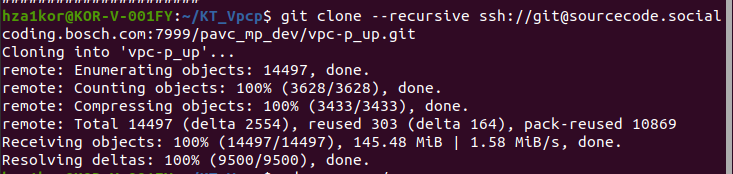
Knowledge Transfer Documentation

# Requirements using Docsascode

## How to use Docsascode in our project

1. Clone the vpcp develop branch

* **git clone --recursive ssh://git@sourcecode.socialcoding.bosch.com:7999/pavc\_mp\_dev/vpc-p\_up.git**



1. Change directory to the project folder

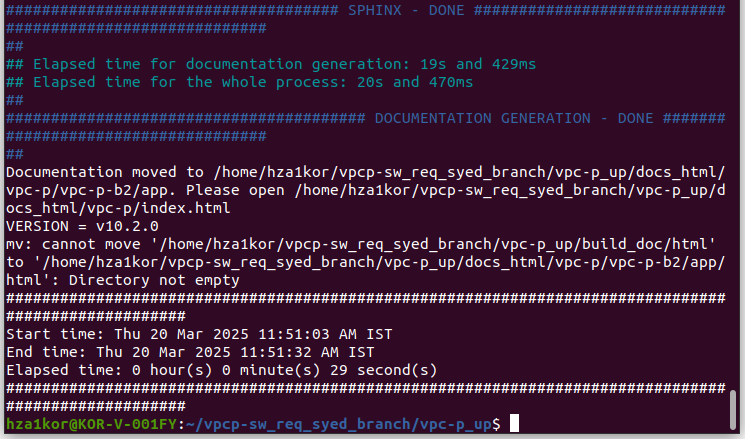
* **cd vpc-p\_up**

1. Execute the build script

* **./meta-bosch/scripts/build.sh -m vpc-p-b2 -i app -d**The -d flag tells the build script to generate documentation as part of the build process. When run with -d, the script converts your source documentation files (reStructuredText files) into formatted HTML output.

1. Locate the generated documentation

* Once the build is done, you can find the documentation inside the 'build\_doc' folder.

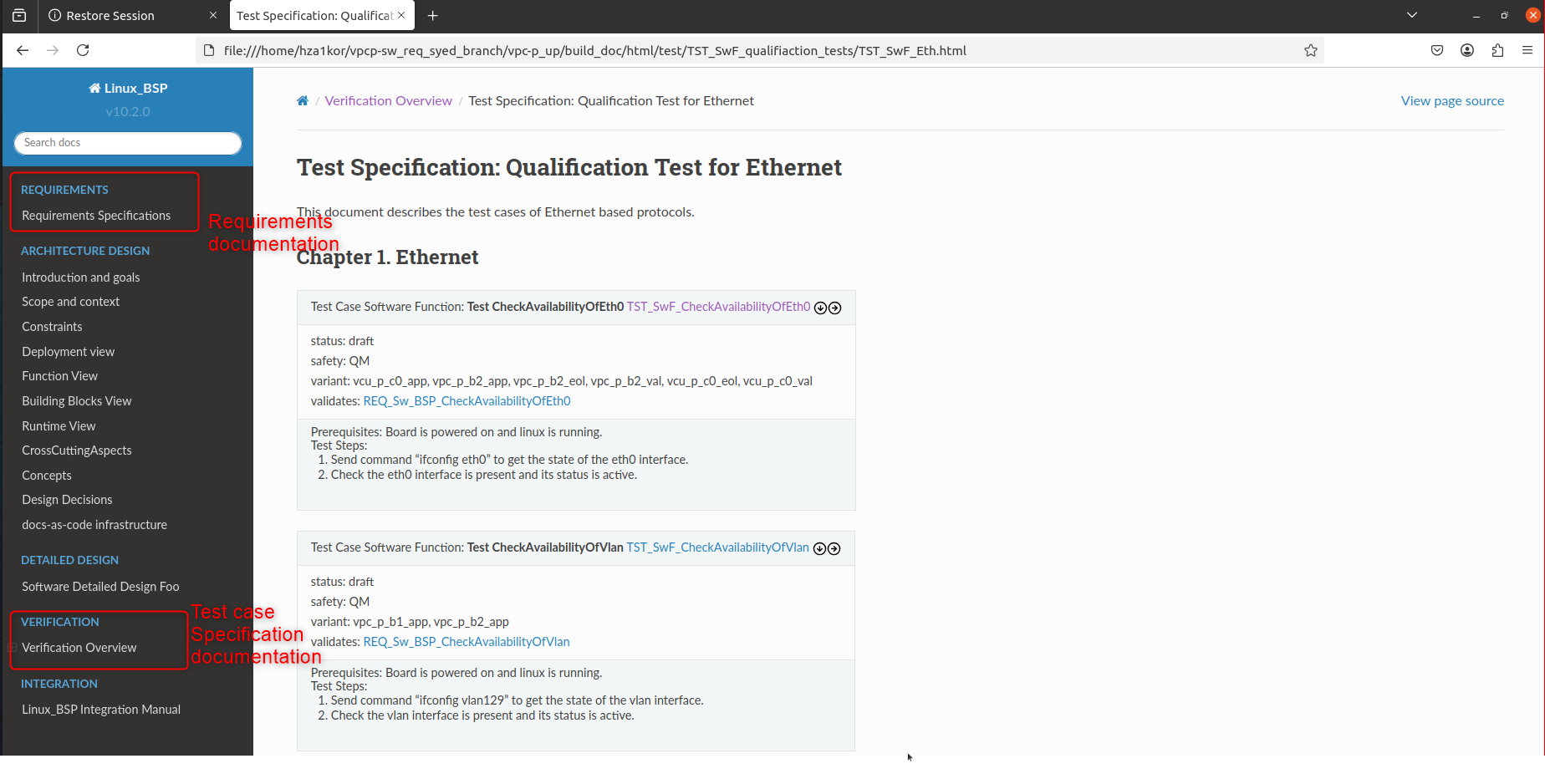


1. Check documentation on requirements

* Navigate to: vpc-p\_up/build\_doc/html/requirements/

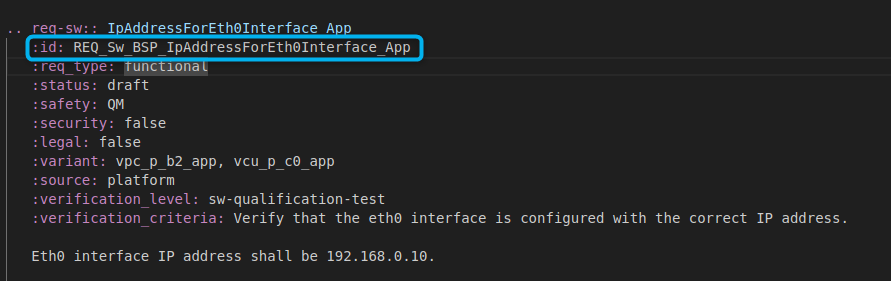
1. Check documentation on test specification

* Navigate to: vpc-p\_up/build\_doc/html/test/

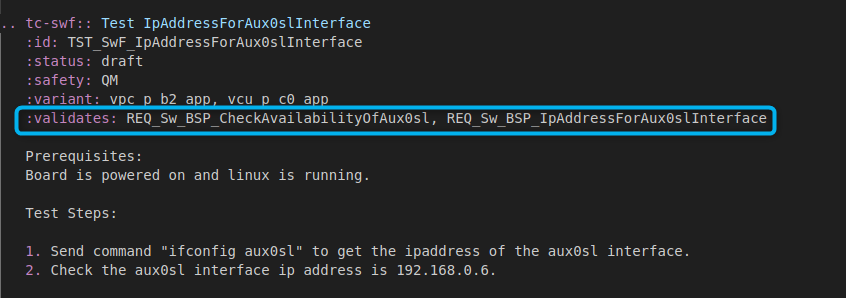


## Requirements and Test Cases Mapping

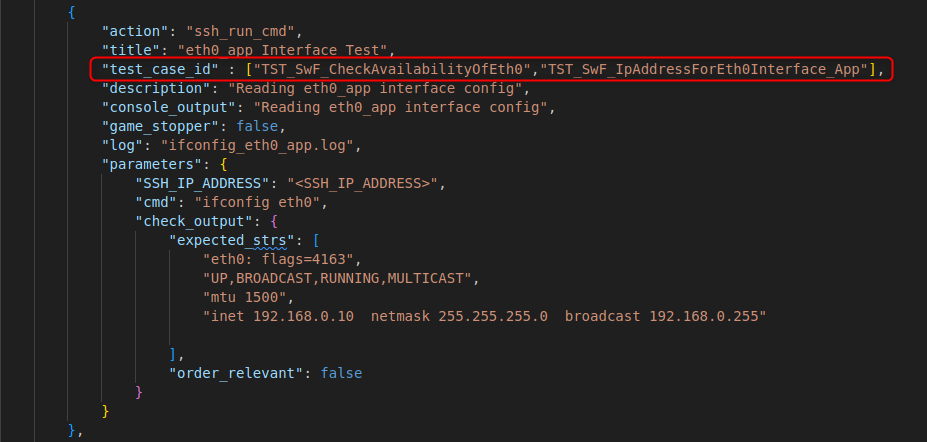
* One requirement can have only one requirement ID.



* Each requirement ID is mapped to test cases; one test case can validate multiple requirement IDs.



* Test cases are provided in JSON format at:
* vpc-p\_up/tests
* For example, test case IDs are added as parameters, and one test case can validate multiple test case specification IDs.



# Additional Information

## Docsascode Explanation

* Docs as code is an approach where documentation is treated like software code. Instead of using traditional word processors, you write docs in plain text with markup languages (such as Markdown or reStructuredText), store them in version control systems like Git, and use automation tools to build and publish them.
* This method enables collaboration, version tracking, and continuous updates, making documentation more consistent and easier to maintain over time.
* Why use docsascode for our project?
* Version Control: Every change is tracked in systems like Git, allowing you to see who changed what and when, and making it easy to revert errors.
* Collaboration: Documentation can be collaboratively edited and reviewed using pull requests, with team members leaving comments and suggestions.
* Automation: Automated pipelines can compile Markdown files into well-formatted HTML or PDF documents and run quality checks for consistency.
* Integration: Linking your requirements and test cases with your codebase and issue trackers creates a strong connection between documentation and development.
* Consistency and Traceability: Using plain text files with standardized markup ensures your documentation maintains a consistent style and structure, and every change is documented.

## Markup Language

* A markup language is a system for annotating text with additional information about its structure and presentation. Instead of writing plain text, you use special symbols or tags to indicate formatting or organization. For example:
* HTML: Uses tags like <h1> for headings and <p> for paragraphs to structure web pages.
* Markdown: Uses simpler symbols like # for headings and \* for lists to format text in a readable way.

## What is Markdown

Markdown is a lightweight markup language that lets you format plain text using simple symbols. For instance, you can create headers by starting a line with one or more '#' symbols, make lists with hyphens or asterisks, and add links or images using straightforward syntax. Its simplicity and readability make it popular for documentation, README files, blog posts, and more. Tools can convert Markdown into HTML for web publishing.

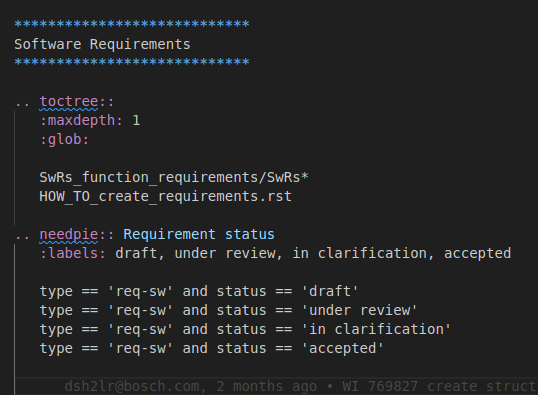
## Importance of index.rst

The Sphinx documentation follows the index.rst format, which serves as the central entry point for your documentation. It defines the structure and table of contents (toctree) for your project. Key benefits include:

* Central Navigation: Acts as the landing page directing Sphinx and readers to various sections.
* Structure Definition: References to other .rst files via toctree directives determine the organization of the documentation.
* Build Process: Sphinx uses index.rst to construct the documentation hierarchy; files not referenced here might be omitted.

## What is toctree

A toctree (table of contents tree) is a Sphinx directive used in reStructuredText files to define the structure and navigation of your documentation.



## What is Sphinx

Sphinx is an open-source documentation generator originally created for Python. It converts plain text files written in reStructuredText into structured, professional documentation in formats such as HTML and PDF.

## How Sphinx and Docsascode are Related

* Sphinx fits perfectly within the 'docs as code' philosophy:
* Docs as Code: An approach that treats documentation like software code by using plain text files stored in Git, with automated build processes.
* Sphinx: A documentation generator that converts reStructuredText files into navigable documentation (HTML/PDF), integrating well with version control and CI systems.

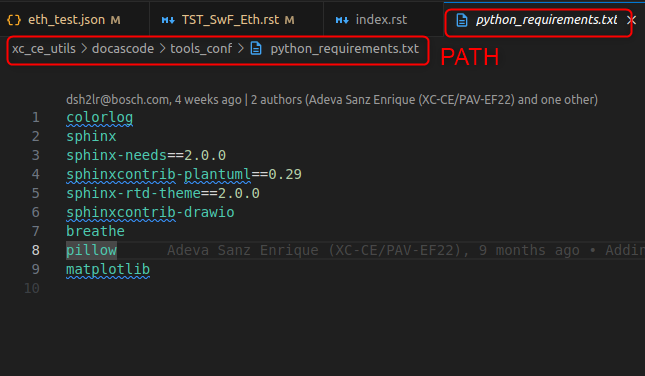
## How to Download and Install Sphinx

* Using pip:
* **pip install sphinx**

## Initialize a New Project

After installing Sphinx, run the following command to set up a new documentation project. This creates a basic directory structure with an index.rst file and a configuration file (conf.py):

* **sphinx-quickstart**
* Sphinx installation in vpc-p - **sphinx is added as a package in python\_requirements.txt in xc\_ce\_utils/docsascode/tools\_conf folder**



## About .rst Files

The '.rst' file extension stands for reStructuredText, a plain text markup language used primarily for technical documentation.