CCI xG Testbed Documentation Project

Comprehensive Guide for Project Handover

May 2025

1. Introduction

This document provides comprehensive documentation for the CCI xG Testbed documentation project. It is intended to serve as a complete guide for anyone taking over the project, covering everything from project structure to deployment processes.

1.1 Project Overview

The Commonwealth Cyber Initiative (CCI) xG Testbed is a cutting-edge research facility designed to support experimentation and prototyping of next-generation technologies in networks and artificial intelligence (AI) solutions. This documentation project aims to provide comprehensive information about the testbed, its capabilities, and how to use it.

The documentation is built using Sphinx, a powerful documentation generator that converts reStructuredText (.rst) files into HTML, PDF, and other formats. The project is hosted on GitLab and deployed to ReadTheDocs for public access.

1.2 Key Features

- Comprehensive documentation of the CCI xG Testbed
- Built with Sphinx and reStructuredText
- Hosted on GitLab for version control
- Deployed to ReadTheDocs for public access
- Covers various aspects of the testbed including 4G, 5G, ORAN, and other related technologies

2. Project Structure

The project follows a standard Sphinx documentation structure with some customizations. Understanding this structure is essential for maintaining and updating the documentation.

2.1 Directory Structure

The project has the following directory structure:

```
CCI_TESTBED_DOCS_PUBLIC/
├── .gitignore                  # Git ignore file
├── .readthedocs.yaml           # ReadTheDocs configuration
├── Instructions.txt            # Basic instructions for
building the docs
├── README.md                   # Project README
├── requirements.txt            # Python dependencies
└── source/                     # Documentation source
files
    ├── _static/                # Static files (images,
CSS, etc.)
    ├── _templates/             # Custom templates
    ├── getting_started/        # Getting started guides
    ├── images-4g/              # 4G-related images
```

```
    ├── images-5g/              # 5G-related images
    ├── images-oran/            # ORAN-related images
    ├── opensas/                # OpenSAS documentation
    ├── overview/               # Project overview
    ├── sample_experiments/     # Sample experiments
    ├── sites/                  # Testbed sites information
    ├── software_architecture/  # Software architecture
documentation
    ├── user-dashboard/         # User dashboard
documentation
    ├── conf.py                 # Sphinx configuration
    ├── index.rst               # Main index file
    └── various .rst files      # Content files
```

2.2 Key Files

The following files are particularly important for the project:

2.2.1 Configuration Files

source/conf.py - Sphinx Configuration:

```
# Configuration file for the Sphinx documentation builder.
import sphinx_rtd_theme
project = 'CCI_TESTBED_DOCS'
copyright = '2024, TEAM'
author = 'TEAM'
release = '1.0.0'

graphviz_dot = r'C:\Program Files\Graphviz\bin\dot.exe'

extensions = [
    'sphinx.ext.graphviz'
]

templates_path = ['_templates']
exclude_patterns = []

html_theme_options = {
'logo_only': True,
'style_nav_header_background': 'black',
'display_version': False,
'collapse_navigation': True,
'navigation_depth': 2
}

html_theme = 'sphinx_rtd_theme'
html_logo = '_static/xG-rev-4c-01.png'
html_static_path = ['_static']
```

```
html_css_files = ['width.css']
```

.readthedocs.yaml - ReadTheDocs Configuration:

```
# .readthedocs.yaml
# Read the Docs configuration file
version: 2

build:
  os: ubuntu-22.04
  tools:
    python: "3.12"
  apt_packages:
    - graphviz

sphinx:
  configuration: source/conf.py

python:
  install:
    - requirements: requirements.txt
```

requirements.txt - Python Dependencies:

```
sphinx
sphinx_rtd_theme
sphinx-autobuild
```

2.2.2 Content Structure

source/index.rst - Main Index File:

This file defines the structure of the documentation and includes all the sections that will appear in the table of contents. It uses the toctree directive to organize content into sections.

```
CCI's xG Testbed documentation
==============================

.. toctree::
   :maxdepth: 1
   :caption: Overview

   Introduction <overview/ccixg_introduction>

.. toctree::
   :maxdepth: 1
```

```
   :caption: Testbed Sites

   sites/indoor
   sites/outdoor

.. toctree::
   :maxdepth: 1
   :caption: Getting Started

   Getting Started <getting_started/user_access>
   OpenStack Guide <getting_started/openstack>
   Gateway and Redmine Guide
<getting_started/gateway_and_redmine>

... (additional sections)
```

3. Setup and Installation

This section covers how to set up the development environment for working on the documentation project.

3.1 Prerequisites

Before you can work on the documentation, you need to have the following installed:

- Python 3.12 or later
- Git
- Graphviz (for diagrams)

3.2 Clone the Repository

To get started, clone the repository from GitLab:

```
git clone https://gitlab.com/ccixgtestbed/cci_docs.git
cd cci_docs
```

3.3 Install Dependencies

Install the required Python packages:

```
pip install -r requirements.txt
```

This will install Sphinx, the ReadTheDocs theme, and sphinx-autobuild.

4. Building the Documentation

This section explains how to build and preview the documentation locally.

4.1 Building with sphinx-autobuild

The easiest way to build and preview the documentation is using sphinx-autobuild, which automatically rebuilds the documentation when files change:

```
sphinx-autobuild source build/html
```

This will start a local web server at http://127.0.0.1:8000 where you can preview the documentation. The server will automatically reload when you make changes to the source files.

4.2 Building with Sphinx directly

You can also build the documentation directly using the sphinx-build command:

```
sphinx-build -b html source build/html
```

This will build the HTML documentation in the build/html directory. You can open the index.html file in a web browser to preview the documentation.

5. Writing Documentation

This section provides guidelines for writing and formatting documentation using reStructuredText.

5.1 reStructuredText Basics

reStructuredText (reST) is a lightweight markup language that is easy to read and write. Here are some basic formatting examples:

```
# Section Heading (H1)
====================

## Section Heading (H2)
--------------------

### Section Heading (H3)
^^^^^^^^^^^^^^^^^^^^

**Bold text**

*Italic text*

`Code`
```

````
[Link text](https://example.com)

- Bullet point
- Another bullet point
  - Nested bullet point

1. Numbered item
2. Another numbered item
   a. Nested numbered item

> Blockquote

```python
# Code block
def hello_world():
    print("Hello, world!")
```

| Column 1 | Column 2 |
|----------|----------|
| Cell 1   | Cell 2   |
| Cell 3   | Cell 4   |
````

5.2 Sphinx Directives

Sphinx extends reStructuredText with additional directives. Here are some commonly used directives:

```
.. toctree::
   :maxdepth: 2
   :caption: Contents:

   file1
   file2
   directory/file3

.. image:: path/to/image.png
   :alt: Alternative text
   :width: 400px

.. note::
   This is a note.

.. warning::
   This is a warning.

.. code-block:: python
```

```
   :linenos:

   def hello_world():
       print("Hello, world!")

.. figure:: path/to/image.png
   :alt: Alternative text
   :width: 400px

   Figure caption

.. raw:: html

   <div style="text-align: center;">
     <video width="640" height="480" controls>
       <source src="../_static/video.mp4" type="video/mp4">
       Your browser does not support the video tag.
     </video>
   </div>
```

5.3 Adding New Content

To add new content to the documentation:

- Create a new .rst file in the appropriate directory
- Add the file to the toctree directive in the parent index.rst file
- Write the content using reStructuredText syntax
- Build the documentation to preview your changes

Example of adding a new section:

```
# Create a new file
touch source/new_section.rst

# Edit the file
nano source/new_section.rst

# Add content to the file
New Section Title
================

This is a new section in the documentation.

## Subsection
-----------

Content for the subsection.
```

```
# Add the file to the toctree in index.rst
.. toctree::
   :maxdepth: 1
   :caption: New Section

   new_section
```

6. GitLab Integration

This section explains how to work with the GitLab repository for version control and collaboration.

6.1 GitLab Workflow

The project uses GitLab for version control and collaboration. Here is the recommended workflow:

- Clone the repository: git clone https://gitlab.com/ccixgtestbed/cci_docs.git
- Create a new branch for your changes: git checkout -b feature/new-section
- Make your changes and commit them: git add . && git commit -m "Add new section"
- Push your changes to GitLab: git push origin feature/new-section
- Create a merge request on GitLab to merge your changes into the main branch

6.2 GitLab CI/CD

GitLab CI/CD can be used to automatically build and test the documentation when changes are pushed to the repository. This helps ensure that the documentation builds correctly before it is deployed to ReadTheDocs.

Example .gitlab-ci.yml file:

```
image: python:3.12

before_script:
  - pip install -r requirements.txt

stages:
  - build

build:
  stage: build
  script:
    - sphinx-build -b html source build/html
  artifacts:
    paths:
      - build/html
```

7. ReadTheDocs Integration

This section explains how the documentation is deployed to ReadTheDocs for public access.

7.1 ReadTheDocs Configuration

The project is configured to build and deploy automatically on ReadTheDocs using the .readthedocs.yaml file. This file specifies the build environment, Python version, and dependencies required to build the documentation.

The current configuration uses:

- Ubuntu 22.04 as the build OS
- Python 3.12
- Graphviz for diagrams
- Dependencies from requirements.txt

7.2 Deployment Process

The deployment process is as follows:

- Changes are pushed to the GitLab repository
- ReadTheDocs detects the changes and triggers a build
- The documentation is built using the configuration in .readthedocs.yaml
- If the build is successful, the documentation is deployed to the ReadTheDocs website
- The documentation is available at https://cci-testbed-docs.readthedocs.io/

7.3 Managing the ReadTheDocs Project

To manage the ReadTheDocs project:

- Log in to ReadTheDocs with the account that has access to the project
- Go to the project dashboard
- From here, you can:
  - View build status and logs
  - Configure project settings
  - Manage versions and translations
  - Set up webhooks for automatic builds

8. Maintenance and Updates

This section provides guidelines for maintaining and updating the documentation over time.

8.1 Regular Maintenance Tasks

To keep the documentation up-to-date and relevant, perform these maintenance tasks regularly:

- Review and update content for accuracy
- Check for broken links and fix them
- Update screenshots and examples to reflect the current state of the testbed
- Add new sections for new features or capabilities
- Remove or archive outdated information

8.2 Updating Dependencies

Periodically update the project dependencies to ensure security and compatibility:

```
# Update pip
pip install --upgrade pip

# Update dependencies
pip install --upgrade -r requirements.txt

# Update requirements.txt with the new versions
pip freeze > requirements.txt
```

8.3 Version Control

Use version control best practices to manage changes to the documentation:

- Create meaningful commit messages that describe the changes
- Use branches for significant changes or new features
- Review changes before merging them into the main branch
- Tag releases with version numbers

9. Troubleshooting

This section provides solutions to common issues that may arise when working with the documentation project.

9.1 Build Errors

Common build errors and their solutions:

9.1.1 Missing Dependencies

If you encounter errors about missing dependencies, ensure that you have installed all required packages:

```
pip install -r requirements.txt
```

9.1.2 Graphviz Errors

If you encounter errors related to Graphviz, ensure that Graphviz is installed and properly configured in conf.py:

```
# Install Graphviz
# On Ubuntu/Debian
sudo apt-get install graphviz

# On macOS
brew install graphviz

# On Windows
# Download and install from https://graphviz.org/download/

# Update conf.py with the correct path to the dot
executable
graphviz_dot = r'C:\Program Files\Graphviz\bin\dot.exe'  #
Windows
# or
graphviz_dot = '/usr/bin/dot'  # Linux/macOS
```

9.1.3 reStructuredText Syntax Errors

If you encounter errors related to reStructuredText syntax, check the error message for the file and line number, then fix the syntax error. Common issues include:

- Incorrect indentation
- Missing blank lines between sections
- Incorrect directive syntax
- Inconsistent heading underlines

9.2 ReadTheDocs Build Failures

If the documentation fails to build on ReadTheDocs:

- Check the build logs on ReadTheDocs for error messages
- Ensure that all dependencies are specified in requirements.txt
- Verify that the .readthedocs.yaml file is correctly configured
- Test the build locally to identify and fix issues

10. Best Practices

This section provides best practices for working with the documentation project.

10.1 Documentation Style Guide

Follow these style guidelines to maintain consistency throughout the documentation:

- Use clear, concise language
- Write in the present tense
- Use active voice instead of passive voice
- Be consistent with terminology
- Use proper capitalization and punctuation
- Include examples and screenshots where appropriate
- Structure content with headings and subheadings
- Use lists and tables to organize information

10.2 File Organization

Follow these guidelines for organizing files in the project:

- Group related content in directories
- Use descriptive file names
- Keep file paths short and meaningful
- Store images and other static files in the _static directory
- Use subdirectories for different types of content

10.3 Collaboration

Follow these guidelines for collaborating on the documentation:

- Communicate changes and updates to team members
- Review each other's work before merging
- Use GitLab issues to track tasks and bugs
- Document decisions and changes in commit messages

11. Conclusion

This document provides a comprehensive guide for the CCI xG Testbed documentation project. By following the guidelines and best practices outlined in this document, you can effectively maintain and update the documentation to ensure it remains accurate, relevant, and useful for users of the CCI xG Testbed.