Version number: 0.1

Issue Date: 8 February 2023

# Introduction

## Purpose

The purpose of this procedure is to outline how to use Track Transfer toolset to track objects (normally VEOs) received in transfers of digital records from agencies

## Background

Public Record Office Victoria (PROV) receives transfers of digital records from agencies. These transfers can contain thousands of records received in multiple tranches. Records can be received multiple times (either due to error, or because later instances can replace earlier ones). At the end of the transfer PROV needs to be able to reconcile all of the objects received with those whose custody has been accepted.

This tool set allows the documentation of digital records and generation of reports on the transfer.

## Audience

This procedure has been developed for:

* Transfer archivists in PROV

# Track Transfer toolset

## What is the Track Transfer toolset?

The Track Transfer toolset is a set of tools that allow the tracking, reporting, and reconciliation of digital objects received in a transfer.

## What tools are in Track Transfer toolset?

The Track Transfer toolset contains the following tools:

* New transfer. Run once when a new transfer is started
* New delivery. Run when a new delivery of records has been received to register the objects in the delivery
* Annotate. Run whenever it is necessary to record a comment on objects in the delivery
* Report. Produce a report about the objects, their status, and/or the events that occurred to them.
* Help. A simple summary of available commands and options.

## How are the Track Transfer tools used?

The Track Transfer tools 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.

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

A bat file (TT.bat) is provided to simplify the use of the tools.

## 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.

# Track Transfer tool set

## Business rules of a Transfer

To understand the tools and how they are intended to work, it is necessary to understand the conceptual structure of a transfer on which the tools are based.

A Transfer consists of (many, many) Items. Some of these Items are eventually accepted into custody as records, others are discarded. Currently Items are identified by their filename – so two files named ‘VEO-12234.veo.zip’ are considered to be the same Item.

Items are delivered to PROV in Deliveries. A Transfer must have one Delivery, but may have several Deliveries over time. Second and later Deliveries might contain additional or replacement Items. They might also contain duplicate Items.

Because second and later Deliveries may contain Items that are already known about (i.e. replacements or duplicates of files with filenames that have already been seen), we consider Deliveries to contain Instances of Items. In the best of all possible worlds, each Item would be represented by one Instance (received in a Delivery). In the real world, an Item might have multiple Instances. Extremely badly structured Deliveries may even contain duplicate Instances of one Item (i.e. the Delivery contains files with the same file name).

At any given time, each Item has an Active Instance. This Instance is the last received (or encountered in a Delivery). The Active Instance is the Instance being currently worked on. NOTE: This means that if you get the same file in two deliveries, the second file is considered to supersede the first and will be the one being worked on.

An exception to this rule that the final instance is the active instance is if an Instance of an Item is received after the Item has been finalised (i.e. custody has been accepted, or the Item has been judged to be abandoned). In this case, the Instance marked as ‘too late’ and is ignored.

It is possible to annotate Items by either setting a status on the Item, or by associating a comment with the active Instance of the Item. Ultimately it will be possible to annotate an Item by using output from processes (e.g. the status report from the Ingest process, or the VEO validation tools).

## Track transfer commands (general)

There is only one program, the ‘TrackTransfer’ program. All the Track Transfer commands are invoked by specifying them as the first command line argument (e.g. “TT newTransfer…”). The commands are

* NewTransfer: Create a new transfer
* NewDelivery: Create a new delivery within a transfer
* Annotate: Annotate a collection of items within a transfer. The collection is specified by files in a directory or by filenames in a file
* Report: Generate a report
* Help: Print a list of the Track Transfer commands.
* DropDatabase: Delete the database. WARNING. This deletes all information about a Transfer, including deliveries, items, instances and event.
* Input: Read a series of Track Transfer commands from a file (intended for testing).

Note that these commands are not prefixed by a ‘-‘ symbol.

## Register a new transfer

This command is used to start the processing of a new transfer. It can only be executed once for each transfer.

The expected model of use is that the transfer archivist will create a transfer directory in which they will put everything related to the transfer. The track transfer database should go in this transfer directory, and there should be only one transfer database in each transfer directory.

The command line arguments are:

* -db <databaseName>: (Mandatory) The name of the database to create. You can use any alpha-numeric string, but avoid punctuation and spaces. You cannot create two databases with the same name in the same directory.
* -desc <description>: (Mandatory) A short description of this transfer (truncated to 100 characters). The string must be enclosed in double quotes.
* -help: (Optional) Print a summary of the command line arguments for newTransfer.

An example of registering a new transfer is:

TT newTransfer -db TR2023-1 -desc “DHS Ward Cards”

Requesting help about the newTransfer command:

TT newTransfer -help

## Register a new delivery

This command is used to process the files received in a new delivery of records received from an agency.

The command line arguments are:

* -db <databaseName>: (Optional) The name of the transfer database to associate this delivery with. The <databaseName> is the same as the <databaseName> you specified in the registerTransfer command. This command line argument is optional. If the current working directory (i.e. the transfer directory) contains just one database, this is assumed to be the desired database.
* -dir <directoryPath>: (Mandatory) The pathname of the directory containing the files received in this delivery.
* -desc <description> (Mandatory) A short description of this delivery (truncated to 100 characters). The string must be enclosed in double quotes.
* -veo: (Optional) Only process VEOs (files that end in ‘.veo’ or ‘.veo.zip’.
* -help: (Optional) Print a summary of the command line arguments for newDelivery.

An example of registering a new delivery is:

TT newDelivery -db TR2023-1 -desc “Test delivery” -dir ./Delivery-20230209

If there is only one transfer database in the current working directory you can omit the -db:

TT newDelivery -desc “Test delivery” -dir ./Delivery-20230209

If you want to only process the VEOs in the directory:

TT newDelivery -desc “Test delivery” -dir ./Delivery-20230209 -veo

Requesting help:

TT newDelivery -help

## Annotate some items

This command is the heart of Track Transfer, it annotates items. Two types of annotations are supported: keywords; and descriptions.

A description is a short piece of text. Descriptions are recorded as events against the Items, and eventually appear in the event history of the item.

Keywords can be set or removed from Items. It is possible to run reports listing the Items that have specific keywords set. Adding or removing keywords is also recorded as an event.

Two keywords (‘Custody-accepted’ and ‘Abandoned’) manipulate the state of an Item. All Items are in one of three states: Processing, Custody-accepted, or Abandoned. All Items start with a state of Processing. If an Item is moved into the states of Custody-accepted or Abandoned, the item is locked. No further annotations are allowed, and if a new instance of the item is received in another delivery, the event is marked as ‘Too Late’. Setting the ‘Custody-accepted’ keyword will always move an Item into the Custody-accepted state (from either Processing or Abandoned). Setting the ‘Abandoned’ keyword moves an Item in Processing to Abandoned (NOTE: this will not move an Item in Custody-accepted to Abandoned). Removing the ‘Custody-accepted’ or ‘Abandoned’ keywords will move the Item back to processing (while it is not good practice to move an Item from Custody-accepted back to Processing, this is allowed to deal with mistakes).

It will work if the directory contains ‘short-cuts’ to the real files.

The basic command line arguments are:

* -db <databaseName>: (Optional) The name of the transfer database to associate this delivery with. The <databaseName> is the same as the <databaseName> you specified in the registerTransfer command. This command line argument is optional. If the current working directory contains just one database, this is assumed to be the desired database.
* -desc <description> (Conditional) A short description of this delivery (truncated to 100 characters). The string must be enclosed in double quotes. The description becomes an event in the history of the item. At least one of -desc, -set, & -remove must be present.
* -set <keyword> (Conditional) Set a keyword on the selected Items. It is not an error to set the same keyword multiple times on an Item. If you set and remove the same keyword from an Item in the one command, the last set or remove is effective. The keyword is free text and can consist of upper and lower case characters, digits, and punctuation. At least one of -desc, -set, & -remove must be present.
* -remove <keyword> (Conditional) Remove a keyword from the selected Items. It is not an error to remove a keyword that is not actually set. . If you set and remove the same keyword from an Item in the one command, the last set or remove is effective. The keyword is free text and can consist of upper and lower case characters, digits, and punctuation. At least one of -desc, -set, & -remove must be present.
* -veo: (Optional) Only process VEOs (files that end in ‘.veo’ or ‘.veo.zip’.
* -help: (Optional) Print a summary of the command line arguments for annotate.

One of two methods can used to select Items: either by annotating Items found in a directory (including subdirectories), or by getting the Item names from a CSV or TSV file.

If the Items are to be identified by files in a directory, the following command line option is available:

* -dir <directoryPath>: (Conditional) The pathname of the directory containing the files received in this delivery.

If the Items are to be identified by names extracted from a CSV or TSV file, the following command line options are available

* -file <filePath>: (Conditional) The pathname of the CSV or TSV file. Note that if the file extension is ‘.csv’, the file is assumed to be a CSV file, and if it is ‘.tsv’ it is assumed to be a TSV file. This behaviour can be overridden by the -tsv and -csv commands.
* -csv (Optional) Force processing of the file as a CSV file, irrespective of the file extension.
* -tsv (Optional) Force processing of the file as a TSV file, irrespective of the file extension.
* -itemcol <number>: (Mandatory if processing a CSV or TSV file) The column in the file in which the Item name will be found (columns in CSV files are separated by commas, in TSV files by tabs). The first column is column 0.
* -skip <number>: (Optional) Skip the first number lines in the CSV or TSV file (assumed to be header lines).
* -pattern <test>: (Optional) Select specific Items (lines) in the file depending on the pattern. Unselected lines will not be annotated. The test is a sequence of patterns separated by commas. Each pattern is of the form <column>’=’<value>. The column is the column number (the first column is 0). The value is a Java regular expression. All of the patterns must be satisfied for the line to be selected. If -pattern is not specified, all lines in the CSV or TSV file are selected.

In the following example, files that are malformed have been collected in the ‘Malformed’ subdirectory and a description annotation applied:

TT annotate -db TR2023-1 -desc “Not valid PDFs” -dir ./Malformed

Again, the -db can be omitted if there is only one database in the current directory:

TT annotate -desc “Not valid PDFs” -dir ./Malformed

If instead of adding a description, you want to set a keyword:

TT annotate -set “Invalid-character-bug” -dir ./Malformed

You can set multiple keywords:

TT annotate -set “Invalid-character-bug” -set “Invalid-PDF” -dir ./Malformed

And you can set/remove a keyword and a description:

TT annotate -remove “Invalid-character-bug” -desc “Not actually invalid – see report XYZ” -dir ./Malformed

Alternatively, this annotates the Items identified by the 5th column in a CSV file (remember first column is column 0):

TT annotate -desc “Not valid PDFs” -file results.csv -itemcol 5

If you want to skip the first line as it is a header:

TT annotate -desc “Not valid PDFs” -file results.csv -itemcol 5 -skip 1

If you want to process the result file of a digital archive ingest, setting the all the VEOs that had been ingested correctly:

annotate -in testAnnotate.csv -csv -skip 2 -pattern 4="VEO Passed",5="DAS Passed",6="AMS updated" -itemcol 8 -desc "Ingest Run 1 - Custody Accepted" -set Custody-accepted -v

Note that the pattern tests that the 5th column is “VEO Passed”, the 6th column is “DAS Passed”, and the 7th column is “AMS updated”. Only those items that match this pattern are updated.

You could reprocess the same file and annotate those files for which the VEO failed validation:

annotate -in testAnnotate.csv -csv -skip 2 -pattern 4="VEO failed" -itemcol 8 -desc "Ingest Run 1 - Failed VEO validation"

It is possible to set the status of an item. If the status is set to ‘Custody-Accepted’ no further updates to the Item can be performed:

TT annotate -status Custody-Accepted -dir ./Finished

The other special status is ‘Abandoned’:

TT annotate -status Abandoned -dir ./Junk

It is possible to specify both a status and a description:

TT annotate -status Abandoned -desc “Not permanent records” -dir ./Junk

Requesting help:

TT annotate -help

## Report on items

This command prints out a report from the database. Five reports are available: complete, keyword, custody-accepted, abandoned, and incomplete.

The command line arguments are:

* -db <databaseName>: (Optional) The name of the transfer database to associate this delivery with. The <databaseName> is the same as the <databaseName> you specified in the registerTransfer command. This command line argument is optional. If the current working directory contains just one database, this is assumed to be the desired database.
* -keyword <keyword>: (Optional). A report listing those items which have the specified keyword set. This command can be repeated; an Item is included in the report if it has all the keywords set.
* -complete: (Optional). A complete report listing each item known. For each item, each instance received is listed and each event that occurred to the item.
* -custody-accepted: (Optional). A report listing every item that has a status of ‘Custody-Accepted’ (case insensitive).
* -abandoned: (Optional). A report listing every item that has a status of ‘Abandoned’ (case insensitive).
* -o <filename>: (Optional) The file to contain the report.
* -help: (Optional) Print a summary of the command line arguments for annotate.

In the following example, files that are malformed have been collected in the ‘Malformed’ subdirectory:

TT report -db TR2023-1 -complete -o ./Full-reconciliation.txt

Again, the -db can be omitted if there is only one database in the current directory:

TT report -complete -o ./Full-reconciliation.txt

List the Items that have the keyword ‘Invalid-PDF’ set:

TT report -keyword Invalid-PDF -o ./Full-reconciliation.txt

List the Items that have the keywords ‘Invalid-PDF’ and ‘Invalid-character’ set:

TT report -keyword Invalid-PDF -keyword Invalid-character -o ./Full-reconciliation.txt

Requesting help:

TT report -help

## Drop database

This command deletes a transfer database and all the information contained in it.

The command line arguments are:

* -db <databaseName>: (Mandatory) The name of the transfer database to associate this delivery with. The <databaseName> is the same as the <databaseName> you specified in the registerTransfer command. Unlike the other commands, the -db argument is mandatory in DropDatabase to prevent accidents.
* -help: (Optional) Print a summary of the command line arguments for annotate.

In the following example, files that are malformed have been collected in the ‘Malformed’ subdirectory:

TT dropdatabase -db TR2023-1

## Input

This command a series of track transfer commands from a text file, one per line, and executes them. The command is primarily intended for testing, but could be used for other purposes.

The command line arguments are:

* <file.txt>: (Mandatory) The file containing the Track Transfer commands. Each command is on a separate line, and is not prefixed by ‘TT’ or ‘TrackTransfer’. Blank lines and lines that start with ‘!’ are ignored.

In the following example, files that are malformed have been collected in the ‘Malformed’ subdirectory:

TT input test.txt

End of procedure