VNFRQTS How to Contribute

Table of Contents

Overview

The goal of the VNF Guidelines and Requirements project is to provide guidance to VNF providers on the guidelines, rules, and specifications to which a VNF must adhere to correctly onboard and operate within ONAP. These rules span across the VNF's lifecycle and are influenced by the various components that comprise ONAP as well as various industry standards that are required for interoperability.

This project welcomes and encourages contributions, feedback, and inquiries through a variety of mechanisms. The intent of this guide is to inform a potential contributor of the various options available to them, and provide instructions on how to use those mechanisms.

Since this project is an approved project of the ONAP community, most of the ways to contribute are consistent with the standard tools used across ONAP. Please refer to the Getting involved with the ONAP community page for general information on getting involved with ONAP.

Available Options

The method for contributing or interacting with the VNF Requirements project depends on the type of contribution. Please refer to the following table for guidance on the best method to use.

Contribution Type	Method(s)
General questions, feedback, or support requests	Leverage the ONAP mailing lists. Send an email to onap-discuss@lists.onap.org and use the project hashtag **#vnfrqts** in the subject line of your email.
Staying apprised of progress or collaborating on new work	Attend the weekly VNF Requirements project meetings. Review prior meeting minutes Additionally you can register for a Linux Foundation ID, and sign into Gerrit and/or JIRA to repository commits and JIRA issues. • Git/Gerrit repositories are prefixed by vnfrqts/* • JIRA Project - VNF Requirements VNFRQTS
Submitting Bugs	If you find an issue in the VNF Requirements project in either the ONAP Developer Guide or the ONAP Wiki, please submit a JIRA ticket under the VNF Requirements project **Project**: VNF Requirements **Issue Type**: Bug Please include the link to the page in question and if applicable the requirement(s) IDs that are impacted.

Contributing Updates to VNF Provider Guidelines and Requirements	We welcome both new requirements and revisions to existing requirements and content. Before contributing the content, please familiarize yourself with the VNFRQTS Requirement and Documentation Standards. Our preferred form of contribution is to submit changes as Gerrit Reviews against the target VNF Requirements repository (see Submitting Gerritt Reviews below for more details) If you are proposing **substantial** changes to the project or the change requires discussion, then we encourage you to create a **Proposal** on the ONAP VNF Requirements Project Wiki to introduce the topic first. Please refer to the VNF Requirements Proposals page for more details. If you do not feel comfortable with the tool chain required to submit changes directly to the reStructuredText via Gerrit reviews, then we would still like to support your contributions and encourage you to put the details of your changes into JIRA. **Project:** VNF Requirements **Issue Type:** Task **Summary:** Short summary of the change In the **Description** field please include the following details: • The full content you are proposing (NOTE - you do not need to provide the requirement IDs, those will be automatically generated. • Where the content should appear (please include the section and link to the page) • The rationale for adding this to the requirements • If adding requirements, please provide the applicable metadata per the **Requirement Metadata** section of the VNFRQTS Requirement and Documentation Standards If you are submitting a ticket, then you are encouraged to attend the VNF Requirements weekly meeting if your schedule permits.

Submitting Gerrit Reviews

The easiest way for the VNF Requirements project to accept and incorporate changes is through merge/review requests via Gerrit. This will enable the team and community to review your proposed changes in context,collaborate on changes, and ensure contributions adhere to the standards of the tool chain required to build the documentation.

Contributing in this manner will require the setup of some tools on your local machine or environment.

Prerequisites

- **Linux Foundation ID:** You will need this to access JIRA and Gerrit. You can register for an ID here.
- **Knowledge of reStructuredText (RST):** All documentation at ONAP is authored in RST. For the most part you can follow the style of other requirements and content without being an RST expert, but understanding basics may be useful. The ONAP Documentation project has some useful guides and tutorials here.
- **Python 3.6+**: You will not need to write any Python code, but it is required to generate the documentation from reStructuredText. If needed, you can find installation instructions for your system here.
- **Git**: This is the version control tool used by ONAP. If not present on your system, then refer to Git's Getting Started - Installation Guide for more information
- **Git Review:** This is the command line extension to Git to enable changes to be submitted to review to Geritt. This needs to be installed **after** you have installed Git and Python.
- **Tox:** This is a tool to run the build. It must be installed after you install Python. Install it using pip.

```
> pip install tox
```

- `VNF Requirements Standards: Familiarize yourself with the VNFRQTS Requirement and Documentation Standards`

Setup

Before submitting your first change, you will need to configure Git and Gerrit to work together. That is outside the scope of this guide, but you can find more information here.

Process

Setting Up the Project

1. Before submitting a change, open a JIRA ticket to introduce the change.
    a. **Project:** VNF Requirements
    b. **Issue Type**: Story for changes that can be incorporated via a single change, Epic if the updates will require multiple commits. If you need an Epic, then you will need a story for each set of changes you wish to make. If the changes you are proposing are substantial, then you may need to create an entry in VNF Requirements Proposals for discussion before proceeding with your requirement changes.

c. **Summary:** Descriptive summary of the changes to be made
d. **Fix Version:** Target ONAP release for your changes

2. Clone the appropriate repository for your content (see VNFRQTS Requirement and Documentation Standards for guidance and links).
   a. Example: git clone ssh://git@gerrit.onap.org:29418/vnfrqts/epics
3. Navigate to the directory you cloned the repository into
4. Run the following command to generate the documentation and verify everything works properly

build docs

```
<project-dir> tox -e docs
```

5. If everything works properly, this will generate the HTML documentation in the **`docs\_build\html`** directory

Editing the Documents

1. The first order of business will be determining if this requirement should be contributed to the latest version of the requirements, or if it is intended for a prior release. If you simply, just need to make a change to the latest content then jump to step 3
2. If you wish to change content in a prior release, then you must checkout that branch. For example, if the **master** branch is now associated with Dublin, but the change you need to make is for Casablanca, then you will need to do the following:
   a. Fetch the remote branches and checkout the appropriate release branch

```
> git fetch
> get checkout casablanca    (Or whatever the appropriate release branch name is)
```

3. If you are adding a new requirement, then you can leave out the :id: and :introduced: attributes as those will be automatically added when you run either ``tox -e docs`` or ``check.py``

Example Requirement

```
.. req::
        :keyword: MUST
        :target: VNF PACKAGE
        :impacts: sdc
        :validation_mode: static

        The VNF Package **MUST** include element XYZ as part of the CSAR package in the ABC directory
```

4. If you are editing the a requirement, then please ensure that the metdata fields are updated appropriately to reflect your changes (example: :updated: should be set to the current release, :keyword: and :target: should still match the requirement, etc.)

Example Requirement

```
.. req::
        :id: R-12345
        :keyword: MUST
        :target: VNF PACKAGE
        :introduced: casablanca
        :impacts: sdc
        :validation_mode: static

        The VNF Package **MUST** include element XYZ as part of the CSAR package in the ABC directory
```

5. After you have made your changes, rebuild the documentation using the **`tox -e docs`** command, and view the changes in browser to ensure they are formatted correctly. This will also perform a variety of quality checks and updates. Please ensure all warnings are addressed before submitting your changes.
   a. **Warnings:**
      i. Requirement missing required attributes
      ii. Invalid values for attributes/metadata
      iii. Invalid section header usage in any file
      iv. :keyword: and requirement mismatch
   b. **Auto Updates:**
      i. Assigning :id: on new requirements where an ID missing
      ii. Adding :introduced: attribute on new requirements
      iii. Adding/correcting :updated: attribute on changed requirements

6. Once you are satisfied with your change, then it will be time to commit and submit your changes for review

Commiting and Submitting Your Changes

1. Initialize your repository to work with Gerrit

```
> git review -s
```

2. Add your changes to git

```
> git add --all
```

3. Commit your changes

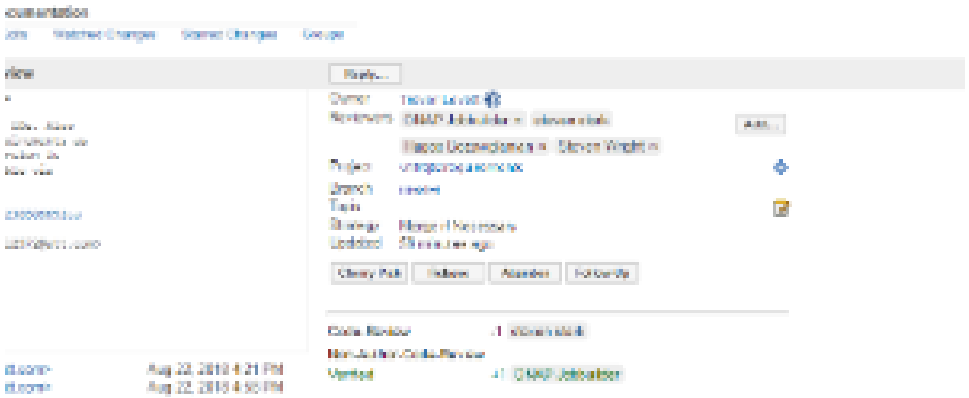
```
> git commit -s
```

4. Your default, configured editor will open to provide a commit message. Please ensure that your commit message adheres to the ONAP Commit Messages standards or your submission will be automatically rejected by Gerrit.
    a. NOTE: At the bottom of your commit message be sure to include the line Issue-ID: VNFRQTS-#### that corresponds to the JIRA ticket you opened for this change.
5. Now you are ready to push your changes to Gerrit using **git review**

```
> git review {branch name}  where {branch name} is the name of your target branch (ex: casablanca). If you are submitting to master, then you you can leave off branch name
```

6. Once you have successfully pushed the change up, please log into Gerrit and add reviewers

    a. Log into your dashboard
    b. See your Outgoing Reviews and click on the link corresponding to your review
    c. Click Add to add reviewers



    d.
    e. Add the following reviewers to your review
        i. Trevor Lovett
        ii. Junfeng Wang
        iii. Alan Weinstock
        iv. Any additional reviewers you feel would add value
7. Please pay attention to your inbox following the creation of your review. The ONAP Jobbuilder will attempt to build your documentation and may reject the change if there are problems. Please review it's messages and resolve any issues that it raises.

Making Updates

1. Often there will be changes you will need to make to get this working correctly.
2. Make any changes locally based on the feedback and add those changes to be committed.

```
> git add --all
```

3. Rather than a normal commit, we'll just amend your previous commit.

```
> git commit --amend
```

4. Finally resubmit your change via git review and your change will be updated

```
> git review {branch name}  where {branch name} is the name of your target branch (ex: casablanca
```

After Your Changes Are Accepted and Merged

1. Update your JIRA ticket to completed
2. Thank you for your contribution!