Version number: 2.0

Issue Date: 13 May 2020

# Introduction

## Purpose

The purpose of this procedure is to describe how to use the Manifest GUI tool to create and verify manifests.

## Background

Public Record Office Victoria (PROV) moves digital objects from agencies to PROV as part of the transfer process. The GUI tool is a way to check that all of the digital objects have been moved, and that none have been corrupted in the process.

The GUI tool creates and verifies manifests of the digital objects in the movement. A manifest is a file containing a list of the digital objects and a hash of each object. Information can be included in the manifest describing its purpose.

Normally the digital objects will be VEOs (VERS Encapsulated Objects), but the tool may be used to transfer any digital object.

The Manifest GUI tool is in development, and the current version is primarily released to gain feedback as to the manifest and the tools.

## Audience

This procedure has been developed for:

* Agency staff to create manifests before moving digital objects to PROV
* Transfer archivists to verify received manifests
* Other users wishing to ensure the validity of a directory of files.

# Manifest tool

## What is the Manifest tool?

The Manifest is a tool developed by PROV that allow the generation and verification of manifests of digital objects.

## How is the Manifest tool used?

The Manifest tool may be run as Java executables or called using an Application Programming Interface (API).

The tools must be run using Java 1.8 or later. They will not work with Java 1.7 or earlier versions.

On a computer operating a Windows operating system, the tools can be invoked using the ‘cmd.exe’ program.

## Legal

The toolset is licensed under the Creative Commons CC BY 4.0 license. This means that you have a license to do anything that you want with the toolset, provided that you:

* Acknowledge Public Record Office Victoria as the source of the toolset.
* Do not misrepresent the license or your relationship with Public Record Office Victoria.

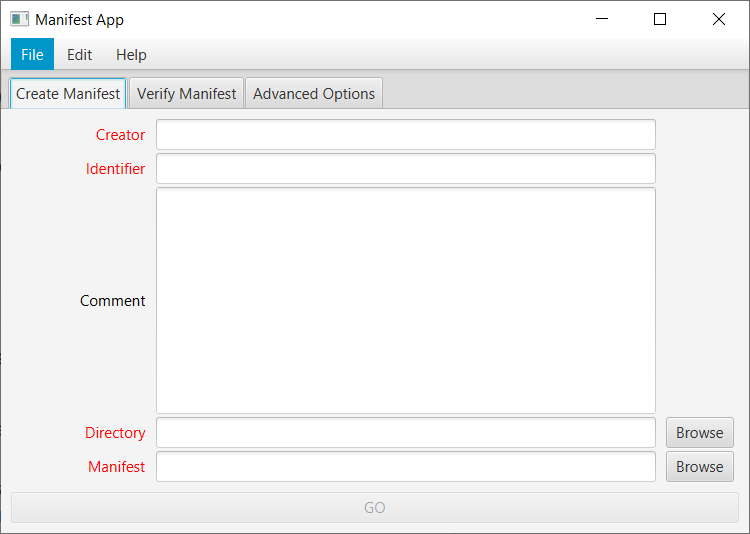
Specifically, you may:

* Include the code from the toolset in your products, either “as is” or in a modified format.
* Use the code from the toolset as the basis of code in your toolset.

# Creating a manifest using the GUI

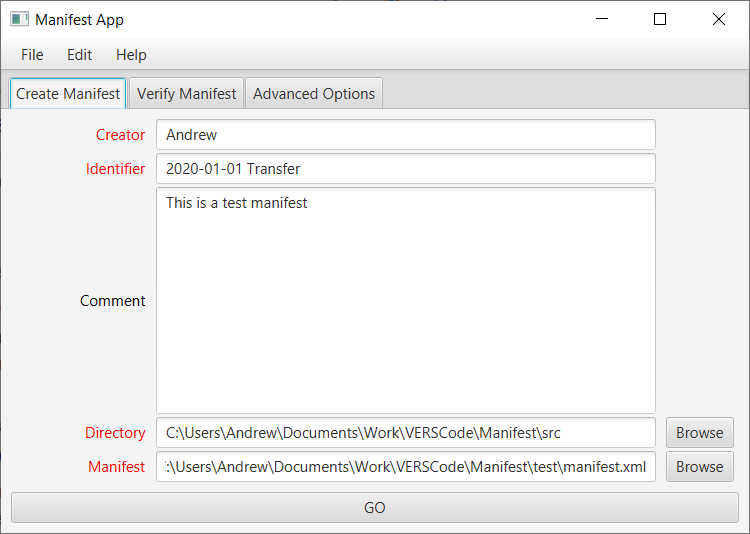
## Starting the GUI

## Creating a Manifest

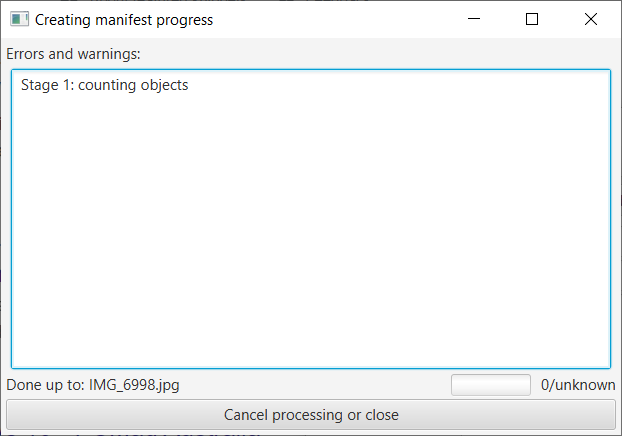
The initial screen of the Manifest GUI allows manifests to be created or verified. In the screen capture below, the ‘Create Manifest’ tab has been selected to create a manifest.

There are five fields that can be entered. The mandatory fields (Creator, Identifier, Directory & Manifest) must contain data, and have red field names. The fields are:

* Creator (Mandatory). A free text field that identifies you, the person creating the manifest. This could be your name or role.
* Identifier (Mandatory). A free text field that identifies this manifest. In the PROV context this would typically identify the Transfer, Series, Consignment and Set.
* Comment (Optional). A free text field in which the creator can make any comments necessary about the collection of digital objects included in this manifest.
* Directory (Mandatory). The computer directory which contains the digital object to be included in the manifest. This can be entered as free text, or the ‘Browse’ button on the right hand side of the text field can be pressed to navigate to the directory.
* Manifest (Mandatory). The name and location of the Manifest file to be created. Again, this can be entered as free text or the ‘Browse’ button next to it can be used to select or create a file. The file created must have the file extension ‘.xml’, but this will be set up by default if no file extension is entered.

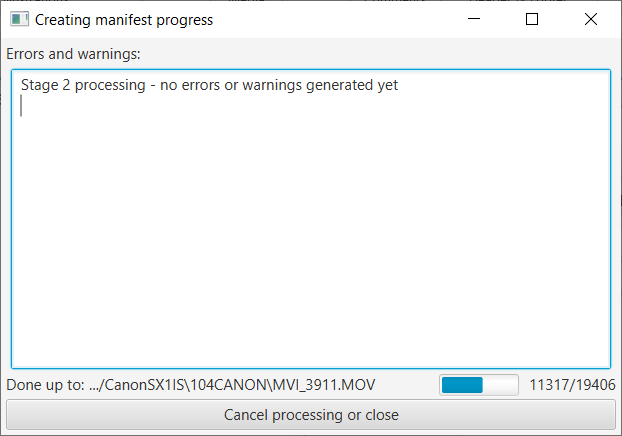
When all of the mandatory fields have been entered, the ‘GO’ button at the bottom of the window will become active. Note that the text ‘GO’ on the button has changed from grey to black.

To avoid having to re-enter the contents of the fields for multiple manifests, it is possible to create and save ‘Jobs’. After entering some or all of the text fields, select the ‘File’ menu and ‘Save Job’ menu item. This allows the current state of the window to be saved as a JSON file. Subsequently selecting the ‘File’ menu and ‘Load Job’ allows the JSON file to be read and the values restored.

Pressing the ‘GO’ button starts the creation of the manifest. This pops up the ‘Creating manifest progress’ window: 

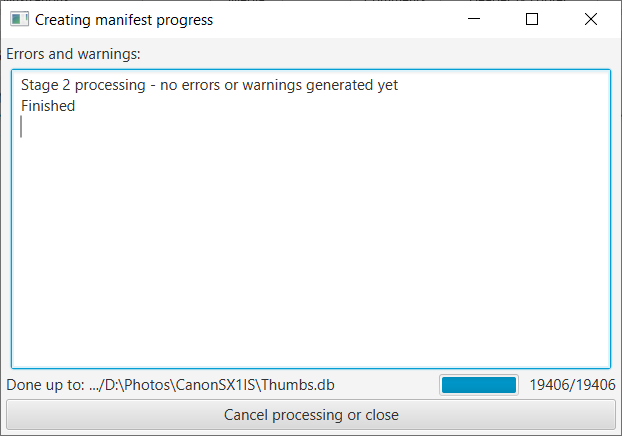
The text area in the centre of the window contains any status or error messages generated when creating the manifest.

The first stage in creating the manifest is to count the number of files within the selected directory that will be included in the manifest. This stage is indicated by the text ‘Stage 1: counting objects’ in the ‘Errors and warnings’ panel. Underneath this panel the label ‘Done up to:’ indicates the file that is currently being counted; this gives an indication of how far the stage has progressed. Creation of the manifest can be cancelled at any time by pressing the close window icon (X) at the top of the window, or the ‘Cancel processing or close’ button at the foot.

After the files have been counted, the actual manifest will be created. The ‘Creating manifest progress’ window will change:

The text in the Status panel will change to ‘Stage 2 processing – no errors or warnings generated yet’. This text will be removed if any warnings or errors are generated. Underneath the status panel, there are three pieces of progress information. On the right is the current count of the number of files processed out of the total to be processed (in this case the program is processing file 11317 of 19406 files). To the left of this is a progress bar indicating visually the progress (in this case, about half of the files have been processed). On the left the current file being processed is displayed.

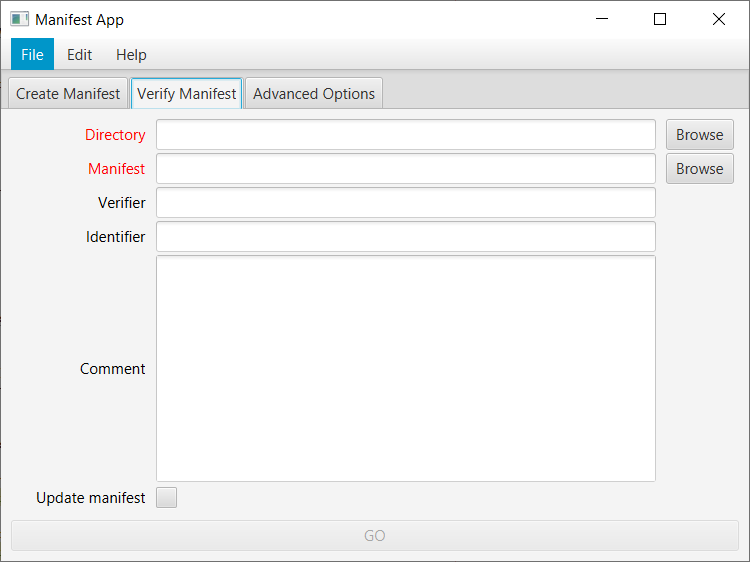
Again, creation of the manifest can be cancelled at any time by pressing the ‘X’ button or the ‘Cancel processing or close’ button.

When creation of the manifest has been competed the progress screen will display:

The text ‘Finished’ has been added to the status panel; the count shows that all files (‘19406/19406’) have been processed, and the progress bar is at 100%.

The manifest has been created and the window can be closed by pressing ‘X’ or ‘Cancel processing or close’.

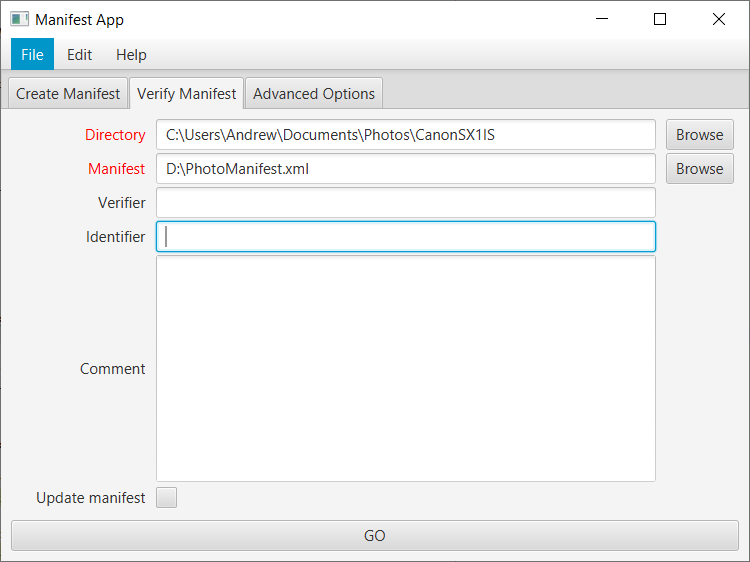
## Verifying a Manifest

The initial screen of the Manifest GUI allows manifests to be created or verified. In the screen capture below, the ‘Verify Manifest’ tab has been selected to verify a manifest.

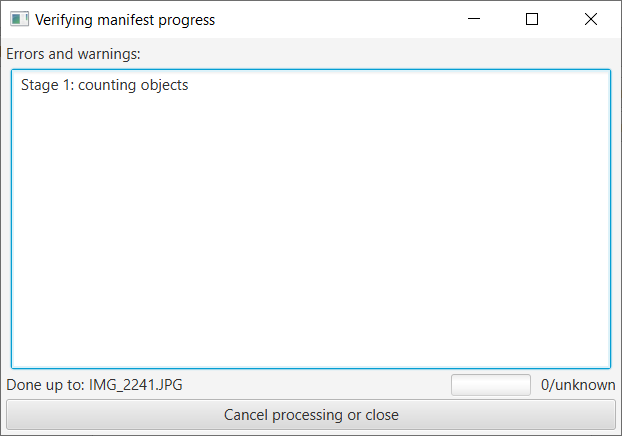
There are five fields that can be entered and one check box. The mandatory fields (Directory & Manifest) must contain data, and have red field names.

The fields & checkbox are:

* Directory (Mandatory). The computer directory which contains the digital objects to be verified against the manifest. This can be entered as free text, or the ‘Browse’ button on the right hand side of the text field can be pressed to navigate to the directory.
* Manifest (Mandatory). The name and location of the Manifest file to be used. Again, this can be entered as free text or the ‘Browse’ button next to it can be used to select or create a file.
* Verifier (Optional). This field is ignored at the moment.
* Identifier (Optional). This field is ignored at the moment.
* Comment (Optional). This field is ignored at the moment.
* Update manifest. This checkbox is ignored at the moment.

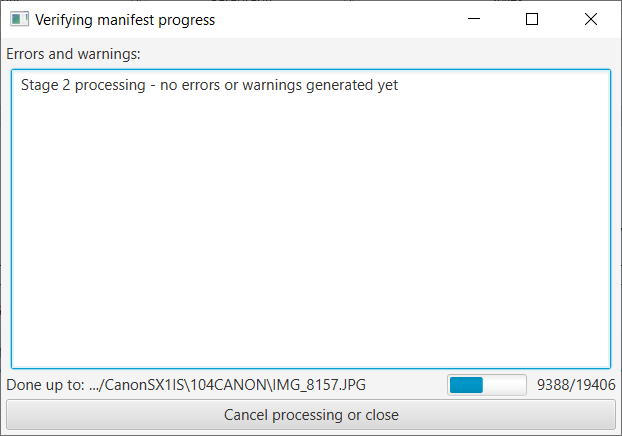
When all of the mandatory fields have been entered, the ‘GO’ button at the bottom of the window will become active. Note that the text ‘GO’ on the button has changed from grey to black. 

To avoid having to re-enter the contents of the fields for multiple manifests, it is possible to create and save ‘Jobs’. After entering some or all of the text fields, select the ‘File’ menu and ‘Save Job’ menu item. This allows the current state of the window to be saved as a JSON file. Subsequently selecting the ‘File’ menu and ‘Load Job’ allows the JSON file to be read and the values restored.

Pressing the ‘GO’ button starts the verfication of the manifest. This pops up the ‘Verifying manifest progress’ window: 

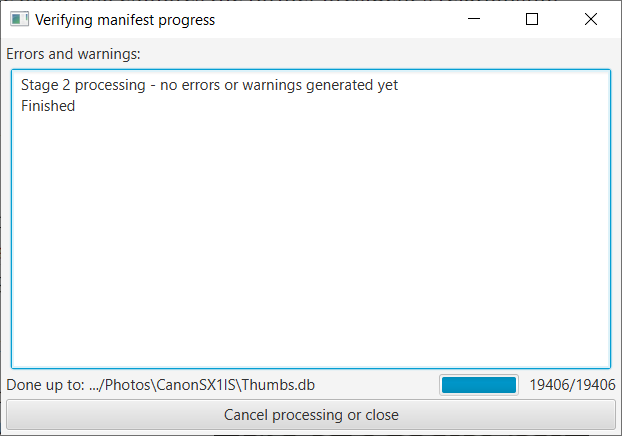
The text area in the centre of the window contains any status or error messages generated when verifying the manifest.

The first stage in verifying the manifest is to count the number of files within the selected directory that should be in the manifest. This stage is indicated by the text ‘Stage 1: counting objects’ in the ‘Errors and warnings’ panel. Underneath this panel the label ‘Done up to:’ indicates the file that is currently being counted; this gives an indication of how far the stage has progressed. Verificationof the manifest can be cancelled at any time by pressing the close window icon (X) at the top of the window, or the ‘Cancel processing or close’ button at the foot.

After the files have been counted, the manifest will be verified. The ‘Verifying manifest progress’ window will change: 

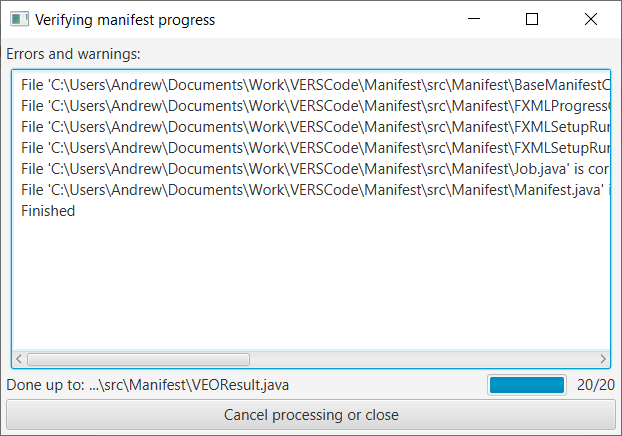
The text in the Status panel will change to ‘Stage 2 processing – no errors or warnings generated yet’. This text will be removed if any warnings or errors are generated. Underneath the status panel, there are three pieces of progress information. On the right is the current count of the number of files processed out of the total to be processed (in this case the program is processing file 9388 of 19406 files). To the left of this is a progress bar indicating visually the progress (in this case, about half of the files have been processed). On the left the current file being processed is displayed.

Again, verifciation of the manifest can be cancelled at any time by pressing the ‘X’ button or the ‘Cancel processing or close’ button.

When verification of the manifest has been competed with no errors the progress window will display: 

The text ‘Finished’ has been added to the status panel; the count shows that all files (‘19406/19406’) have been processed, and the progress bar is at 100%.

The manifest has been verified and the window can be closed by pressing ‘X’ or ‘Cancel processing or close’.

If, however, some of the files do not verify, details of the files will be displayed in the status window:

Note that the text ‘Stage 2 processing – no errors or warnings generated yet’ has been removed.

The information in the status report can be cut and pasted into a text file if it is necessary for it to be kept.

# Creating a manifest using Java code

The VEOCreate tool creates multiple Version 3 VEOs from a control file. The control file is a text file containing multiple rows of tab separated commands. Each command builds a part of a VEO (or controls how subsequent VEOs are to be built). This class also processes the command line arguments and reads the metadata templates.

## Command line arguments

The following command line arguments must be supplied:

* **-sf <directory> the directory in which the standard VEOReadme.txt file will be found.**
* **-t <directory>** the directory in which the metadata templates will be found. See the section below for details about the metadata templates.
* **-c <file>** the control file which controls the production of VEOs. See the next section for details about the control file.

A minimal example of usage is

createVEO -c data.txt -t templates

The following command line arguments are optional:

* **-v** verbose output. By default off.
* **-d** debug mode. In this mode more logging will be generated, and the VEO directories will not be deleted after the ZIP file is created. By default off.
* **-ha <algorithm>** The hash algorithm used to protect the content files and create signatures. Valid values are: ‘SHA-1’, ‘SHA-256’, ‘SHA-384’, and ‘SHA-512’. The default is 'SHA-1'. If no –ha command line argument is present, the hash algorithm must be set in the control file. If a hash algorithm is specified in both the command line and in the control file, the control file specification overrides the command line.
* **-s <PFXfile> <password>** a PFX file containing details about the signer (particularly the private key). A PFX file is protected against reading and can only be unlocked using the password. If no -s command line argument is present, the PFX file must be specified in the control file. If PFX files are specified in both the command line and in the control file, all PFX files are used to sign the VEOs resulting in multiple signatures.
* **-o <outputDir>** the directory in which the VEOs are to be created. If not present, the VEOs will be created in the directory where the createVEO program is run.

## APIs

An API can be used to create and verify Manifests.

The Javadoc may be consulted for more details about the API

End of procedure