EDLM

132nd Virtual Wing
Documentation overhaul

APPLIES TO:	**132nd Virtual Wing**
TYPE:	**Enhancement Proposal**
VERSION:	0
PUBLISHED DATE:	unpublished
DOCUMENT RESPONSIBLE:	etcher
SUMMARY OF CHANGES:	nil

Contents

Figure 1: Documentation

1 Enhancement Proposal

1.1 Abstract

This document contains an enhancement proposal submitted to CMD for review.

The proposal is about documentation in the 132nd Virtual Wing. I would like to explore possible ways to make it better, easier to produce and easier to maintain.

1.2 Rationale

I was working a while back on the 696th TURNSKINS TRP (which produced no useful end-result), and was taken aback by many of Microsoft Word pitfalls. Why does it decide to auto-format so many things on my behalf ? And why can't it keep it consistent ? Reznik and I were editing with different locale settings (fr_BE and da_DK), which added even more fun to the mix: different paper size, different quote characters, and so on ...

I've also been offered the chance to review the new 176th EYES OF THE NORTH SOP before it officially came out, and the whole review process, using a proxy Google Docs temporary document seemed archaic and painful to me. This extra step adds a lot of undesirable overhead to the process.

Last example: as a side-project, I'm maintaining a kneeboard document written with Microsoft Excel, and that takes the pain to a whole new level.

I would like to propose a solution to get rid of those issues, and implement a better workflow for everyone. This document is published for review in order to assess how interested/willing people are interested to give it a try.

1.3 Objective

What I'm trying to achieve:

- Increase quality and consistency of the documentation
- Allow for easier collaboration and review
- Maintain an easy workflow for everyone

What I'm trying to get rid of:

- Variations in style or formatting
- Older documents not updated to latest layout/formatting
- Passing documents around during review/editing process

1.4 Getting there

This section conceptually describes how to achieve the goals outlined above.

1.4.1 Decoupling the content from the format

Writing a document with Microsoft Office Word implies formatting it as well. Despite the use of a template, guaranteeing a consistent format across a library of dozens of documents, written by many different authors, sometimes even multiple authors for the same document, is almost impossible.

Due to the inherent complexity of Microsoft Word Office, as well as the debatable pertinence of some choices it sometimes seems to automatically make for us, some subtle format changes are bound to sneak unnoticed within the document over time.

My first goal is to get rid of this ambiguity, and have one and only one way to express a given construct (a paragraph, a heading, a list, a table, ...).

This would also have the beneficial side-effect of freeing editors from worrying about formatting their content while they are writing it, giving them more brain power for the actual content creation

Decoupling content and format also means that any older document, even one that hasn't been touched in years, will be automatically updated to the latest format/layout whenever the format/layout is updated, since their content has not changed.

1.4.1.1 Specification

Once the content has been created by the editors, my goal is to provide a system that will take that "raw" content, and format it, consistently, into different formats that will later be published. A choice format is of course PDF, but I'm thinking HTTP (web) or EPUB (books) as well.

The output should be:

- Consistent across builds: the same content must **always** yield the same result, even on different computers, operating systems, or software versions
- Uniformly formatted: **all** the documents in the library should have the same general layout, giving all documentation published by the 132nd Virtual Wing a visual identity of their own
- Retroactively managed: all documents that have been published in the past should be **updated without human intervention**. If a logo changes, if we decide to change the title page, or the space after paragraphs, those changes should be **automatically propagated across the entire library**
- Adapted to our needs: the documentation should not look "generic" and bland

1.5 Pros & cons

This section objectively (as much as I could) describes the pros and cons of the method I propose to implement.

1.5.1 The cons

Allow me to start with the cons, and provide, for each of them, a way to mitigate them.

1.5.1.1 No WYSIWYG ("What you see is what you get")

Markdown is pure text, therefore an editor who is busy writing documentation do not see the result appear as he types. Font does not resize for headings, pictures do not appear, tables do not build, etc.

1.5.1.1.1 *Mitigation*

- Converting from Markdown to Word/PDF/HTML is trivial, and the tool that I will be providing can be installed on any modern computer. Output can therefore be refreshed as often as needed to get a "sneak peek" into the actual result.
- Numerous tools exists *online* to edit Markdown. Some of those tools are bare editor, providing a side by side "Edit" window and a "Result" window. Their offline equivalent are available as well. Moreover, and this gets really interesting, Markdown is mature enough that many *free* online editors offer amazing synchronization capabilities with Google Drive, Dropbox, Github, ... , **effectively allowing anyone to work on any document from any computer connected to the web, and directly sends their work for review into the Github repository of the 132nd Virtual Wing**.

1.5.1.2 Less liberty when it comes to customizing the format/layout

Having a common template for the layout/format of our document effectively "castrates" editors, denying them the liberty to get creative with the way their content is rendered. This could get on the nerves on some, especially the most perfectionists among us.

1.5.1.2.1 *Mitigation*

- The rendering, formatting and layout is done by a professional (although free) typesetting application that has been in existence since 1985. 32 years of development, testing and improvements have made it quite robust. It has been in use for decades by the scientific and teaching community all around the world for papers, essays, reports, etc.
- The layout/format will be 100% identical for all documentation published by the 132nd Virtual Wing, branding our documents with a unique "personality", and giving an overall "neat" picture of the Wing to the external world.
- In case of necessity, when part of the output does not fit a specific need of ours, we can take advantage of the maturity of the tools and customize every little detail to our needs (this would of course be my job).

1.5.1.3 Resources like pictures, videos, sound, are to be included on the side

All the files that are to appear in the final document will be referenced in Markdown as links only, pointing to a file that exists near the Markdown source document (I'll come back on the structure later). To give an example, if I wanted to include a file named `picture.png` in a document named `index.md`, I would have to write the following in `index.md`: `[Picture caption](path/to/picture.png){width: 8cm}`. "Picture caption" simply describes the picture, and can be any text. The `path` part is irrelevant, since that will be handled by the tools I am writing, but the problem persists: media only appear as text and are to exist

alongside the source. The `{width: 8cm}` part is optional, but is amazing to "force" the same width (or height) on all pictures throughout a document (which looks real nice).

1.5.1.3.1 *Mitigation*

- Declaring pictures by name offers a finer grained controls on their size, and allow for dynamic resizing of pictures from all origins
- Updating pictures can be done without even opening the source. A file named `picture.png` will be included in the final document, whatever that file contains. Updating batch of pictures is thus very easy, and does not require updating them one by one in every Word document (imagine how easy it would be to be able to run batch updates on the pictures library ...).
- Pictures are automatically indexed and referenced in the final document, and an automatic "Table of figures" is automatically generated, with hyperlinks to the pictures within the text.

1.5.1.4 New technologies

While all the cons so far are of relatively small import, I fear this one might raise the most shields in our community. Please bear with me for a little while ? For this project, I plan on using two pieces of software for the front-end (the parts people are expected to interact with): **Markdown** and **Git**. For more information about them, see the "Technical" section of this document.

1.5.1.4.1 *Mitigation*

- While switching to new software always implies somewhat of a **learning curve**, I used the following criteria to select them:
  - Free (as in *no dinero*)
  - Open source
  - Mature
  - Widely used across the world
  - Well documented
  - Will be supported for years to come
  - Easy enough for the intended usage
  - Has a luxuriant and flourishing ecosystem of tools around them

1.5.2 The pros

TODO

1.6 Technical

This section describes the tools chain that would be used to build and publish the documentation.

1.6.1 Overview

Despite the vast possibilities that those tools permit, with a staggering amount of options, configurations and features, the whole process will be mostly automated, and the editors/reviewers will have to deal with very few technicalities.

1.6.2 Specific tools, part1: the front-end

The front-end is what editors will have to work with.

1.6.2.1 Markdown

Markdown is a markup language widely used across the Internet. Its syntax is simple (similar to the syntax we use on our forums) and is meant to be readable. This very document is written in Markdown; click here to see the "raw" Markdown text.

Excerpt from Wikipedia:

> Markdown is a lightweight markup language with plain text formatting syntax. It is designed so that it can be converted to HTML and many other formats using a tool by the same name. Markdown is often used to format readme files, for writing messages in online discussion forums, and to create rich text using a plain text editor. As the initial description of Markdown contained ambiguities and unanswered questions, many implementations and extensions of Markdown appeared over the years to answer these issues.

Here is a cheat-sheet with the available formatting.

1.6.2.1.1 Why Markdown ?

I selected Markdown mainly because **it decouples the content and the format**, but also for the following reasons:

- It's readable and easy to "learn"
- It's used all over the web; tutorials are plenty, and all questions have been answered
- It's supported by all the major players
- It's very easy to transform into PDF, HTML, EPUB, RST, Microsoft Word, etc.

1.6.2.1.2 Try it out

You can try Markdown right now, without installing anything. Just head to one of those online editor, and write away:

- dillinger (my favorite)
- classeur
- stackedit
- jbt.github.io/markdown-editor
- markdownlivepreview
- hackmd
- etc. . .

1.6.2.2 Git & Github

This is where things get a little bit hairy. It will seem very complicated at first, but I can assure you that after doing it a few times it makes a lot of sense and becomes actually quite easy (we're in the business of simulating a very complex environment, I reckon a few commands won't scare you that much =)).

Here is an introductory tutorial about Git & Github.

Here is another tutorial about the Git philosophy, and one way to use it.

Please don't be scared by all the commands, there is a GUI application that lets you do all those things with a click of the mouse.

I selected Git & Github because it allows for a *outstanding way of working together on the same project.* With Git & Github, you can:

- Track *all* the changes, and roll back whichever one you'd like (constant backup of everything)
- Associate changes with their author at a glance
- Collaborative editing (many people can work on the same document at the same time)

- Incremental review process supporting discussion
- Issue tracker
- Allows for a lot of automation under the hood
- Git is the *de facto* Source Control Manager nowadays (alternatives: Subversion, Mercurial; compare them)
- Github is the *de facto* hosting website for open source Git projects (alternatives: Bitbucket, Gitlab; compare them)

Note: I'm leaving out alternatives that are not open-source, self-hosted and free to use.

1.6.2.3 What you'll need

This is the list of what you would have to install on your computer in order to be able to work on the documentation.

1.6.2.3.1 Option 1: work online only

This is absolutely possible. For example, with http://dillinger.io/, I'm writing this document in Markdown, I get to have a live preview of what I'm writing, and I can push it to Github with one click of the mouse. No fuss, no problemo.

1.6.2.3.2 Option 2: work offline

While option 1 is perfectly fine, you can also decide that you need more control of what is going on. In that case, you'll need a minimal suite of tools.

Edit Markdown

This one is easy: you don't need anything. Markdown is pure text, so any text editor will do

Work with git & Github

First, make sure you followed this tutorial and are a little bit familiar with Git.

Then install the following: * Git for windows * Github windows client application (alternatives: smartGit (my favorite, more complex), Gitkraken)

That's it ! Create an account on https://github.com/, start your local Git client, and you're ready to rock !

1.6.2.4 The workflow

List of Tables

List of Figures