

1. ReST Editor for Eclipse
2. Visual Studio Code with the vscode-restructuredtext extension

VIM

> *reStructuredText syntax highlighting mode*
>
> *VST (Vim reStructuredText) plugin for Vim7 with folding.*
>
> *VOoM plugin for Vim that emulates two-pane outliner with support for reStructuredText (since version 4.0b2).*
>
> *Riv: Take notes in rst Vim plugin to take notes in reStructured text.*

1. Emacs with rst-mode turned on
2. PyCharm
3. Atom with RST plugins

Web-based Editors

1. NoTex - it can handle complete projects. You can upload your files and edit stuff.
2. Online reStructuredText Editor - It does not support all reST constructs (like the `.. codeblock ::` directive),

Project Documentation Guide

Documentation is an important aspect to any software project. LF-Releng provides some recommended tools for projects to get setup with their own documentation and we will attempt to describe them in this guide.

Screen Capture and Edit

Ubuntu

Shutter: Ubuntu Software Store or from PPA instructions on the project's websiteMac

Mac

Greenshot: Mac App Store ($1.99)

Windows

Greenshot (free)

Component Docs Guide

In the past, standard documentation methods included ad-hoc Word documents, PDFs, poorly organized wikis, and other, often closed, tools like Adobe FrameMaker. The rise of DevOps, Agile, and Continuous Integration, however, created a paradigm shift for those who care about documentation because:

1. Documentation must be tightly coupled with code/product releases. In many cases, particularly with open-source products, many different versions of the same code can be installed in various production environments. DevOps personnel must have access to the correct version of documentation.

2. Resources are often tight, volunteers scarce. With a large software base like OSC, a small team of technical writers, even if they are also developers, cannot keep up with a constantly changing, large code base. Therefore, those closest to the code should document it as best they can, and let professional writers edit for style, grammar, and consistency.

Plain-text formatting syntaxes, such as reStructuredText (RST), are a good choice for documentation because:

1. They are editor agnostic
2. The source is nearly as easy to read as the rendered text
3. Documentation can be treated exactly as source code is treated
4. Shallow learning curve

The Documentation team chose reStructuredText largely because of Sphinx, a Python-based documentation build system, which uses reStructuredText natively. In a code base as large as OSC', cross-referencing between component documentation was deemed critical. Sphinx and reStructuredText have built-in functionality that makes collating and cross-referencing component documentation easier.

The Sphinx version is defined in `documentation/etc/requirements.txt`.

RST vs the Wiki - Which Docs Go Where

Frequently, developers ask where documentation should be created. Should they always use reStructuredText/Sphinx? Not necessarily. Is the wiki appropriate for anything at all? Yes.

It's really up to the development team. Here is a simple rule:

The more tightly coupled the documentation is to a particular version of the code, the more likely it is that it should be stored with the code in reStructuredText.

Two examples on opposite ends of the spectrum:

Example 1: API documentation is often stored literally as code in the form of formatted comment sections. This would be an ideal choice for reStructuredText stored in a docs repo.

Example 2: Meeting notes, release plans – the wiki would be a better choice for this.

The Docs team encourages component teams to store as much documentation as reStructuredText as possible because:

- The Docs team can more easily edit component documentation for grammar, spelling, clarity, and consistency
- A consistent formatting syntax across components will allow the doc team more flexibility in producing different kinds of output
- The documentation can easily be reorganized
- Wiki articles tend to grow stale over time as the people who write them change positions or projects

RST Tools and Resources

For detailed information ReStructuredText and how to best use the format, see:

1. RST Primer
2. RST Quick Reference
3. RST Directives
4. RST Cheatsheet

A list of RST tools is available on the Tools for Creating Documentation in RST page. Sphinx is also a good tool on Linux system.

Start your repo documentation

Documentation directory Structure

In O-RAN SC, before you starting document your repo documentation, it is suggested to prepare a set of rst files (both required files and optional files) under your-repo/docs (e.g. doc/docs/).

Required Files

Documentation team suggests to contain following files in your-repo/docs to help build up O-RAN SC documentation.

1. `index.rst` the entrance of a repo documentation, includes what documentation files your repo has. You will need to list what optional files rst you have in your index.rst.

2. `overview.rst` state the overview of the your project

Optional Files

Optional files in the following are depends on what a specific repo needs. Your may need all of them or part of them.

1. `installation-guides.rst` contains how to install O-RAN SC component
2. `release-notes.rst` contains the release notes for the component
3. `developer-guide.rst` contains information that a developer needs to know in order to work on the component

    a. this should be very technical, aimed at people who want to help develop the components
    b. this should be how the component does what it does, not a requirements document of what the component should do
    c. this should contain what language(s) and frameworks are used, with versions
    d. this should contain how to obtain the code, where to look at work items (Jira tickets), how to get started developing
4. `api-docs.rst` contains details on the component's API. There are some good tools to help generate API document.

    a. Doxygen: it is the defacto standard tool for generating documentation from annotated C++ sources, but it also supports other popular programming languages such as C, Objective-C, C#, PHP, Java, Python, IDL (Corba, Microsoft, and UNO/OpenOffice flavors), Fortran, VHDL, Tcl, and to some extent D. http://www.doxygen.nl/
    b. Swagger: Simplify API development for users, teams, and enterprises with the Swagger open source and professional toolset. Find out how Swagger can help you design and document your APIs at scale. Good for converting yaml and restful API format files to doucment. https://swagger.io/

1. `user-guide.rst` contains information on how to use and configure the component; most components will not have a user guide
    a. if the guide contains sections on third-party tools, is it clearly stated why the OSC platform is using those tools? are there instructions on how to install and configure each tool/toolset?
    b. does the guide state who the target users are? for example, modeler/data scientist, OSC platform admin, marketplace user, design studio end user, etc
    c. if there are instructions, they are clear, correct, and fit for purpose
    d. does the guide contain information more suited for a different guide?
    e. a user guide should be how to use the component or system; it should not be a requirements document
    f. a user guide should contain configuration, administration, management, using, and troubleshooting sections for the feature

Note: It is not mandatory to have all files in your repository, please select the files you need.

Templates

Templates are available in the documentation project under doc/doc-templates. You can clone the documentation project or download from here. People who wish to contribute OSC docs please free free to download templates. Feedback are welcome.

Current list of templates:

index.rst

overview.rst

installation-guide.rst

release-notes.rst

developer-guide.rst

api-docs.rst

user-guide.rst

The templates themselves also contain guidance on what topics to include in the contents. **Please read the contents of the templates!**

Configure Doc Builds

This section provides instructions for configuring your component's documentation area so the O-RAN-SC Jenkins will automatically build and deploy the material to https://o-ran-sc.readthedocs.io/en/latest/ upon change of any file within your docs/ directory. There are many options you can choose to configure your repo documentation in O-RAN SC.

A detailed guide for the setup process is here: Configure Repo for Documentation

Another option is git clone doc repo, in doc/doc-templates/ directory, your can have what your need. Please start with README.txt

The automated setup process that is documented at https://docs.releng.linuxfoundation.org/projects/global-jjb/en/latest/jjb/lf-rtdv3-jobs.html

If you have questions about this process or need help, please contact the O-RAN-SC Documentation Project Technical Lead, weichen ni

Writing Guidelines

Documentation writing guidelines can be found at Documentation Writing Guidelines

Having Questions

If you have any questions on documentation, please send e-mail to Weichen Ni <niweichen@chinamobile.com>