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:

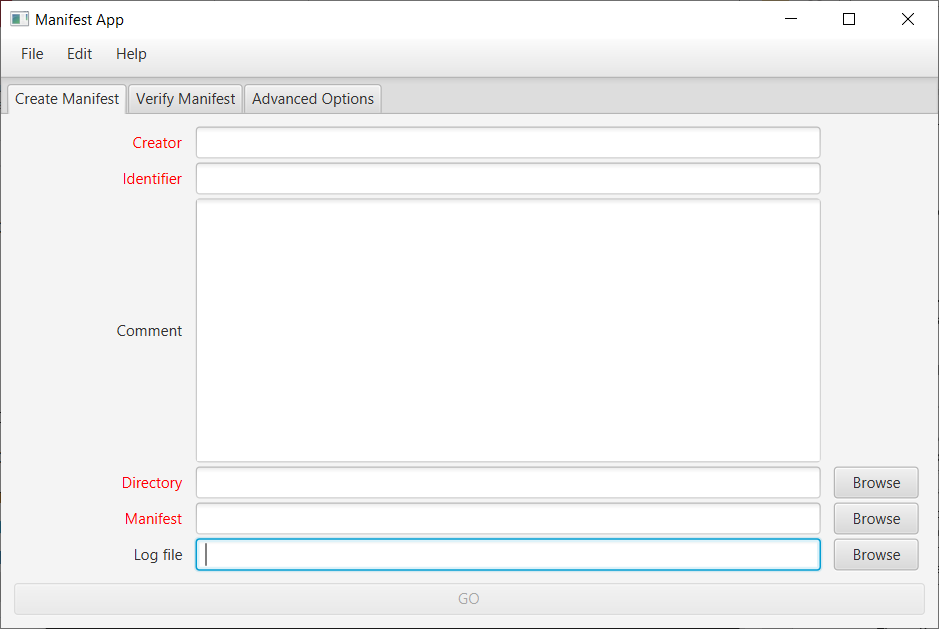
* Include the code from the toolset in your products, either “as is” or in a modified format.
* Use the code from the toolset as the basis of code in your toolset.

# Creating a manifest using the GUI

## Starting the GUI

Double click on the ‘ManifestGUI.bat’ file.

## 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.

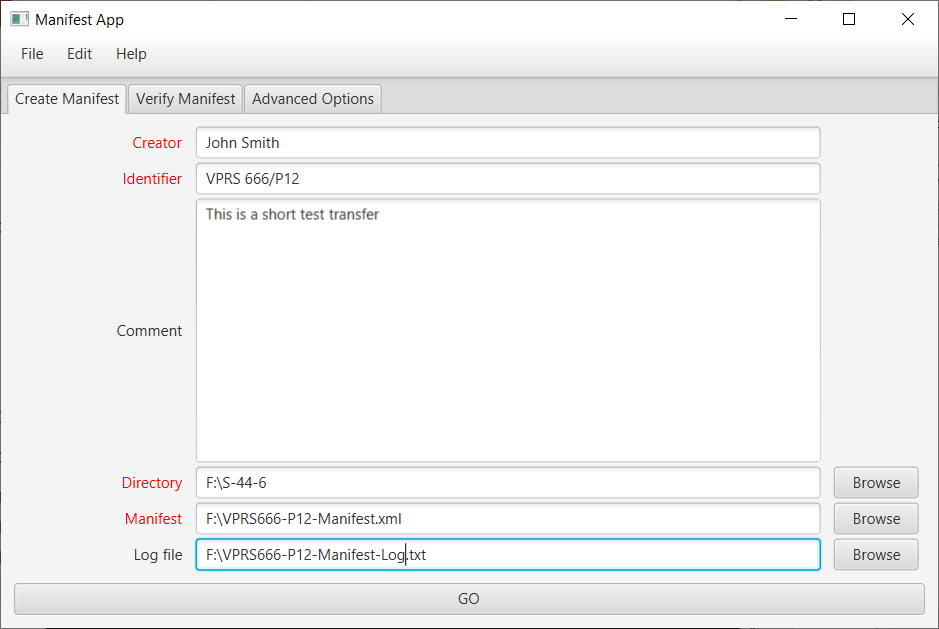
There are six fields that can be entered. The mandatory fields (Creator, Identifier, Directory & Manifest) must contain data, and have red field names. Note that the ‘GO’ button at the bottom of the window will not become active until all of the mandatory fields have been entered.

The mandatory fields are:

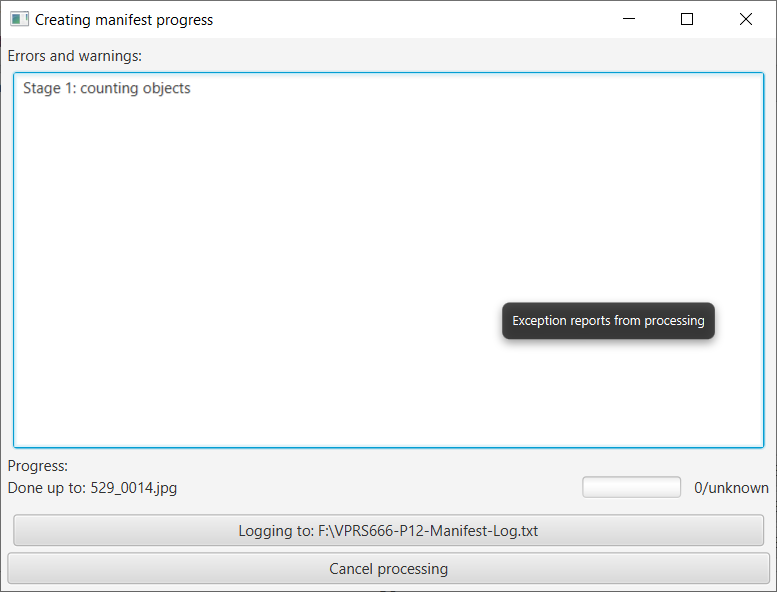
* Creator (Mandatory). A free text field that identifies you, the person creating the manifest. This could be your name or role.
* Identifier (Mandatory). A free text field that identifies this manifest. In the PROV context this would typically identify the Transfer, Series, Consignment and Set.
* Directory (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.

The optional fields are:

* 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.
* Log File (Optional). The name and location of a log file to be created. This log file contains a copy of the logging generated during the run. You can use the ‘verbose’ option on the ‘Advanced Options’ to select more or less detail.

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.

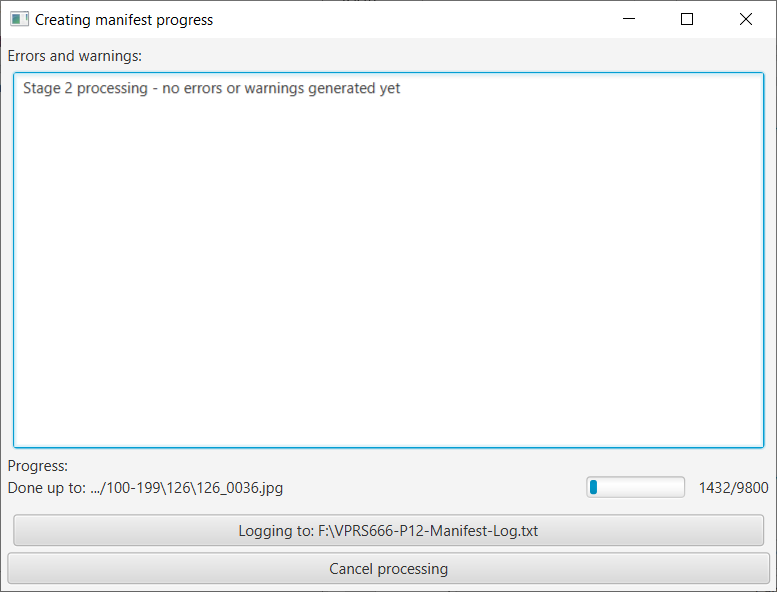
To avoid having to re-enter the contents of the fields for multiple manifests, it is possible to create and save ‘Jobs’. After entering some or all of the text fields, select the ‘File’ menu and ‘Save Job’ menu item. This allows the current state of the window to be saved as a JSON file. Subsequently 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 amount of information shown depends on the setting of the ‘verbose’ flag on the ‘Advanced Options’ tab. This and the following two screen shots were captured with the ‘verbose’ flag unticked.

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 (note that this window may appear and disappear very quickly if only a small number of files are to be included in the manifest). 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.

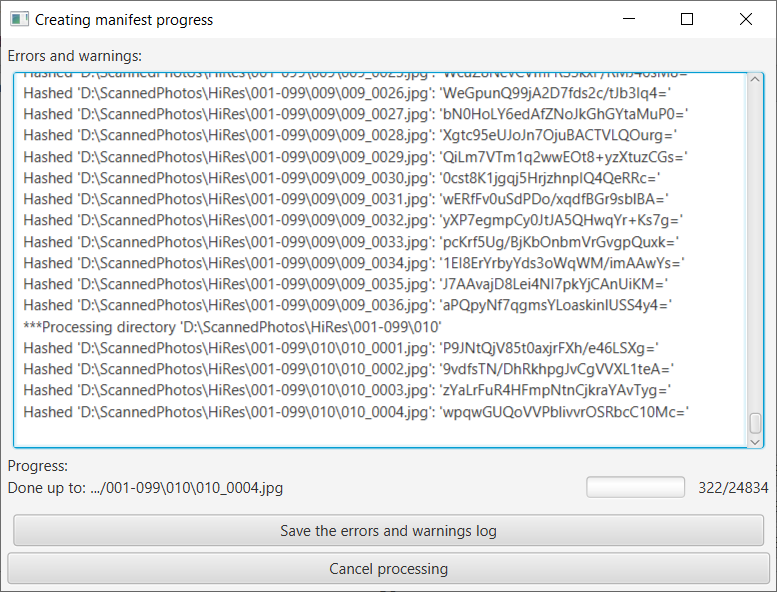
Note that the error and status files are being logged to the file specified in the set-up window

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

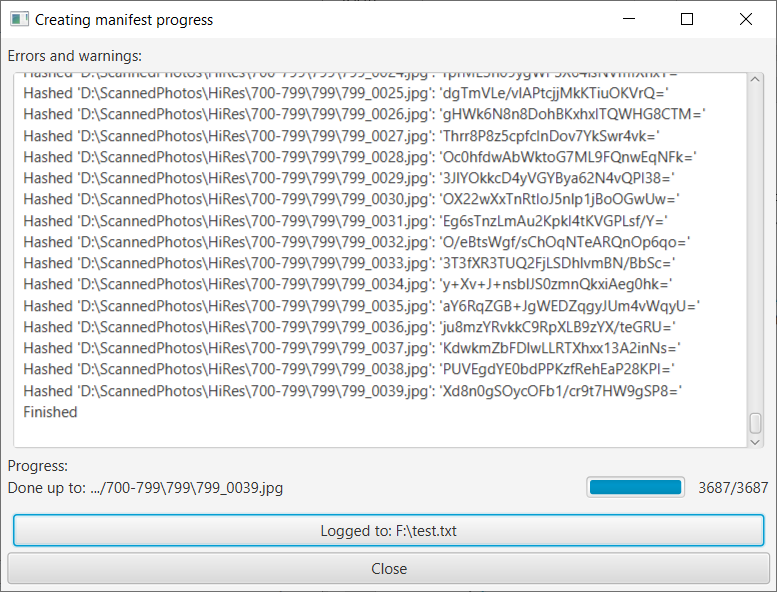
The text in the Status panel will change to ‘Stage 2 processing – no errors or warnings generated yet’. This text will be removed if any warnings or errors are generated. Underneath the status panel, there are three pieces of progress information. On the right is the current count of the number of files processed out of the total to be processed (in this case the program is processing file 1432 of 9800 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.

Note that the error and status files are being logged to the file specified in the set-up window.

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

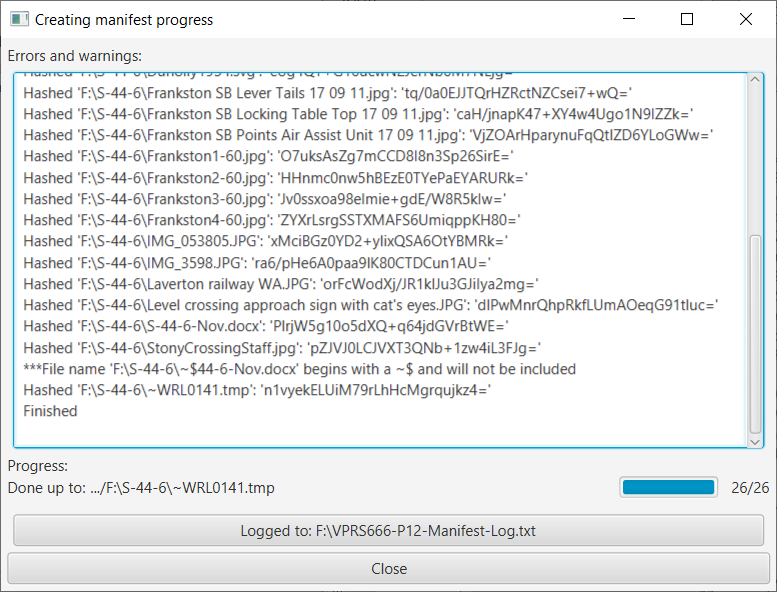
If a log file had not been selected on the set-up screen, the label on the logging button will change to ‘Save the errors and warnings log’:

If the button is pressed at this point, a file select window will pop-up and allow a log file to be selected. All of the status messages will be captured in the log file (even those generated before a log file was selected). This can be done at any time until this window is closed, even after the generation of the manifest has been completed. Note that the log file is not generated until the processing is completed.

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

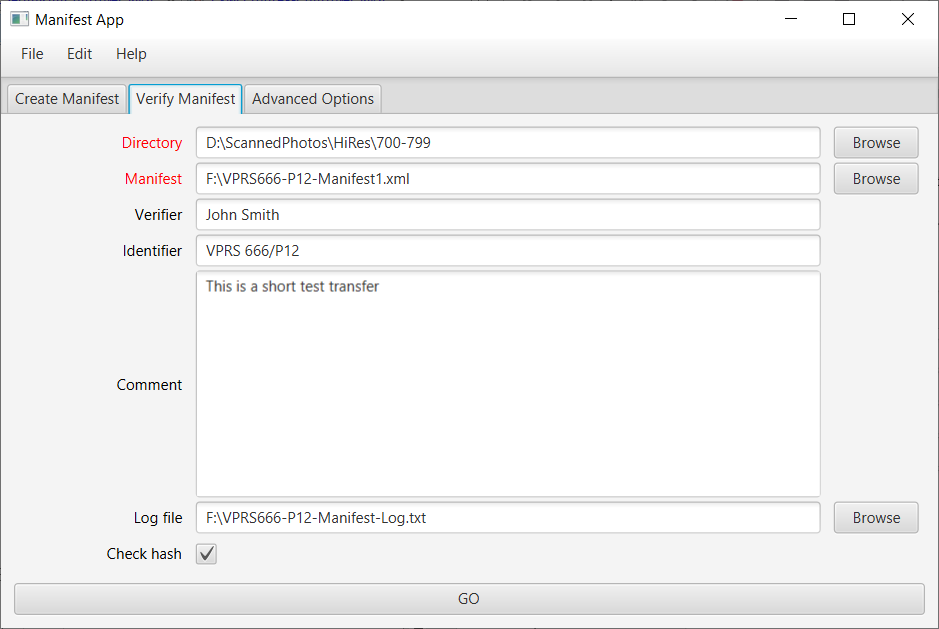
The text ‘Finished’ has been added to the status panel; the count shows that all files (‘19406/19406’) have been processed, and the progress bar is at 100%.

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

The following screen shot shows the output when the ‘verbose’ flag on the Advanced Options tab has been selected:

This screen shot has been captured when the manifest creation has completed. Note the ‘Finished’ on the ‘Errors and Warnings’ pane, the completed progress bar (26 of 26 files finished), that the status messages have been logged to the log file (rather than ‘loggin’) and the finish button has changed to ‘Close’ instead of ‘Cancel processing or close’.

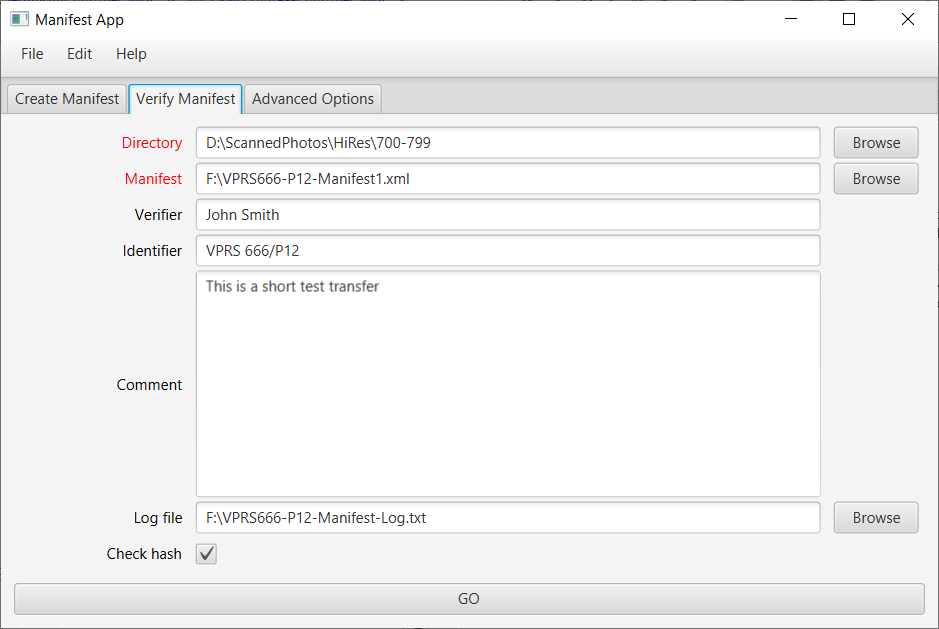
## Verifying a Manifest

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.

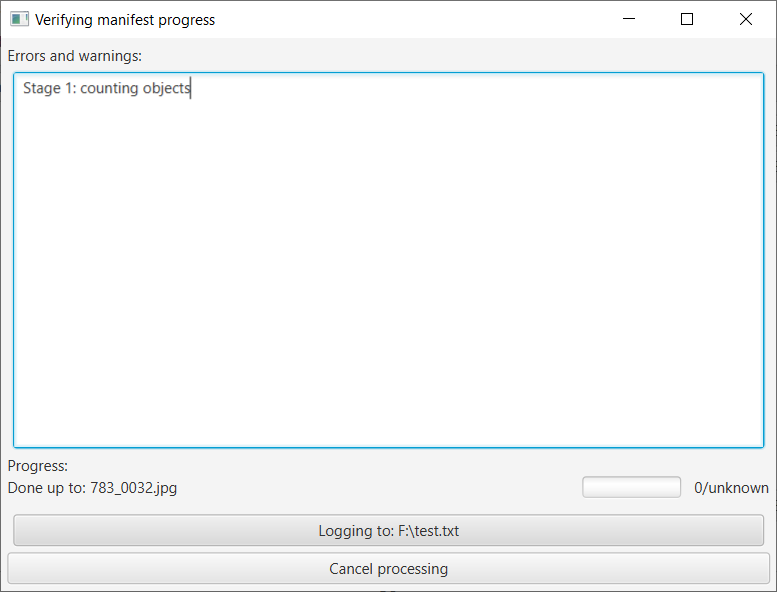
There are six fields that can be entered and one check box. The mandatory fields (Directory & Manifest) must contain data, and have red field names.

The fields & checkbox are:

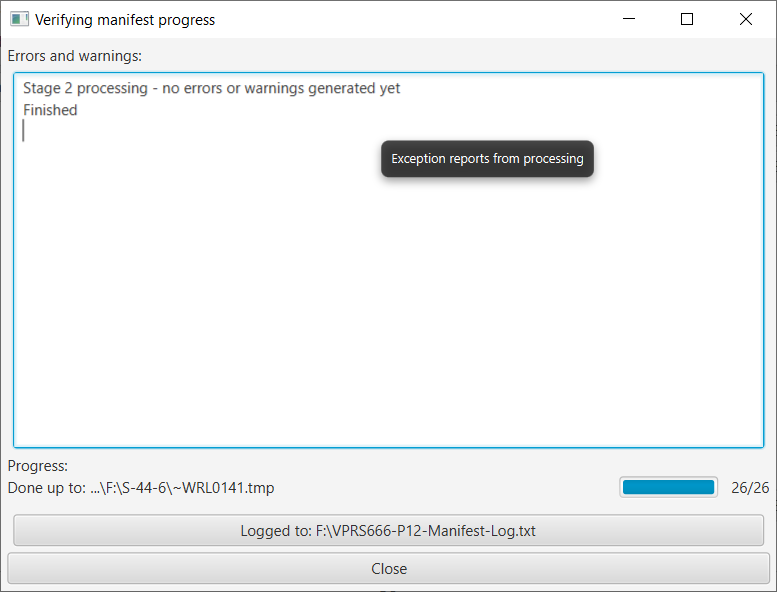
* Directory (Mandatory). The computer directory which contains the digital objects to be verified against the manifest. This can be entered as free text, or the ‘Browse’ button on the right hand side of the text field can be pressed to navigate to the directory.
* Manifest (Mandatory). The name and location 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.
* Log file (Optional). This field contains the name of a file where the status messages are to be saved
* Check hash (Optional – by default ticked). If this tickbox is unticked, the hash values are NOT verified. Only the presence of the file is checked. This, of course, is slightly faster.

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.

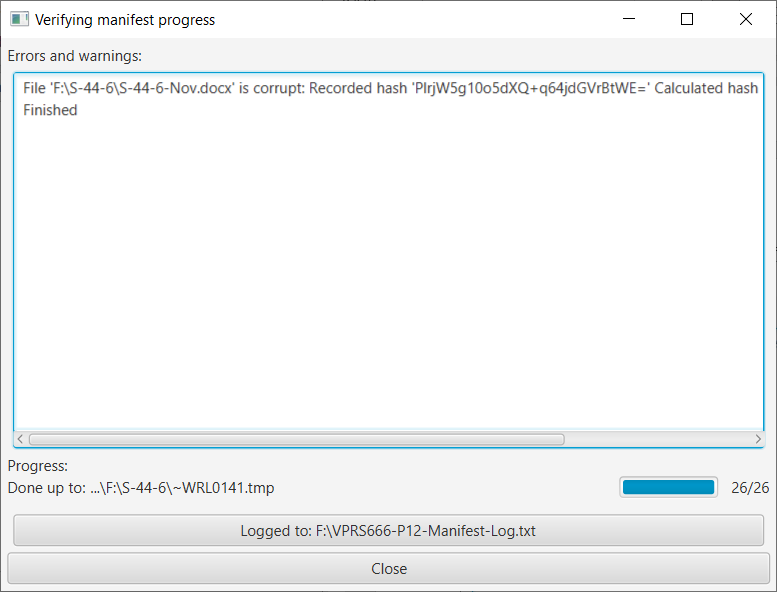
Pressing the ‘GO’ button starts the verfication of the manifest. This pops up the ‘Verifying manifest progress’ window:

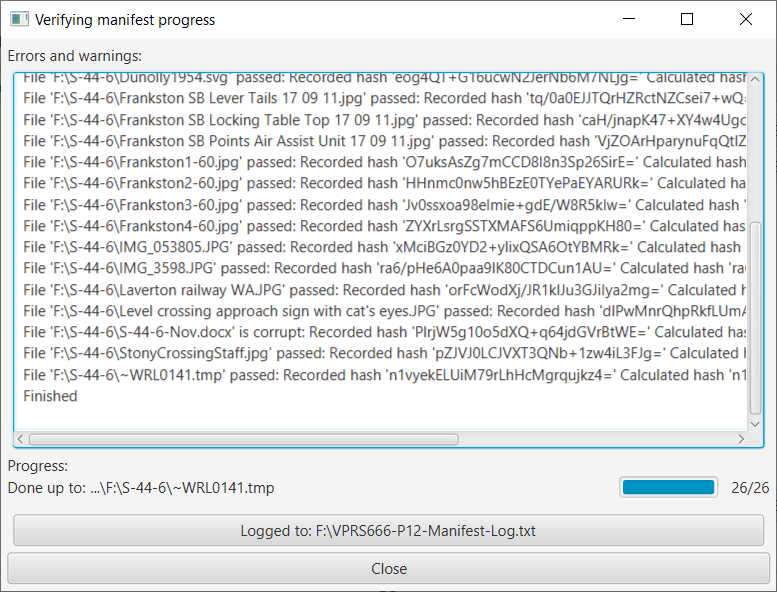
The user interface of the verification stage is exactly the same as for creating a manifest. The first stage is to count the number of objects to be verified, then the identified files are verified against the hash in the manifest. As this happens, messages are displayed in the ‘Errors and warnings’ section. If the ‘verbose’ flag is ticked on the ‘Advanced Options’ tab more detail is displayed, including listing every object that is verified. If the ‘verbose’ flag is NOT ticked only warning and errors are displayed. The messages can be logged to a file as well as being displayed. The progress of the verification is displayed in two ways: showing the current file being processed, and by a progress bar.

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

The text ‘Finished’ has been added to the status panel; the count shows that all files (‘26/26’) have been processed, and the progress bar is at 100%.

If, however, some of the files do not verify, details of the files will be displayed in the status window (with the verbose flag NOT ticked):

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

If the verbose flag is set, details of all files checked will be reported: 

# Creating a manifest using the command line interface

Instead of the GUI, it is possible to call the Manifest tool from the command line. It is recommended that the standard Manifest.bat file be used to easily set the environment of the tool correctly.

## Command line arguments

The Manifest program can be used in one of three modes, controlled by the following command line arguments:

* **-o <ManifestFile> <directory> create (output) the specified manifest file from the specified directory.**
* **-i <ManifestFile> <directory>** check (input) the specified manifest file against the specified directory.

A minimal example of usage to generate a manifest

Manifest -o f:/VPRS421-P10-Manifest.txt f:/VPRS421-P10

And to verify against the manifes

Manifest -i f:/VPRS421-P10-Manifest.txt f:/VPRS421-P10

The following command line arguments are optional:

* **-l <file>** copy the output to a log file
* **-v** verbose mode. Include details about the run, and the results of processing each file. If this mode is not set, only errors are reported
* **-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'.
* **-help** print a summary of the command line arguments.

## 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