OvGME: proposal

132nd Virtual Wing
Mods and skins management

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PUBLISHED DATE:	unpublished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 managing mods and skins in the 132nd Virtual Wing. I would like to make it easier for maintainers to offer and maintain mods, while keeping it simple for the end-user to install them.

1.2 Rationale

The current system for skins distribution is working fine, from the end-user point of view at least. Using the //ski to install files is a one-click process and that is really, really good. There two issues with it though:

1. While it is easy for *users* to use the //ski, updating the skins with it is a pain for the *maintainer*
2. The //ski only installs files under the "Saved Games" directory. This is a problem for some of the mods we need nowadays, they need to go in the DCS installation directly. Now the //ski ws never meant to manage mods, so this isn't something I'm holding against it, but the needs we have have changed.

This proposal has been cooking since July 2017, but I've decided to rush it because of the issues we've had lately with some of the mandatory mods (Helipad, WASP, Range targets, ...). It has become difficult and troublesome for people to make sure that their DCS installation is compatible with the 132nd Virtual Wing's server. My apologies for the rough state of this proposal, as well as for the half-finish documentation.

1.3 Objective

> **Note:** in tis document, I'm using the terms "end-user and"maintainer" to describe: End-user: a pilot that will use our skins and mods; they might be part or the 132nd Virtual Wing or memeber of an external organization. Maintainer: a maintainer is a member of the 132nd Virtual Wing who is responsible for maintaining a mod/skin up to date. They also are the ones who initially creates mods/skins packages, and upload them.

What I'm trying to achieve:

1. Simplify the building/updating process for maintainers
2. Keep the installation process as simple as possible for the end-user
3. Allow for he installation of mods in the DCS installation as well as in the Saved Games folder

1.4 Getting there

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

1.4.1 Using //ov

This proposal is very simple. The plan is to use //ov instead of the //ski to distribute mods and skins within the 132nd Virtual Wing.

Documentation website for //ov.

1.4.1.1 Specification

OvGME is a mod management application, much like JsGME, that allows to install/uninstall mods (obviously), but also to subrscribe to remote repositories of versionned mods. This makes propagating and updating the mods much simpler.

1.5 Pros and 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

- Depending on yet another dependency: we will have to transition from using the //ski to using /ov, which might pose a problem to the less tech-asvvy of our userbase. See "Transition" section below.
- Modular dependencies: no more *one click install* of mods and skins. Since some mods are optional (for mission makers for example) and others contextual (a pit for the Ka50 for example), users will need to install some mods and skip others. **To mitigate that, my solution is currently to tag every mod/skin as "MANDATORY" or "OPTIONAL"**, making it obvious which one MUST be installed to join the server.
- External organizations do not need access to all of our mods, and some mods might resitrict re-distribution, forcing us to redirect external users to their official website and to assume they'll be able to install them themselves. Internally, distributing "protected" mods is not an issue.

1.5.2 The pros

- Single source of thruth: this a big one. No more fiddling around with the //ski on one side, then going to google Drive/Dropbox to grab mods and install them manually, or with JsGME///ov. This should help a lot with the current situation, where it is not an uncommon occurence to have to debug with users before the flight on TS because a missing mod won't let them join the server.
- Most of us are already using JsGME or //ov, to integrating the new workflow won't be a problem for those.
- Mods and skins are modular: this is also a pro, because it allows squadrons to ditribute mods that are not mandatory, but nice to have (ex: new cockpit, maps for the TAD, ...). This allows getting newcomers onbaord much faster.
- Versionning of mods and skins: each mod and each skin uploaded to our repository receives a version. This makes it *very* easy to make sure that everyone has the correct, latest version of the mods. When a maintainer releases a new version of a mod or a skin, a NOTAM would be issued, and the people who currently have this mod installed (or everyone in acse of a "MANDATORY" mod) just need to fire up //ov and update the mod (a few click and a few seconds).

- Much simpler job for maintainers: creating, uploading and updating mods and skins in the repository is much simpler than with the //ski. It takes but a few clicks and a few minutes to have a new version of a mods on in the repository, available for users to download.
- Sharing the load: splitting the mods and skins into smaller, more manageable units allows for a better repartition of tasks. Each mod/skin should have an official "maintainer", and that person is responsible for keeping it up-to-date. For example, someone might be responsible for the skins of the 259th POLAR BEARS, while someone else would be responsible for the "MISSION MAKERS" mods. No more huge, monolithic list of mingled skins.
- Using external repositories: in the future, maybe an external organization will be using //ov to manage their list of skins and mods. Adding their reporitory to //ov is a trivial process, and it would allow us to user their skins directly, instead of having to re-package them ourselves.
- //ov is capable of managing multiple target destinations; this would be very useful in case we need one of the following:
  - Manage different version of the mods/skins for different DCS versions
  - Install some mods in the Save Games folder, and other directly in the main DCS installation

1.6 Technical

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

1.6.1 Overview

This section describes the tools and workflow needed to implement the new system.

1.6.1.1 End-user

The only tool needed for the end-user is //ov itself. Some might need guidance during the initial transition process, which will be provided by the documentation. Using //ov is straight forward enough that everyone should be able to maintain with ease.

1.6.1.2 Maintainer

Maintainers will need //ov, as wel as a FTP client to be able to upload files on the 132nd Virtual Wing's FTP (Filezilla is a good starting point).

1.6.1.2.1 Brief description

The documentation will eventually cover this in a more detailed fashion, but here is a brief overview of how it works:

Note: to follow along, open http://132virtualwing.org/files/ovgme/ in your borwser.

The 132nd Virtual Wing's repository is described by the 132nd.xml file. That file is automatically generated by //ov.

All the mods themselves are contained in the "132nd" folder, right next to the file.

The mods are in two format: * bare folder: the mod itself, deflated, containing all the files, in the same format as JSGME; this is the "source" * ZIP: mod packaged and ready to be distributed; this is the "package"

To create a mod, the first thing to do is to create the source folder, containing the files of the mods. For example, a simple skin. This folder follows the JsGME standard, replicating the target directory structure, adding and replacing files on the fly.

Onca created, the source folder is available locally for installation, thus permitting to test the mod before packaging it.

Then, using //ov, the source folder is packaged into a ZIP file, along with "metadata" about (metadata is things like version, description, author, etc.).

That ZIP file is also availabe locally for installation, and it should be tested as well.

Now, stil using //ov, there is a command to create a "repository" file from all the mods currently available locally. That command will look for every ZIP-ped mod, and create a "*.xml" file containing metadata for all of them (see, for example, 132.xml).

At that point, all there is to do is upload both the ZIP file and the XML file to the FTP, and you're set.

Note: I've upload the mods in folder formats on the FTP as well, so if someone wants to update one of the mods, they have access to the source folder I've used to create the ZIP package.

Profiles

I've explained briefly how to go from a mod folder to a ZIP package, and how to make a repository out of a set of packages (for example the "132nd" folder and the "132nd.xml" file on the FTP). This section describes how to manage and switch repositories.

To manage different repositories, you need "profiles". When you build the "*.xml" file, //ov will use* all* the mods that are present in the curent profile (the current mods folder of //ov), or to be put it another way, all the mods that are visible in //ov at that time.

So, if we want to create another repository for "externals", for example, we need another profile. That is what I did with the "external" folder and the "external.xml" on the FTP.

So, in the end, I have 3 profiles in //ov:

- My regular profile, pointing to DCS, to install mods and skins
- A profile for the "132nd" repository, in sync with the "132nd" dir of the FTP
- A profile for the "externals" repository, in sync with the "exeternal" dir of the FTP

Workflow

Now that we know all that, the work flow to add or update a mod is as follows:

1. Connect to the FTP and download the latest changes into the local profile
2. Make the changes needed or add the new mod/skin
3. Test the new/updated mod
4. Create a ZIP package for the mod, adding or updating the metadata (version, author, decsription, ...)
5. Update the XML file with //OV
6. Upload the new/updated ZIP package and the XML to the FTP

1.7 Transition

This section describes how to transition to the new system if we decide to switch.

1.7.1 Overview

Description of the transition process step by step:

1. Freeze the current mods and skins list
2. Transfer them into the //ov repository of the 132nd Virtual Wing
3. Create another repository for external organizations
4. Publish //ov documentation along with a NOTAM
5. Help users who have trouble getting up to speed

Step 1 and 2 are almost taken care of, I have a working, test repository that's been going on for a while now. you can find it at: http://132virtualwing.org/files/ovgme/ (**note**: the only two relevant folders are "132nd" and "externals", the others are artifacts from previous testing.).

Step 4 is ongoing, as I've already written the documentation for the end-users. I still have to write documentation for maintainers.

Step 5 would take very little time, as I expect little to no issue by using this process. The only trouble I can foresee is for people who are not using a mod manager at this time, relying instead on just dropping mod content in their install; they'll have a bad time removing old mods betfore switching.

1.7.2 Documentation

I'm writing a document describing, step by step, how to use //ov as:

- an end-user, to downloads, install and update mods and skins (done)
- a maintainer, to create, upload and update mods and skins (ongoing)

1.7.3 Crash course

I plan on offering crash course with OvGME to the people who are interested in it.

The crash course should take about 10 minutes on TS for end-users, and maybe 30 minutes for the maintainers.

List of Tables

List of Figures