# AssetX

## Table of contents

Asset·X ... 5
  Installation ... 5
  Introduction ... 6
  What's New in Version 3.2 ... 8
Asset List ... 10
File List ... 13
Outline ... 14
  Outline Context Menu & Toolbars ... 17
  Creating and Editing Tags ... 19
  Replication ... 23
Viewer Pane ... 25
  Text Editor ... 25
  Image Editor ... 26
    Supported File Formats ... 26
    Viewer Controls ... 27
    Imaging Commands ... 28
    Alpha from Image ... 30
    Import Alpha from File ... 31
    Resizing Images ... 32
    Make Non Uniform ... 33
  Mesh Viewer ... 33
  Texture File Editor ... 36
  LOD Mesh File Editor ... 38
  Grid Viewer ... 40
  Audio Editor ... 41
Main Menu Reference ... 42
  File Menu ... 42
  Asset Menu ... 43
  Trainz-Build Menu ... 45
  Diagnostics Menu ... 45
  Repair Menu ... 46
  Scripts Menu ... 47
  Tools Menu ... 49
  Options & Help Menus ... 50
    Preferences ... 51
    Paths & Files ... 53
    User Data ... 55
    Tool Definition ... 55
    Tool Examples ... 59
Backups ... 61
Scripts ... 62
  Script Files ... 64
  Parameters & Data ... 65
  Operators ... 65
  File Searches ... 68
  Inheritance ... 68
  Script Variables ... 69

Batch Mode Scripts ... 71
Run Once Scripts ... 72
Script Command Listing ... 73
-BackupAsset ... 77
-BackupFile ... 78
-Break ... 78
-Category ... 79
-CallScript ... 79
-Case ... 80
-Clear ... 81
-Comment ... 83
-Combine ... 83
-Compute ... 84
-Condition ... 85
-CopyFile ... 86
-CopyFromAsset ... 86
-Delete ... 87
-DeleteFiles ... 87
-EndCase ... 88
-EndFor ... 88
-EndSelect ... 88
-Exchange ... 88
-Exit ... 89
-Expand ... 89
-FixKUIDs ... 90
-ForEach ... 90
-If -IfNot -Else -EndIf ... 92
-IfAsset ... 93
-IfMatches ... 94
-IfQuery ... 94
-IfThumbnail ... 96
-IgnoreCase ... 97
-Images2TGA ... 97
-IncrementKUID ... 97
-Info ... 98
-INIRead ... 98
-InstallTool ... 99
-MakeBogeyTable ... 99
-MakeMeshTable ... 100
-MakeNormalMap ... 101
-MakeTGA ... 101
-MakeThumbnailTable ... 102
-MessageBox ... 102
-MenuExecute ... 102
-PointRead ... 104
-PointPosition ... 104
-PointRotation ... 105
-PM2IM ... 106
-Promote ... 106
-Rename ... 107

-RepairTextures ........ 107
-RepairWaves ........ 109
-Replace ........ 110
-ReplaceAll ........ 110
-Restore ........ 111
-Save ........ 111
-SaveConfig ........ 112
-Select ........ 112
-SetEnv ........ 114
-Silent ........ 114
-Sort ........ 115
-Store ........ 116
-TagForDeletion ........ 116
-Tokens ........ 117
-Update ........ 117
-UpdateAssetFilename ........ 118
-UpdateCamera ........ 119
-UpdateCategoryEra ........ 119
-UpdateCategoryRegion ........ 119
-UpdateKUIDTable ........ 120
-UpdateUserData ........ 120
-UpdateUsername ........ 121
-Write ........ 121
Asset·X Macros ........ 122
Keyboard Shortcuts ........ 123

Next

Asset·X Help File, 22 January 2017
2:09:59 PM

**Asset·X, Cleaning up the parts that other software can't reach.**

Welcome to Asset·X created by the late Andi Smith (Andi06) and Peter Villaume (PEV).

Asset·X is a multi-purpose tool box for maintaining, upgrading and creating Trainz assets. Some of the facilities provided require an interface with external tools and the mechanism for implementing this has been made as flexible as possible to allow the software to be customised and extended to the greatest possible extent.

Our intention has been to empower the user by providing viewers and editors for almost every type of file used in Trainz Assets.

Though this program has backup facilities, keep in mind that the software may have faults that could corrupt your Trainz Data. Please ensure that any testing or work done is backed up before making any changes with this software.

PEV is not liable in any way for any data corruption or loss caused by the use or mis-use of Asset·X. This program was released by Andi06 and PEV in good faith to assist the Trainz User Community.

The programme is now a mature product that will be updated when necessary by PEV. For the latest changes or additions, refer to the Release Notes, which can be accessed via the main menu Help section.

Thanks to Ian Woodmore for his invaluable assistance in the on-going development and testing this software and to Paul Cass for initiating the help link system and carrying out most of the tedious work required to implement it.

**Version 3.2, January 2017.**

In memory of Andi Smith.

## Installation

Previous Top Next

To install and configure Asset·X run the provided installer.  The default installation path can be changed as required.

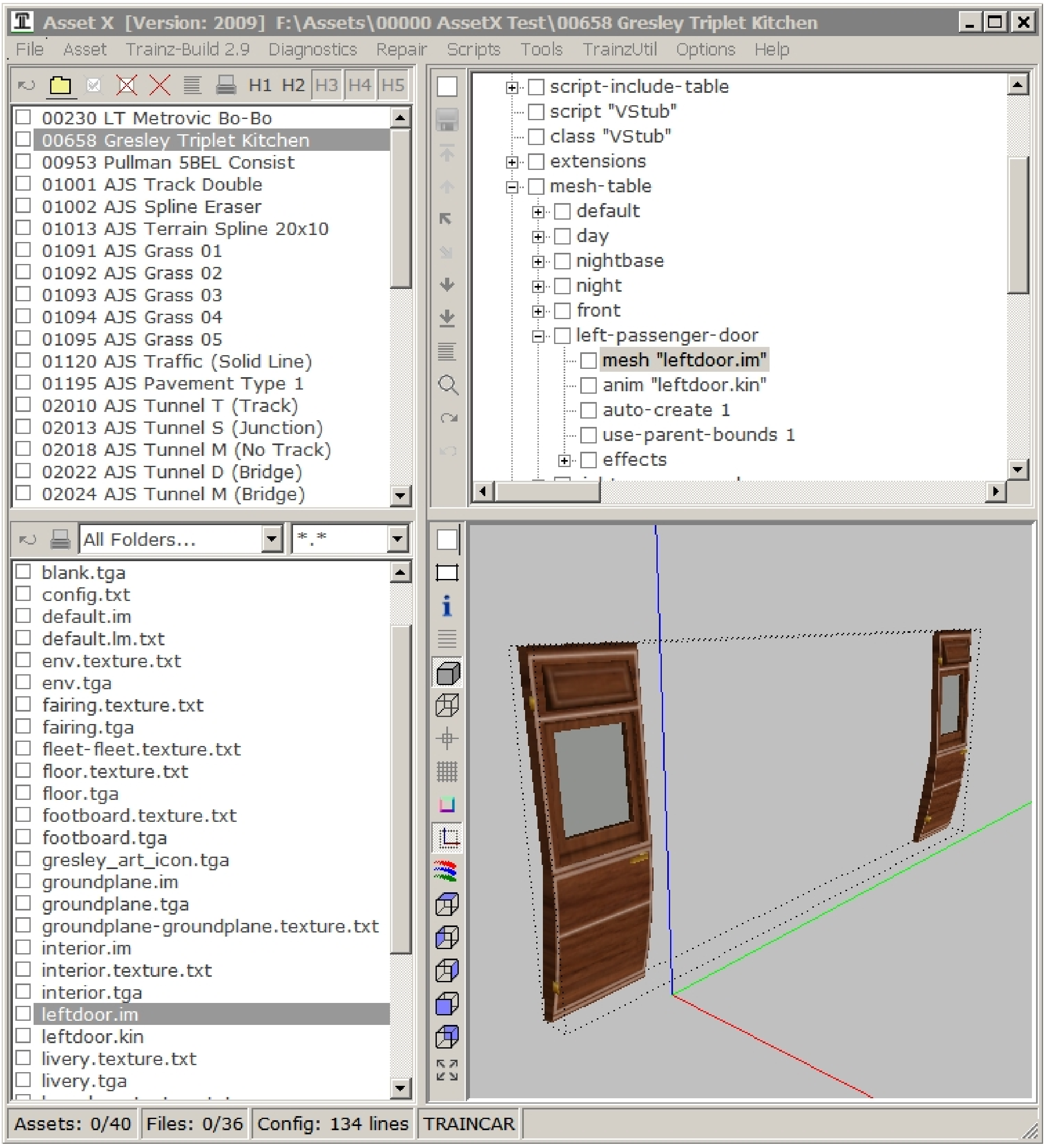
If you are running Vista, Windows, Windows 8 or Windows 10 you may need to install and/or run the software with administrative privileges.

The same installer can be used for a new installation or to update an existing version.

After installation you should run the programme, read the welcome screen, select *Options/ Preferences* from the main menu and make sure that the various settings, particularly the path to your TRS installation and home folder, are entered correctly.

## Introduction

The main screen in Asset·X consists of a main menu bar and four resizable panes:



Asset X [Version: 2009] F:\Assets\00000 AssetX Test\00658 Gresley Triplet Kitchen

```
00230 LT Metrovic Bo-Bo
00658 Gresley Triplet Kitchen
00953 Pullman 5BEL Consist
01001 AJS Track Double
01002 AJS Spline Eraser
01013 AJS Terrain Spline 20x10
01091 AJS Grass 01
01092 AJS Grass 02
01093 AJS Grass 03
01094 AJS Grass 04
01095 AJS Grass 05
01120 AJS Traffic (Solid Line)
01195 AJS Pavement Type 1
02010 AJS Tunnel T (Track)
02013 AJS Tunnel S (Junction)
02018 AJS Tunnel M (No Track)
02022 AJS Tunnel D (Bridge)
02024 AJS Tunnel M (Bridge)
```

```
script-include-table
script "VStub"
class "VStub"
extensions
mesh-table
  default
  day
  nightbase
  night
  front
  left-passenger-door
    mesh "leftdoor.im"
    anim "leftdoor.kin"
    auto-create 1
    use-parent-bounds 1
    effects
```

All Folders... *.*

```
blank.tga
config.txt
default.im
default.lm.txt
env.texture.txt
env.tga
fairing.texture.txt
fairing.tga
fleet-fleet.texture.txt
floor.texture.txt
floor.tga
footboard.texture.txt
footboard.tga
gresley_art_icon.tga
groundplane.im
groundplane.tga
groundplane-groundplane.texture.txt
interior.im
interior.texture.txt
interior.tga
leftdoor.im
leftdoor.kin
livery.texture.txt
livery.tga
```

Assets: 0/40 | Files: 0/36 | Config: 134 lines | TRAINCAR

**Header Caption**
The application header records the version and build of TRS which Asset·X is currently using together with the path of the current asset.

**Main Menu**
The main menu contains commands and options which are relevant to the current state of the application. Each of the viewers also has a context sensitive popup menu which can be accessed via the right mouse button.

**Asset List:**
The Asset List (top left) contains a list of the assets you are currently working with, the highlighted item represents the currently selected asset. Assets can be loaded from any location on disc including the *...\Editing* folders of the various versions of TRS.

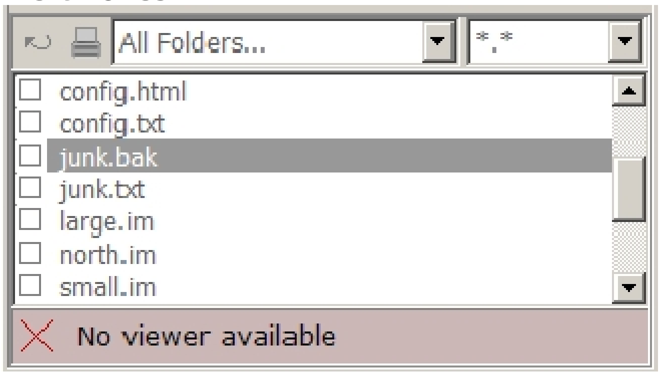
**File List:**
The File List (bottom left) contains a directory listing of all the files in the current asset

folder and any sub folders. The list can be filtered by file type and/or by sub folder using the drop down lists on the toolbar.

**Outline:**
The Outline or Configuration Pane (top right) contains the contents of the current *config.txt* as a hierarchical tree. Whilst this differs from the standard format it has the same meaning, however it allows a focused view of the various configuration items and it frees the user from worrying about the syntax required by the *config.txt* format. The Outline offers extensive facilities for editing and copying configurations and can be maximised to occupy the entire right hand section of the application.

**Viewer Pane:**
The Viewer pane (bottom right) displays information related to the currently selected item from the configuration or file panes (whichever was last selected). If the selected item is a text file the contents of the file will be displayed in a simple text editor, for images, the image itself will be displayed. There are other viewers and editors to display various types of information. Like the Outline, the viewer pane can be maximised.

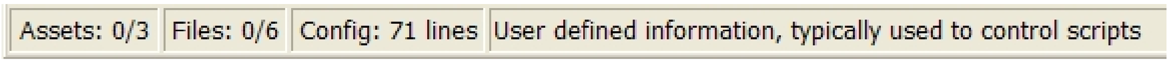
**Alert Boxes**



No viewer available

Each of the four display panes contains an alert panel which is used to report messages back to the user without interrupting any action which is under way. In some cases the alert box is also used to report the results of a command. Alerts can be closed by clicking on the red cross.

**Status Bar**



Assets: 0/3 | Files: 0/6 | Config: 71 lines | User defined information, typically used to control scripts

The status bar at the foot of the screen keeps a count of the number of tagged assets and of the total number of assets which are currently loaded together with a count of files in the current asset, the size of the current *config.txt* and other information relevant to the state of the application, such as help strings.

## What's New in Version 3.2

Previous Top Next

Version 3.2 is a minor update which includes. Refer Release Notes for more detail.

**Paths and Files Options**
External script editor option added
ConTEXT editor highlighters added

**New Script Routines**
-IfThumbnail
-MenuExecute

**Mesh Viewer**

support for TANE SP1 textures

**Image Viewer**
support for TANE SP1 textures

---

Version 3.1 was a minor update included the following:-

**Mesh Viewer**
support for TANE textures

**Image Viewer**
support for TANE textures

**Scripting**
Variables 0..9 in string form have no length limit

Note: This version was mistakenly released by PEV as Version 3.0.2

---

Version 3.0 was a major update that included many new and improved features, some of which are listed below.

**Asset List**
Home Folder Filtering & Conditional Installation
Automatic Refresh
Alternative Sort Ordering
Revert and Commit assets open in CMP

**Outline**
Find and Replace Text
Copy and Paste Tags and Containers
Help Links to Wiki
Direct Web Links
Support for Tracks and Splines
Detection of Subsidiary Kinds
Load Dependencies of current asset
Display status of dependencies

**File List**
Automatic Refresh

**Image Editor**
Repair Uniform Colour Textures
Create Normal Maps
Create and edit Thumbnails

**New Audio Editor**
Repair Faulty Audio Files

**Tools**
Scripts can now be configured as custom commands.

**Script Interpreter**
Script Variable Support
Expanded Wildcard Support

Inline Comments

**New Script Routines**
-Case / -EndCase
-Compute
-Foreach / -Break / -EndFor
-IfMatches
-IfNot
-IgnoreCase
-IncrementKUID
-MakeNormalMap
-MakeTGA
-PointRead
-PointPosition
-PointRotation
-Replace
-ReplaceAll
-Select / -EndSelect
-Silent
-Store
-RepairTextures
-RepairWaves
-Write

## Asset List

Previous Top Next

The Asset List pane contains an ordered list of the assets you are currently working with. The search is based on folder name and location and not on the asset kuid, as a result of this it is perfectly possible to have multiple copies of the same kuid loaded into AssetX.

**Adding Assets to the List:**
There are several ways to add assets to the list. In each case the selected folders are scanned and if a file named *config.txt* is found, this will be opened to ensure that the folder represents a valid TRS asset:

- By specifying one or more 'Home Folders' via the Options/Paths & Files dialogue. These folders will be loaded automatically every time that the programme runs.

- By selecting one of the source locations predefined in the Asset section of the main menu. These include the *...\custom* and *...\editing* folders for each version of TRS which is installed on your computer. Keyboard shortcuts are also assigned to these folders.

- By clicking the Add Directory button in the Asset List toolbar. This will call up a Browse for Folder dialogue. You can select any folder from this list, an entire drive if you wish, and it will be scanned for TRS assets. Any assets found will be added to the list. This process can take some time if the folder contains a lot of content. You can interrupt or cancel a scan by pressing *[Esc]*.

- By pasting folders or dragging from Windows Explorer (or any other application which supports Drag and Drop) and releasing the mouse button when inside the Asset List.

- By right clicking a folder in Windows Explorer and selecting *Open with AssetX*. This option is not implemented by default but you can install the necessary handler via the Options dialogue. If AssetX is already running the selected folders will be added to the asset list, if not the application will be started.

- Again assuming that you have enabled the option, you can highlight assets in CMP, right click and select *Open With/AssetX.exe* from the context menu. If AssetX is already open the selected assets will be added to the contents of the list, if not AssetX will be started and the assets you selected in CMP will be placed in the list. AssetX does not currently impose any restrictions on what you can do with folders loaded in this way, but you should take care in editing assets which are open in CMP. Some actions, such as incrementing or changing a kuid, will be rejected by CMP when it attempts to commit an open asset and it would be unwise to create a new asset directly in an editing folder.

**Modified Assets:**
Assets which have been, or may have been, modified are denoted using coloured check boxes.

Unless they have been opened from the TRS editing folder, modified Assets will need to be reinstalled in TRS before you will see any of your changes reflected in game.

**Asset List Actions:**
In carrying out your instructions the list differentiates between clicks on an asset name and clicks on the checkbox to the left of the name.

- Left clicking on an asset's name will select that asset. The file list and the outline will be updated with the asset's details and the asset folder name will be displayed in the application caption.

- Right clicking on an asset's name will call up a context menu offering various options for copying, renaming or deleting assets.

- Left clicking on the checkbox will tag or untag the asset for future action.

- Right clicking on the checkbox will invert the state of all the asset checkboxes. Control + Right click will tag or untag all of the assets in the list (you might need to do this twice to achieve the required result).

- You can also tag and untag assets using the Keyboard.

**Exporting via Drag and Drop**
You can drag asset folders from the Asset list to any other windows application which supports drag and drop. The outcome depends on the destination, if the drop target is CMP the assets will be installed, if it is Windows Explorer the assets will be copied to the selected drive or sub-folder.

Dragging with the left mouse button will drag the selected asset only, holding down the control key whilst dragging will cause all tagged assets (as well as the current selection) to be processed.

To prevent creating multiple copies of the same asset the drag function is disabled for any assets which are in a TRS editing folder.

**Asset List Toolbar**

**Refresh Asset List [^F2]**
Reloads the asset list, including the contents of Home Folders. This is not normally necessary since the lists are self-updating.

**Add Folder [^O]**

Opens a file dialogue allowing source assets to be loaded.

**Remove Tagged or Untagged Assets**
Options to remove tagged or untagged assets from the list (neither of these buttons will remove the current asset) If any of the assets selected for removal have unsaved changes you will be prompted to save or to abandon your edits.

**Remove all Assets**
Clears the entire list (including the selected asset)

**Sort Assets**
Sort the asset list, either by asset name or by asset folder path. The sort order can be amended via Options / Preferences.

**Print**
Prints the current asset list. You can also print a list of tagged assets via the right click context menu.

H1 H2 H3 H4 H5 **Home Folder Filters**
These five buttons control the visibility of assets in each of the first five Home Folders. Depressing the corresponding button will hide the contents of the folder. The buttons themselves can be dragged and dropped into CMP, this will result in the re-installation of any assets which have changed since they were last installed.

**Asset List Context Menu**
Right click in the Asset List to access these commands.

**Clone**
See Asset Menu

**Revert**
Reverts the asset abandoning any changes made. The asset status in CMP will be updated accordingly.

**Commit**
Commits the asset. The asset status in CMP will be updated accordingly.

Revert and Commit will only be available for assets in TRS ..\editing folders and (for the time being) will only be enabled if TADDaemon is running or if CMP is open.

**Open in Driver, Open in Surveyor, Open in Railyard**
These commands will only be available for sessions and traincars which have been opened from the editing folder of the current version of TRS.

**Cut, Copy, Paste, Delete, Rename**
Carry out standard Windows actions on the selected file.

**Copy Tagged, Delete Tagged**
Perform similar actions on all tagged files.

**Main Menu Asset Commands**
Additional actions which operate on the Asset List, principally those connected with backing up assets and with loading or reloading specific paths, can be found on the Main Menu *File* and *Asset* sections.

## File List

The File List contains a list of files in the current asset folder, including any sub folders. The two drop down lists at the top of the File List allow the contents to be filtered by their relative path within the asset or by their file type. The Refresh button allows the file list to be manually reloaded.

There is no foolproof way of determining whether or not a folder is a valid trainz asset. This can only be done by looking for the existence of a *config.txt* file and making a cursory check of its contents. If multiple *config.txt* files are found when an asset is opened you will be prompted to verify that the folder is in fact a TRS asset.

Alien filetypes, those that will not be used by the game, are included in the list. These will mostly be source files such as max, gmax, xml, psd or psp files. Some of these can be viewed by Asset·X but they will not actually be exported to TRS when the asset is installed. There will usually be a menu option allowing these files to be opened using their default application.

**File List Actions:**
In carrying out your instructions the list differentiates between clicks on a filename and clicks on the checkbox to the left of the name.

- Left clicking on a filename will select that file. In most cases the file contents will be displayed in the viewer pane.

- Right clicking on an asset's name will call up a context menu offering various options for copying, renaming or deleting assets and for running tools on the selected file.

- Left clicking on the checkbox will tag or untag the file for future action.

- Right clicking on the checkbox will reverse the state of all the file checkboxes. Control + Right click will tag or untag all of the files in the list (you might need to do this twice to achieve the required result).

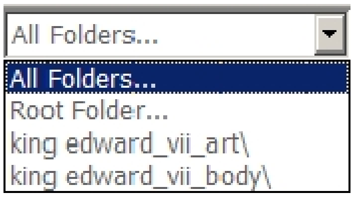
- You can also tag and untag assets using the Keyboard.

**File List Toolbar**

**Refresh**
Manually refreshes the file list.

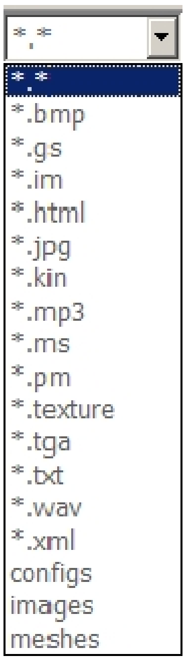
**Print**
Prints the current contents of the File List.

**Folder Filter Box**



Where assets contain sub-folders this filter box allows you to restrict the contents of the File List by folder.

**File Filter**



Allows the File List to be restricted by File Type. If the *Save File Mask* (see Options/ Preferences) option is selected the filter will persist between changes of asset and changes of session.

**File List Menus**

**Open**
If Windows includes an associated application for the selected file this will normally be the first item on the menu. Selecting this command, on a **.doc* file for instance, would normally open it in Microsoft Word.

**Play Audio**
There is an option to play audio files automatically simply by selecting them in the File List. If this option has not been selected then the *Play Audio* command will appear for sound files.

**User Defined Tools**
Commands will appear for tools which have been set up by the user for all files or for particular types of file. See Options/Configure Tools for more details of this feature.

**Cut, Copy, Paste, Delete, Rename**
These commands carry out standard windows actions. Files can also be dragged and dropped into the File List from Windows Explorer.

**Cut Tagged, Copy Tagged, Delete Tagged**
These commands again carry out standard windows actions but will operate on all of the currently tagged files.

## Outline

Previous Top Next

The Outline holds a copy of the currently selected asset configuration in a binary tree format. Containers are represented as branches in the tree which can be collapsed to give an overview of the data or progressively expanded to expose more detail.

**Loading Configs**
The standard config.txt format is converted to the tree view when the asset is opened and converted back when the asset is saved and closed. Asset·X takes care of making sure that the file is properly formatted and has the correct number of brackets in the correct positions. The majority of common formatting errors are accounted for by the loading routine.

Where there is a bracket mismatch in the source file, additional opening brackets are added at the start of the file or closing brackets at the end to restore the balance. Where opening brackets were missing, the first item(s) in the editor will be labelled 'ERROR'. Where closing brackets were missing the last item will still have a valid name but will contain tags and containers that shouldn't actually be there.

In both cases right clicking on the errant container and selecting *Promote All* (possibly more than once) will usually correct the problem quite quickly. Where the file is completely garbled this might not be the case.

**Tags and Containers**
TRS Tags are represented as text strings containing spaces, all of the text before the first space is the tag name and all of the text after that space is the tag value. Quotation marks are normally added automatically where appropriate.

If there are no spaces the text is interpreted as a container. You can collapse containers, or expand them to view their contents, by clicking on the [+] and [-] icons to the left of the text. If nothing happens when you click on a [+] icon it will mean that the container is empty.

**Contextual Information**
Wherever possible selecting an item will display relevant data in the Viewer Pane or Status Bar. If the item is a kuid reference for instance, and the relevant asset is also loaded, then brief details of the dependency will be displayed in the Status Bar. If the item is a text file it will be opened for edit, if an image or mesh then it will be displayed.

**Sorting the Outline**
The whole outline, or parts of it, can be sorted alphabetically by selecting *Alpha Sort Contents* from the right click menu. You can also define a customised sort order by creating a text file named *SortOrder.txt* and placing it in the *...\AssetX\bin* folder.

*SortOrder.txt* contains a list of tag names with an optional control character inserted anywhere in the list. Tags placed above the control character will be moved to the top of the tree, tags below it will be moved to the bottom.

Two control characters are supported:

| Control Character | Effect |
|---|---|
| @ | defines the position at which the list is split, items not specified in *SortOrder.txt* will be left in their existing positions within the file. |
| > | causes the first level of the Outline to be alphabetically sorted before the custom ordering is executed. |

If no control character is used the tags in config.txt are moved to the top following the order defined in the file. The remainder are left unchanged. A sample *SortOrder.txt* is provided as part of the installation.

**Moving Tags**
You can move items (whether they are isolated tags or containers) up and down their own part of the tree by using the arrow buttons in the toolbar. The *Promote* and *Demote* buttons allow you to move tags and sub containers into or out of their existing containers.

**Creating and Editing Tags**
Double click on an item or select it via the popup menu to edit the tag. Similarly right-clicking an item and selecting the appropriate menu item allows you to create new tags or containers.

You can also edit an existing item in the same way that you can edit a file or folder name in Windows Explorer, using a slow double click and overtyping the existing text. The programme does not prevent you from typing whatever you want but it won't allow you to change a tag to a container and it won't allow you to edit container names which are fixed by the config structure.

**Tagging Items**
Items in the tree can be tagged for batch operations. To do this click on the box to the left of the tag name. Clicking with the left button will tag the item and will cascade the tag markers to all of the parents and children of the selected item. Right Click (or Control + Left Click) tags or untags the item without cascading the change in status.

**Tagging for Deletion**
You can mark a tag for deletion by selecting *Tag for Deletion* from the popup menu, marking a container will also mark all of its children.

**Copy and Paste**
You can tag any number of items and copy them to an internal clipboard. The copied items can be pasted back to a different config file or to a different place in the current config. This can result in duplicate tags - it is your own responsibility to correct this, AssetX does not check for duplication.

**Find and Replace**
You can search for text strings within the current config and replace them, one at a time or globally.

**Comments**
You can mark a tag or container as a comment by selecting *Comment* from the context menu, commenting a container also comments all of its children. When an item is commented it will not be exported to config.txt if the Outline is saved but it will be retained in the Outline so that it can be reinstated at any time during the current asset session.

**Modifications**
Newly created items and items which have been modified will be marked in colour.

**Saving the File**
Clicking the *Save config.txt* button exports the current Outline to a standard config.txt file which does not include comments, tags marked for deletion or any other markers.

**Undo System**
The Asset·X Outline includes a system for undoing changes made to the Outline. Most of the built in operations, such as editing tags or running menu commands which modify items in the tree-view, will set an undo marker before modifying the Outline, you can also set a marker manually using the toolbar button or *^K*. Click the Undo button or press *^Z* to return to the previous state, you may have to do this more than once.

Note that Undo will only restore changes to the Outline and will only do this within the current asset session.  The Backup system provides a higher level of security.

The Undo system is disabled during scripts running in batch mode.

## Outline Context Menu

Previous Top Next

**Expand [+]**
Expands a container to view its content.

**Collapse [-]**
Collapses a container.

**Comment**
Marks the selected tag and all of its children as comments.  This prevents the marked tags from being saved to *config.txt*.

**Tag for Deletion**
Marks the selected tag and all of its children for deletion.  This marker is used by the Replication command to mark tags for deletion in the target assets.  Unlike the comment marker, marking a tag for deletion does not prevent it from being saved to file.

**Find & Replace**
Allows you to search for text within the current config and to replace it as required. The routine includes the name of the tag plus its value (separated by a single space character), so you can search for complete or partial data.

The Replace All option refers to text within a single tag.  With this option enabled you could change *kuid <kuid:1234:5678>* to *xxxx <xxxx:1234:5678>* in a single command, with the option turned off the result would be *xxxx <kuid:1234:5678>*

The *Replace* button exchanges the text and then skips to the next occurrence, The *Find* button moves on without changing anything.  Both buttons will continue cycling through the config as long as there is more than one match present.

A history of recent search terms is kept and will maintain its settings throughout a session. This enables you to set up queries in one asset and then execute them on another.

**Help on Tag**
Links to the Auran WIKI page which contains information on the currently selected tag and on its container where appropriate.  The URL is sent to your default browser. The linked page will usually contain links to related areas and should serve as a jumping off point for the tag you are dealing with.  If you don't have an internet connection, or if you don't want to use the system, it can be disabled via a checkbox on the Options/Preferences page (this is enabled by default).

**Asset Details**
This option will be available for a dependency if TADDaemon is running or if CMP is open. Selecting this item will return data on the dependency including the latest locally available version, its username and its current status.

**Open Dependency**
This item is always available when the selected tag contains a kuid reference which is currently loaded into AssetX.  Selecting it will open the referenced kuid.  This enables you to work on an asset and to have its dependencies available at the same time.

In addition if TADDaemon is running, or if CMP is open, you will also be able to open the latest locally installed version of a dependency.

**Increment KUID**
If you are the creator of the current asset, this command increments the KUID2 revision level by 1, there is no point in doing this if the asset is open in CMP, since CMP will reject any change to the kuid tag.

**Promote Contents**
Moves all of the children from a selected container to the next higher level of the tree and places them immediately above the original container. This is mainly useful in correcting corrupt configuration files.

**Alpha Sort Contents**
Sort the contents of a selected container or sub container alphabetically. Note that only the current level of the tree is sorted by this command.

**Insert Tag [Insert]**
Insert a new tag or container immediately above the selected item.

**Add Tag to Container**
Add a new tag or sub container to a selected container.

**Edit Tag [^Return]**
Edit an existing tag or container value. Only user defined container names can be changed by this method.

**Copy Node**
Copies the selected tag or container to an internal clipboard.

**Copy Tagged Nodes**
Copies any currently tagged nodes (both tags and containers) to an internal clipboard. To be included the tag must be visible on screen and must be at the same indentation level as the currently selected tag, this is to protect users from unintentionally copying tags in collapsed containers.

**Paste Nodes**
Inserts the contents of the AssetX clipboard above the selected node. You can copy and paste within assets or from one asset to another but you cannot paste to another application using these routines.

You should be aware that copied items will be pasted immediately above the currently selected item. You can use this to copy items into and out of containers.

**Delete [Del]**
Delete the selected item. If the item is a container this will also delete all of its children.

**Delete Tagged [^Del]**
Delete all tagged items.

**Run Script**
If the selected item is an embedded script, this command will execute it on the current asset.

**Outline Buttons**

☐ **Maximise/Restore**
Increases the size of the Outline such that it will occupy the entire height of the application

screen, a second press and restores the default state.

**Save Outline [^S]**
Saves the Outline to *config.txt* without prompting for a filename. If you need to save to a different path or filename, use the menu item *File/Save Outline as...*

**Move to Top, Move to Bottom**
Moves the selected tag or container to the top or the bottom of the tree-view.

**Move Up, Move Down**
Moves the selected tag or container up or down the tree-view.

**Promote**
Moves the selected tag or sub container out of its parent container and places it at the next higher level.

**Demote**
Moves the selected tag or sub container into the container immediately below it in the tree-view.

**Sort Order**
Sorts the tree-view to match the order defined in *...\AssetX\bin\SortOrder.txt*. Note that this operates on the first level only, the ordering of tags in sub containers will not be affected by this command.

**Find & Replace**

**Set Undo Mark [^K]**
Saves the current state of the tree such that it can later be restored by Undo. Most of the commands which operate on the tree-view set an undo mark automatically.

**Undo [^Z]**
Reverses the last change to the tree-view contents.

## Creating & Editing Tags

Previous Top Next

### Creating New Tags

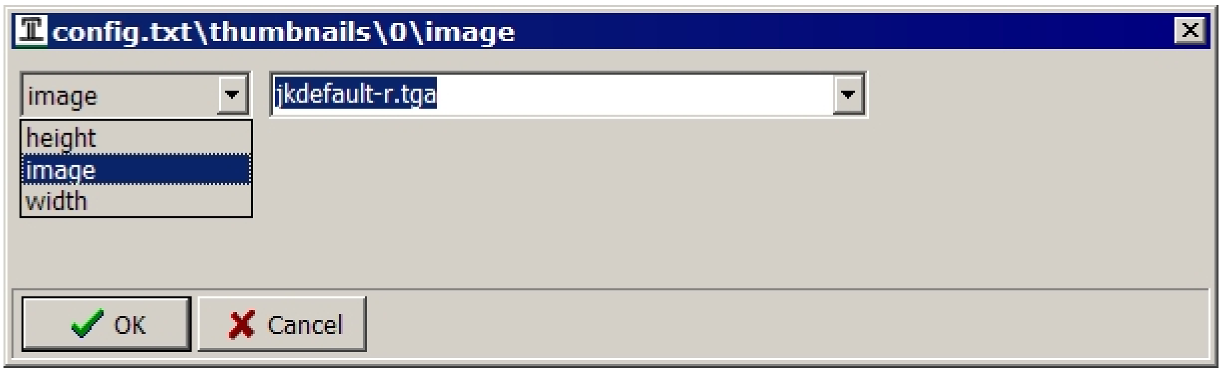

config.txt\thumbnails\0\image

image: jkdefault-r.tga

When you create a new tag you will be offered a context sensitive list of possible items from which you can select. For instance the example above shows the options that will be presented when you add a new item to a thumbnail sub container. In this circumstance the only tags available are *image, height* and *width*. If the *image* tag already existed you would only be offered the *height* and the *width*.

In other cases the controls in the edit dialogue will be different, depending on the type of data required. The dialogues for editing existing tags are similar.

Wherever possible you will be provided with a list or edit box containing options, or default values, for the tag data.

The tags and containers available will depend on the trainz-build which is currently selected. (see Trainz-Build Menu)

**Editing Tags**
In general you can only edit the tag data and not the tag name. If you need to do this you can either create a new tag and delete the old one or edit the tag manually in the configuration main view.

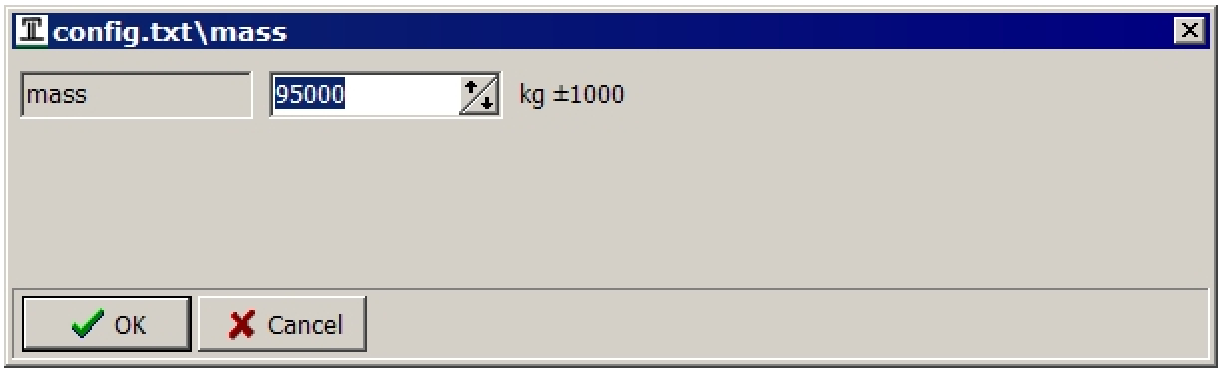
All of the editors will wrap the output in quotes if appropriate for that particular tag.

**Boolean Data**
Simple yes/no items will present a checkbox.

**String Data**
String data can be edited via an edit box. In some cases, kuid values for instance, invalid keys will be ignored to help ensure that the output format is correct.

**Localisation**
If an alternative language is specified via *Options/Preferences*, the English language tag and its localised equivalent will be created or opened for simultaneous editing.

**Numeric Data**

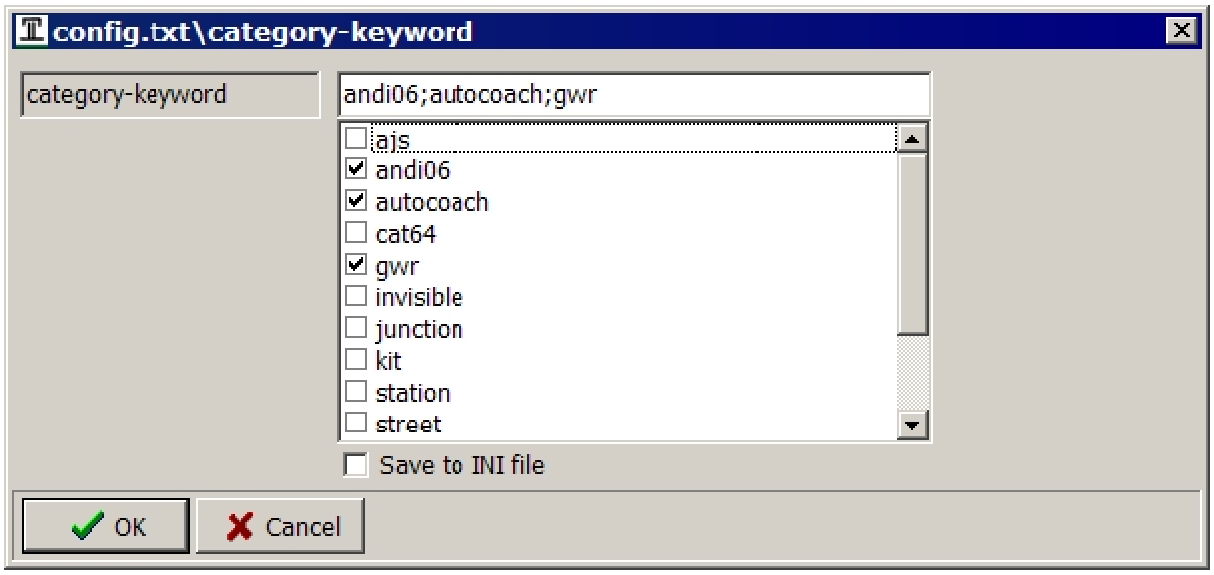

config.txt\mass

mass 95000 kg ±1000

Tags which require numeric entries (or number lists) provide edit boxes with spin buttons, you can either type the value or use the buttons to increase or decrease it. Left or right clicking on the units label to the right will increase or decrease the amount by which the data changes. Right clicking a spin button will zero the editor. Maximum, minimum and default values will be applied as appropriate.

**Enumerations**
Tags which require selection from a predefined range of values (such as *nightmode* or *trainz-build*) will provide a drop down list, usually including the meaning of the selection.

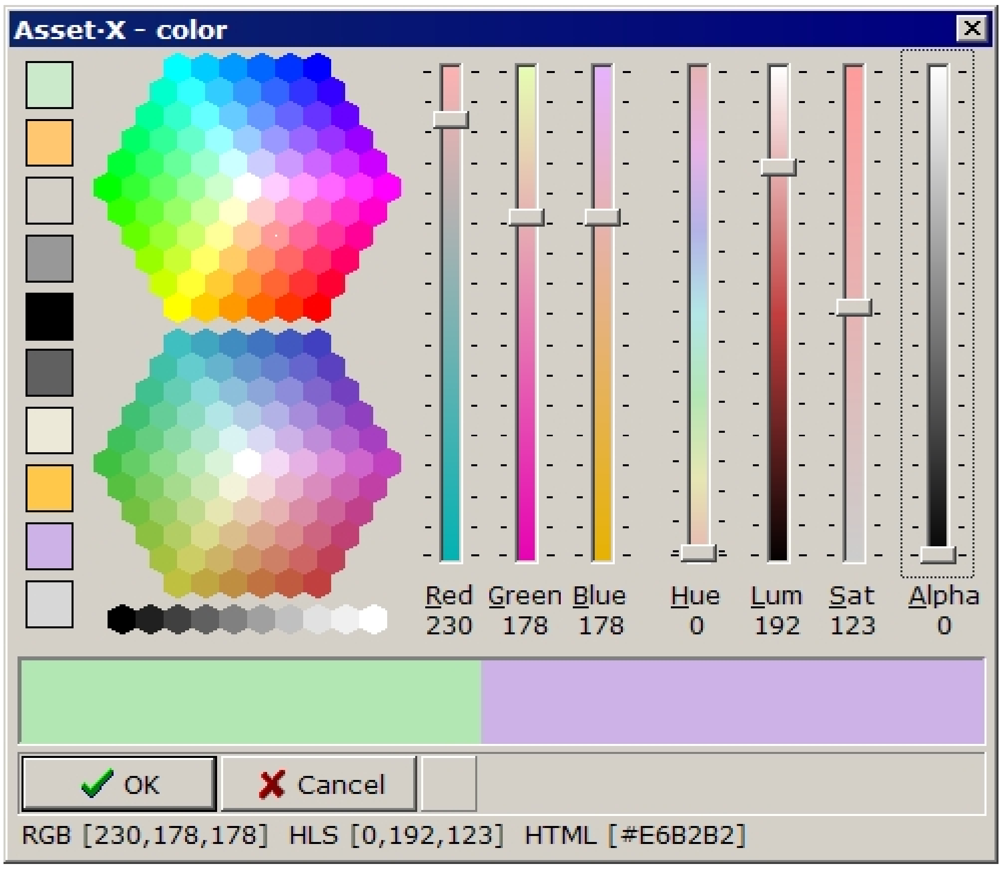
**Set Data**
Tags (such as *category-region*) which allow more than one predefined item to be included will provide a checkbox list.



Select all of the items required and the tag value will be formatted to match your selection. Alternatively you can enter items directly into the edit box.  For *category-keyword* and similar tags checking *Save to INI file* will save the contents of the list to the INI file for future use.

**Colour Values**

TRS uses several colour specifications with and without transparency and the same editor dialogue is used for all of them.



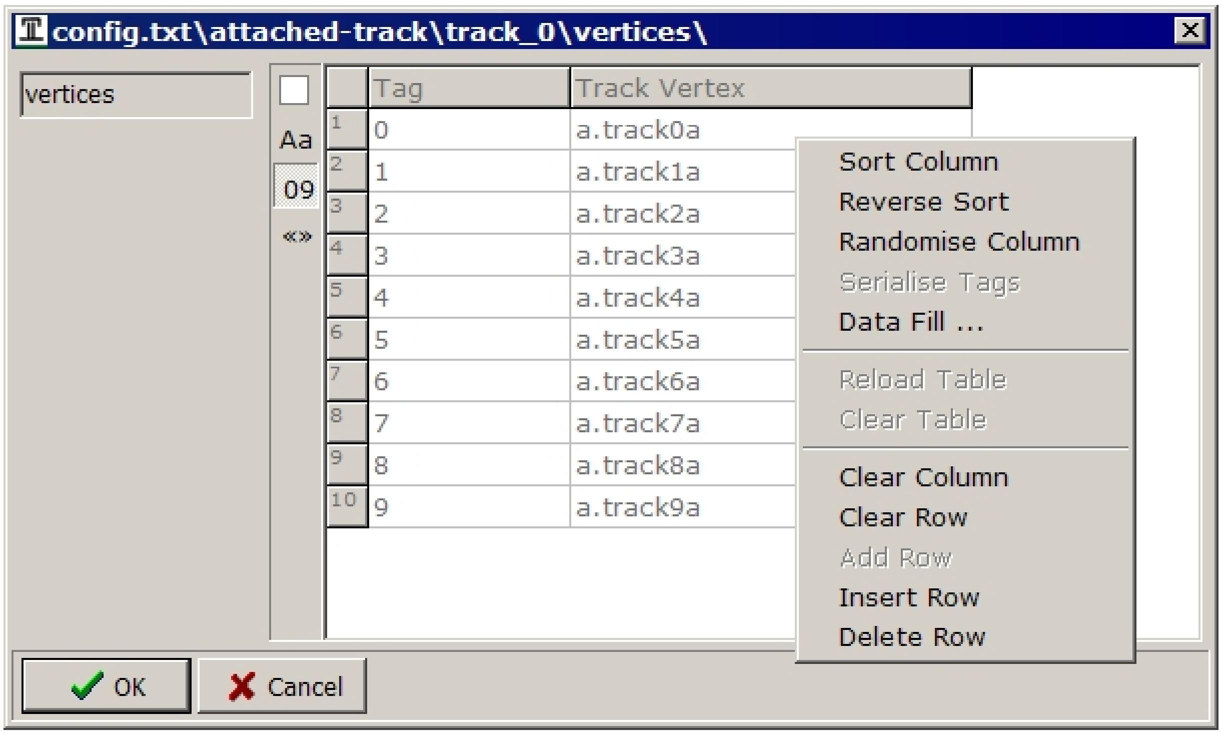
RGB [230,178,178]  HLS [0,192,123]  HTML [#E6B2B2]

Select a new colour from the sample swatches and use any of the trackbars to make fine adjustments, the mouse wheel is the easiest way to do this.  You can also use the Colour Picker button to capture a colour from anywhere on your monitor.

If the colour specification includes an alpha component, in smoke blocks for instance, the sample panel will be partially transparent to allow you to visualise the result. Click OK to save the new colour.

**Files, Meshes and Attachment Points**
A drop down list will be provided containing options generated by scanning the asset.

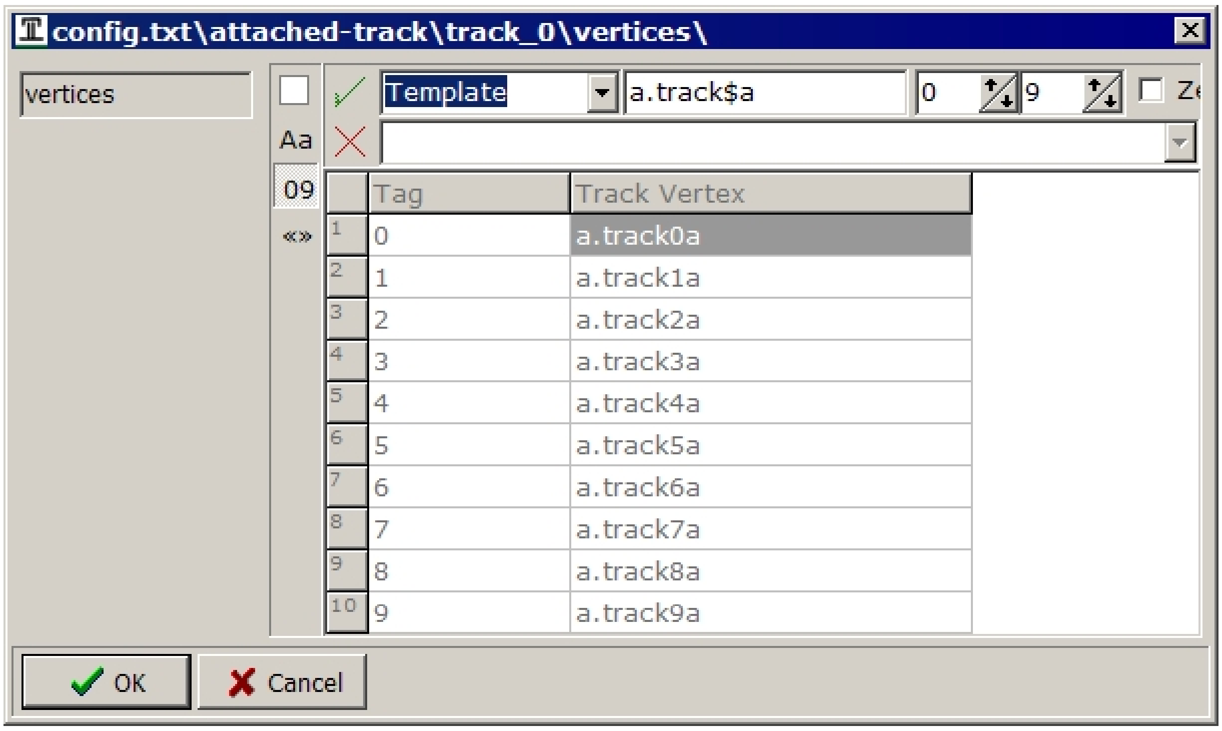
**Tables**
Containers such as kuid and attachment-point lists can either be edited by selecting their individual components or by selecting the container itself.  In the latter case the editor will consist of a grid within which any number of items can be added at the same time:



|  | Tag | Track Vertex |
|---|---|---|
| 1 | 0 | a.track0a |
| 2 | 1 | a.track1a |
| 3 | 2 | a.track2a |
| 4 | 3 | a.track3a |
| 5 | 4 | a.track4a |
| 6 | 5 | a.track5a |
| 7 | 6 | a.track6a |
| 8 | 7 | a.track7a |
| 9 | 8 | a.track8a |
| 10 | 9 | a.track9a |

Use the right-click menu to add or delete rows, to randomise a column or to clear the table. Click on the column headers to sort by that column, click twice to reverse the order.

Use *Data Fill* to fill the grid with new data:


config.txt\attached-track\track_0\vertices\

vertices

Template | a.track$a | 0 | 9

| | Tag | Track Vertex |
|---|---|---|
| 1 | 0 | a.track0a |
| 2 | 1 | a.track1a |
| 3 | 2 | a.track2a |
| 4 | 3 | a.track3a |
| 5 | 4 | a.track4a |
| 6 | 5 | a.track5a |
| 7 | 6 | a.track6a |
| 8 | 7 | a.track7a |
| 9 | 8 | a.track8a |
| 10 | 9 | a.track9a |

The template option will generate a list based on the contents of the text and number boxes to the right. In the example above 10 values will be produced by substituting each number in the range for the $ sign in the text box. Select the green tick to execute *Data Fill* and use *Serialise Tags* to provide the numeric indices. The *Data Fill* command will add new rows as necessary.

As an alternative to using a template, the lower drop down list will contain a list of points found by scanning the asset. You can select from these to fill the table.

Whilst this example is for a list of attachment points, the same editor is used for other kinds of list.

In all cases the formatting of the tag value is checked and modified as necessary when you close the editor.

**Validation**

Asset·X does not enforce validation except in the sense of attempting to guarantee that the formatting of the document is correct. As far as possible the tag and container editors disallow invalid keys and add quotes where appropriate but neither copy and paste, nor direct editing of data is disallowed.

The only validation performed within the Configuration Tree and when data is written to file is to ensure that container names have no spaces and that tag names have at least one. This is necessary to guarantee that the layout of the file will be valid, with braces in all the right places.

Unlike CMP therefore, you are free to get it wrong. Auran's error checking will advise you when something should be corrected although I should warn you that CMP isn't always right and will identify some items as errors when they are not and vice versa.

## Replication

Previous Top Next

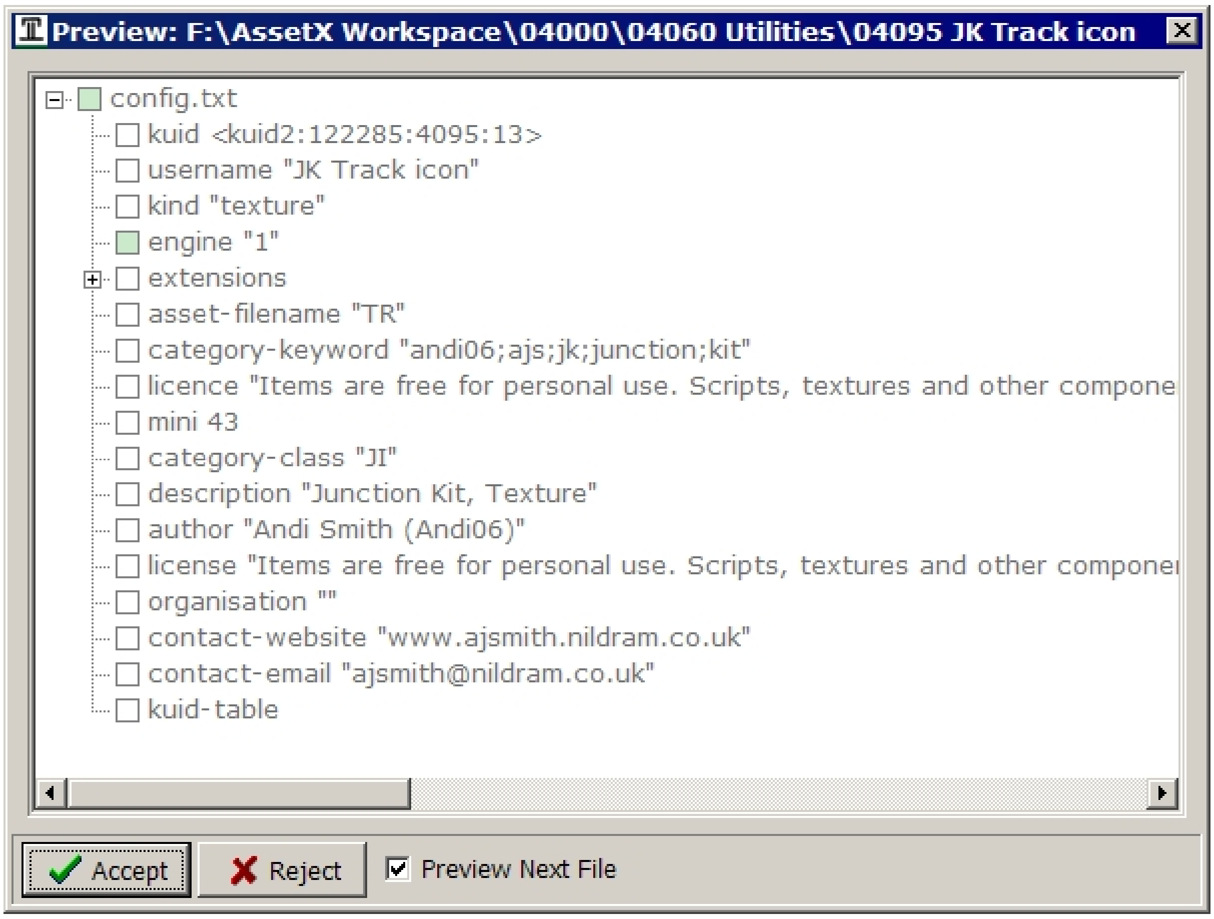
Asset·X can be used to replicate tag data from the currently loaded *config.txt* to any number of others. This requires the user to mark up the required changes in the Outline and to tag any number of target assets in the Asset List.

The changes that can be replicated are as follows:

- The value of any tagged item in the current Outline will be used to create or amend the corresponding item in each of the target assets.
- Any item tagged for deletion in the source will be deleted in each of the targets.

This can be applied to containers or sub containers as well as individual tags.

When the source data has been marked up and the target assets selected, the operation is initiated by selecting *File/Replicate to Checked Assets* from the Main Menu, which will display this dialogue:



Preview: F:\AssetX Workspace\04000\04060 Utilities\04095 JK Track icon

```
config.txt
  kuid <kuid2:122285:4095:13>
  username "JK Track icon"
  kind "texture"
  engine "1"
  extensions
  asset-filename "TR"
  category-keyword "andi06;ajs;jk;junction;kit"
  licence "Items are free for personal use. Scripts, textures and other compone
  mini 43
  category-class "JI"
  description "Junction Kit, Texture"
  author "Andi Smith (Andi06)"
  license "Items are free for personal use. Scripts, textures and other compone
  organisation ""
  contact-website "www.ajsmith.nildram.co.uk"
  contact-email "ajsmith@nildram.co.uk"
  kuid-table
```

The dialogue shows the end result of applying the defined changes to the first of the target assets (changes which will be made are marked in colour).

Press *Accept* to save the proposed changes to file or *Reject* to leave the target asset unchanged.

If *Preview Next File* is checked then the next target asset will be presented for approval in the same way. If it is unchecked however your choice will be applied to all of the remaining targets with no request for further confirmation.

The *Replicate* command creates backups of *config.txt* for each of the target assets but in case of error the backups would need to be manually and individually restored. In consequence you should use this command with care.

## Viewer Pane

Previous Top Next

If possible the viewer pane will display the contents of the currently selected item from the configuration or file panes. If this is a text file the contents of the file will be displayed in a simple text editor; for images or meshes, the file contents will be displayed graphically.

Where the selected item is a configuration tag or container the viewer panel may show options, help text or additional information applicable to the selection. In the case of an attachment point a table will be produced showing the names and locations of points within all of the meshes included in the model.

Viewer panes each have their own options, see the appropriate sections below for details.

## Text Editor

Previous Top Next

When a text based file is selected from the file pane, that file will be opened in the text editor. The contents may be edited and saved back to disc.

The text editor is also used to display information related to tags which have been selected in the Configuration Pane, in this case the editor will normally be read-only but clipboard cut and copy will be available. In some cases the viewer contents can be saved or appended to a disc file.

The display font, word-wrap and tab width are preselected based on the expected contents of the selection. These can be changed using toolbar buttons. Tab-width will have no affect on *config.txt* files imported from CMP or on other files where spaces rather than tabs are used to align text.

Some text files are provided with additional options.

**config.txt**
The configuration pane shows *config.txt* in a tree view format, but is still possible to carry out manual editing if the file is loaded into the viewer. You will need to reload the Outline if you edit *config.txt* in the Text Editor.

***.xml**
These files can be manually edited if required and the file saved or passed to *TrainzMeshImporter* to create a mesh file for use in game. You can create a User Defined Tool to do this.

***.gs**
Script files can be compiled or encrypted using user defined tools. If your asset makes use of a *script-include-table* the included script must be installed in the game or a path to it's folder must be configured via the Options dialogue.

***.log**
AssetX log files report the results of validation and script commands. See Options/ Preferences for variations in the format of log files.

☐ **Maximise/Restore**
Increases the size of the Text Viewer such that it will occupy the entire height of the application screen, a second press restores the default state.

**Save File**
Opens a dialogue to obtain a path and filename and saves the currently loaded file.

**Print**
Prints the current file. Tab width, font and word wrap settings will be respected.

**Increase or Decrease Tab Size**
Adjusts tabulation for the loaded file.

**Change Font**
Toggles between proportional and fixed face fonts.

**Word Wrap**
Toggles the word wrap setting for the current document.

**Find**
Provides a dialogue in which you can enter a text string to search for. Pressing OK will move the cursor to the next instance of the search string if it exists. Pressing F3 repeatedly will move to subsequent instances. You can also select text and press F3 to search for repeated instances without opening the Find dialogue.

## Image Editor

Previous Top Next

The image viewer loads image information from most common image formats and from the standard TRS filetypes, including binary textures. Images can be resized, rotated and mirrored and tools are provided to add, remove and edit alpha channels.

Files with images larger than the viewer will be reduced to fit, very small images will be rescaled to an easily visible size, other images will be shown at actual size. You can use the toolbar zoom button to alter the image scaling.

Images with alpha channels are displayed in a twin viewer which can be configured to show the effect of the alpha and to visualise the alpha itself as a greyscale image. Note that the in game interpretation of the alpha channel varies for some texture types.

The editing capabilities are simple and are aimed at correcting common problems with textures and at manipulating transparency, for more complex procedures an external image editor can be configured via the Tool Options dialogue.

The images are read into memory and any changes are made to the in memory copy. For simplicity's sake there is no undo system but the original image can be reloaded at any time by selecting the *Revert* button. When carrying out editing that involves multiple steps you might wish to backup the image (*File/Backup File*) before editing and to save at each stage of the process. Where commands can be reversed this is noted in the relevant section.

## Supported File Formats

Previous Top Next

**File Load Formats**
The editor will read the following file formats. Where the format supports full transparency the alpha channel will be read and displayed. Where the image contains single colour transparency (sometimes known as 'colour key' and present in some *.gif and *.png images)

this will be translated to a full alpha channel.

| Filetype | Description & Limitations |
|---|---|
| *.texture | Jet Textures, Uncompressed, DXT1, DXT3 or DXT5 compression |
| *.tga | Truevision Targa, Uncompressed, RLE compression |
| *.bmp | Windows/OS2 Bitmap |
| *.jpg | JPEG images |
| *.png | Portable Network Graphic images |
| *.tif | Multiple images not supported |
| *.gif | Compuserve Gif, animation not supported |
| *.psd | Photoshop, multiple layers not supported |
| *.psp | Paintshop Pro (early versions), multiple layers not supported |

**File Save Formats**
Files can be saved to these standard TRS filetypes:

| Filetype | Description |
|---|---|
| *.tga | 24 bit RGB or 32 bit RGBA, uncompressed |
| *.bmp | 24 bit |
| *.jpg | 24 bit High Quality |

Files are always saved to a format which is compatible with TRS, irrespective of their original format on disc.  This allows you to open an image which TRS cannot load and re-save it to the same filename and filetype to overcome the problem.

Technical information relating to the image file on disc can be accessed via the *Image Information* button on the toolbar.

## Viewer Controls

Previous Top Next

The viewer panel is split horizontally and vertically into different zones.  As the mouse cursor is moved around the display panel it will change shape and caption to indicate the action which will follow a left click.  Wherever it is appropriate the alpha panel (and the floating preview if it is open) will update to match.

| Command | Effect |
|---|---|
| Rotate Left: | Rotate the image 90 degrees anti clockwise. |
| Rotate Right: | Rotate the image 90 degrees clockwise. |
| Mirror: | Flip the image horizontally. |
| Flip: | Flip the image vertically. |
| Visualise Alpha: | Toggle between a normal diffuse display and a semi transparent display mode.  This option is only available if the current image actually has an alpha channel data. |
| Zoom: | Enlarge or reduce the image in the display window. The effect of this command depends on the size of the image and the current size of the display panel. For larger images there may be no effect. |
| Background: | Change the image background.  This may make it easier to visualise the effect of alpha transparency. |

These commands are all reversible.

Other editing and display options can be accessed via the toolbar buttons or the right click context menu.

**Alpha Channels**
There are two possible classes of data which an alpha channel can contain:

Masked alpha holds pixel values of 0 or 255 only, black or white. Masked alphas should be used wherever possible since there are fewer in game display problems associated with this kind of transparency.

Blended alphas are similar to 256 colour greyscale images. The alpha can contain any value between 0 and 255 for each pixel.  This allows the edges of images to fade but it will often give rise to problems in game because it is difficult for the graphics card to work out whether an alpha blended plane is in front of, or behind another plane.

An alpha channel is blended if any single pixel contains a value other than black or white.

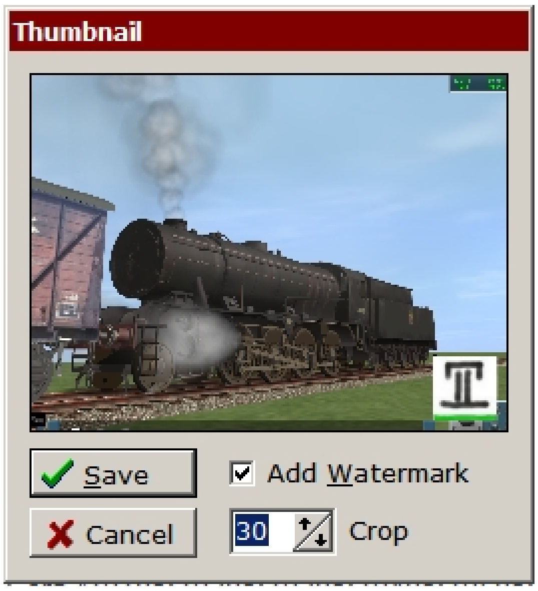
## Imaging Commands

Previous Top Next

**Maximise/Restore**
Increases the size of the Text Viewer such that it will occupy the entire height of the application screen, a second press restores the default state.

**Save File**
Opens a dialogue to obtain a path and filename and saves the currently loaded file.

**Create Thumbnail**
Opens a dialogue to allow a standard240 x 180 thumbnail image to be created from a file on disc. If the file *watermark.tga* exists in the *...\AssetX\bin* directory this can be added to the thumbnail as an overlay. *Watermark.tga* should be a 240 x 180 x 32 bit tga image with an appropriate alpha channel. You can use the Image Editor Alpha from Image command to create a suitable file.



If the currently selected image is a thumbnail the command will open the thumbnail image to allow a watermark to be added. In any other case you will see a dialogue allowing selection of a disc image. Once opened the image can be zoomed in to crop any extraneous detail.

**Image Information**
Opens a text viewer with technical information related to the loaded file.

**Floating Preview**
Floating preview launches a separate window showing the current image with alpha transparency enabled. Click and drag on any visible part of the display to move the window, reselect the toolbar button or press any key to close it.

The preview can help to assess the effect of the current alpha channel data when viewed against a range of different backgrounds. Editing commands which change the main image or its alpha channel will also update the Preview window.

Floating Preview is only available if the current image has valid alpha channel data.

**Invert Alpha Channel**
This reverses the transparency of the image. For masked alphas black is changed to white and vice versa. For blended alphas the alpha channel is changed to a photographic negative of itself. This command is reversible.

**Remove Alpha Channel**
Clears any alpha channel data and closes the alpha channel viewer pane.

**Blend to Mask**

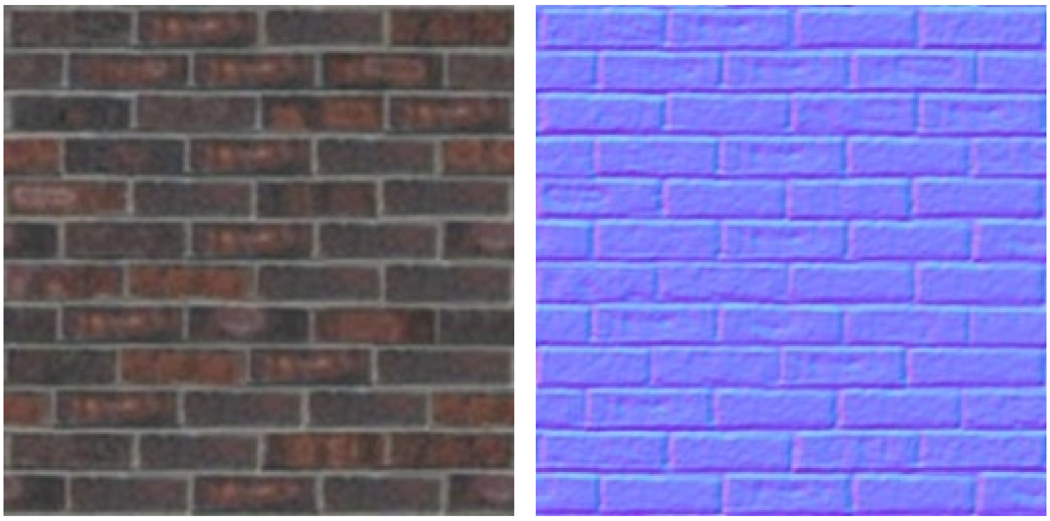
Converts a blended alpha to a masked alpha. The image is scanned and any pixel value of 127 or less is changed to 0 (Black). All other pixels are changed to 255 (White).

**Mask to Blend**
This carries out the reverse operation by blurring the edges of any boundary between white and black. This will have the effect of smoothing out any hard edges in the image transparency but will be prone to depth sorting issues in the game.

**Alpha from Image**
Creates an alpha channel from the current image.

**Import Alpha from File**
Imports an alpha channel from another file on disc.

**Normal Map Filter**
The Normal Map command allows you to create a normal map based on the current image. The image is converted to greyscale before processing.

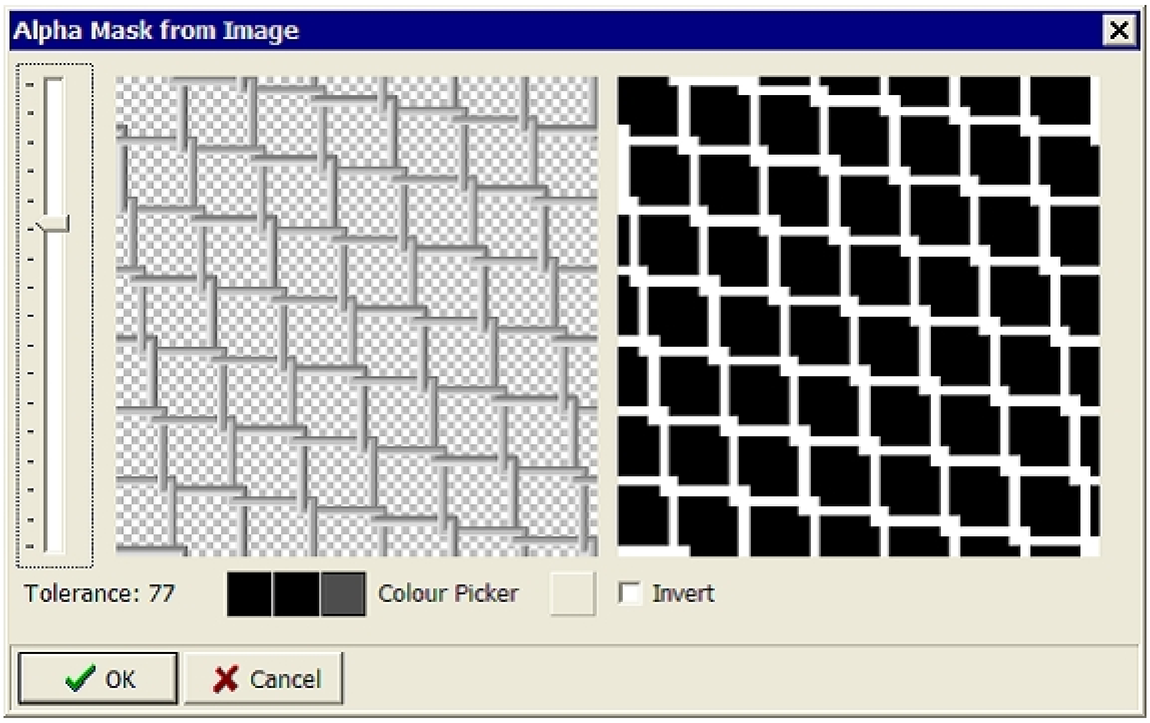


If the alignment of the texture mapping in your model is rotated or inverted you may find it helpful to rotate or flip the image in the editor before applying the normal filter.

**Resize Image**
Resizes the current image.

**Make Non Uniform**
Converts any image which has a uniform colour to pass CMP validity checks.

**Revert**
Reloads the current image from disc, abandoning any changes you might have made.



The *Alpha from Image* command generates an alpha channel by reading the colour data in the image and changing any colours within a specified range to black, rendering them transparent in the combined image.

When the editor is first loaded it assumes that the colour of the top left pixel will be used to create the mask, using the tolerance setting that was last applied. For straightforward tasks this will usually be what is required. The mask colour can be changed using the *Colour Picker* which allows you to capture a new colour from the image (or in fact from anywhere on screen) and the tolerance can be altered using the slider on the left hand side of the editor.

The three coloured panels show the selected mask colour and the lightest and darkest limits of the range. The tolerance value sets out the limits of the range selection. A selected colour of 127,127,127 with a tolerance of 10 will select any colour within the range 117,117,117 to 137,137,137 and set the corresponding pixels to be transparent.

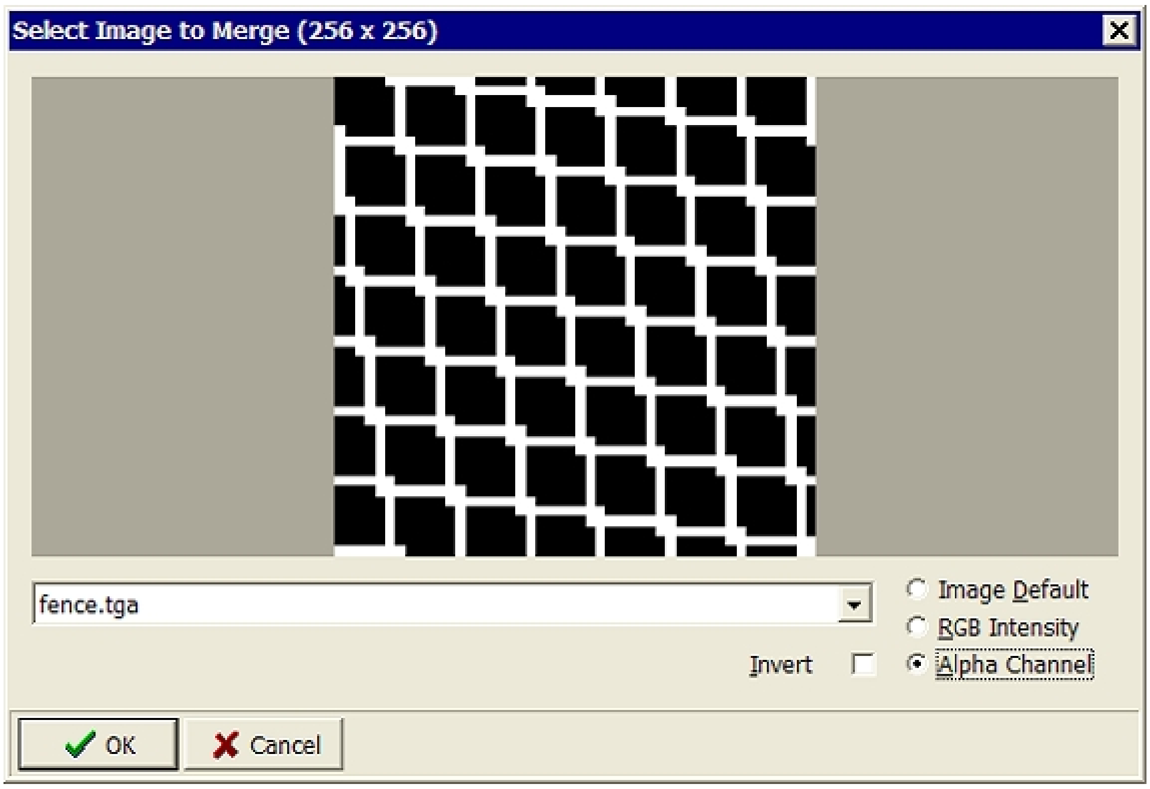
*Alpha from Image* produces masked alpha channels. If you need to soften the edges you should use *Mask to Blend* after you have finished with the *Alpha from Image* tool.

The editor operates on a thumbnail of the image to allow the tolerance to be adjusted in real time on most machines. For this reason there may be discrepancies in the finished result, particularly where the detail is fine.

You will find that selecting the trackbar and then using your mousewheel gives the easiest method of controlling this tool.

## Import Alpha from File

Previous Top Next


Select Image to Merge (256 x 256)

fence.tga

Image Default
RGB Intensity
[✓ Alpha Channel]

Invert ☐

*Import Alpha from File* enables you to add alpha channel data from another file on disk. This is most often used where *texture.txt* files specify different images for the Primary and Alpha components. Call *Import Alpha* on the Primary texture to import the secondary image and produce a single file with an alpha channel. You will need to adjust the corresponding **.texture.txt* file to match.

There will sometimes be a short delay after pressing the command button before the editor dialogue is displayed, particularly if the asset folder contains many large images.

The dialogue provides a list of all the images in the current asset folders (including subfolders) which are exactly the same size and orientation as the target file. You can use the image RGB data (which will first be converted to greyscale) or any included alpha channel information to create a new alpha channel for the main image.

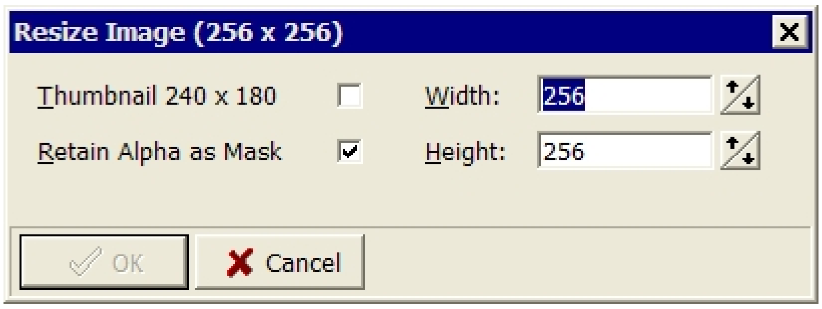
Pressing OK will create a new alpha channel for the main image based on your selections, any existing alpha data will be overwritten.

The imported image will usually be redundant after this command is executed in which case it can be deleted after checking that is not also part of another texture.

This command will also be invoked in response to the Make Non Uniform command in cases where AssetX recognises that importing an alpha channel may be an appropriate solution.

## Resizing Images

Previous Top Next



The *Image Resize* tool allows you to amend image dimensions to suit the standards required by TRS. Image dimensions for use in textures must be powers of 2 with maximum width:height ratios of 8:1 or 1:8.

Apart from the *Thumbnail* checkbox, the dialogue buttons are locked to these standard image sizes, if for any reason you need to use a different dimension you can type your values into the width and height boxes.

The resampler code used is the most accurate available but it is always better to go back to the original image if possible, this is especially true for jpeg files.

## Make Non Uniform

Previous Top Next

This command reduces the resource demands of textures which are composed of a single colour only. It can be applied to TGA images which have a consistent RGB or RGBA value for every pixel. These files will be flagged as errors or warnings when the asset is installed via CMP and in some cases may prevent the asset from appearing in game.

This is done by reducing (or sometimes increasing) the image size to 8 x 8 pixels and making a marginal adjustment to the blue value of the bottom left pixel which will be nearly impossible to detect visually. The same size is used irrespective of the size and proportions of the existing file since mapping co-ordinates are irrelevant when the texture has no variation.

In some instances a single colour file will be part of a texture which uses opacity. Resizing the image in these cases will replace the single colour issue with a different error because the diffuse and alpha textures will then be of different sizes. In these cases the best solution will often be to combine the diffuse and alpha textures into a single tga with an alpha channel. AssetX will recognise the situation and will call up the Import Alpha From File editor rather than amending or resizing the texture. This will usually clear the error although occasionally the Make Non Uniform command will still be necessary after combination.

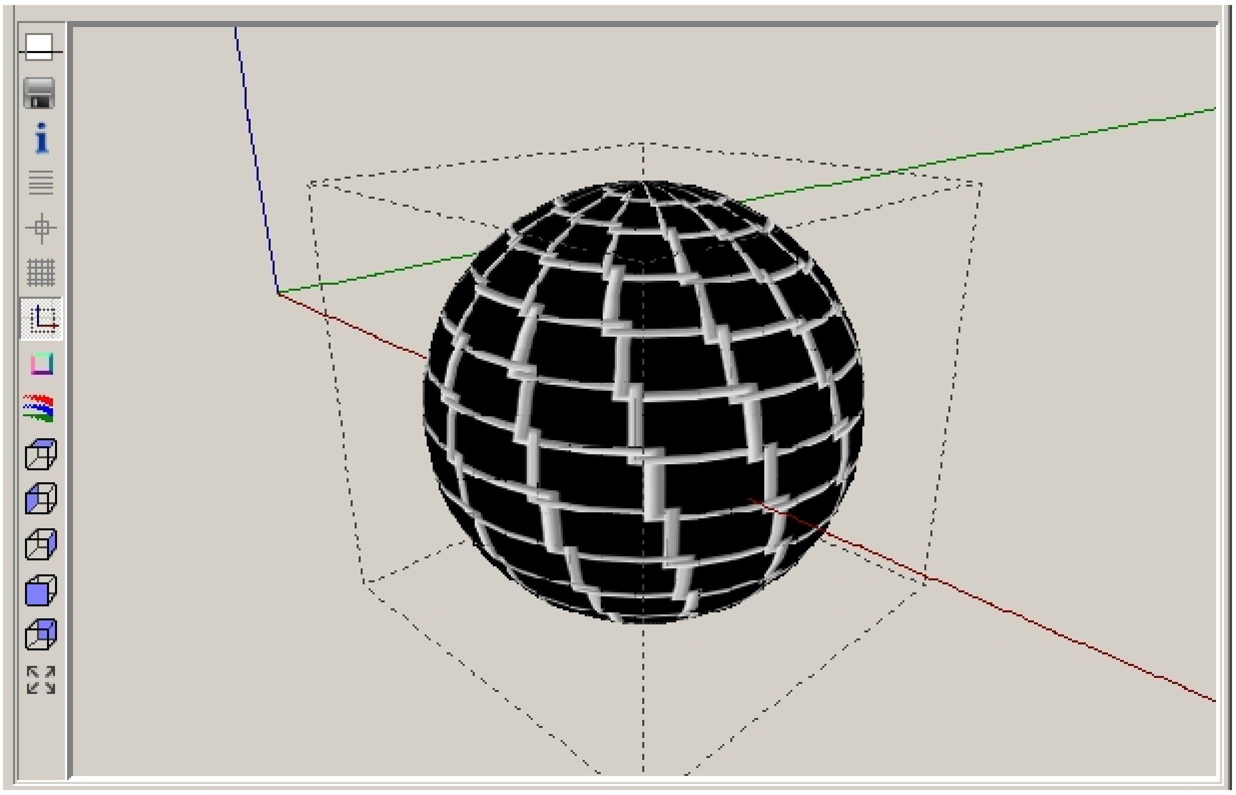
This command works around CMP error checking but it does not solve all of the problems that CMP is looking for. Where you have access to the original source files you should always modify them in preference to editing the texture. This might be done by remapping to a small area of an alternative existing texture or by applying an m.notex material. You should only use *Make Non Uniform* where these options are not available.

In some cases the image is not edited but is replaced with a new file with the same base colour. For this reason the command cannot be reverted.

*Make Non Uniform* can also be applied to BMP and JPG images.

## Mesh Viewer

Previous Top Next



This viewer displays the currently selected mesh in a rendered perspective view.  The textures used by the model must be available within the asset folder to allow the mesh to be rendered.

**Display Control**
The toolbar buttons can be used together with mouse and keyboard to modify the display.

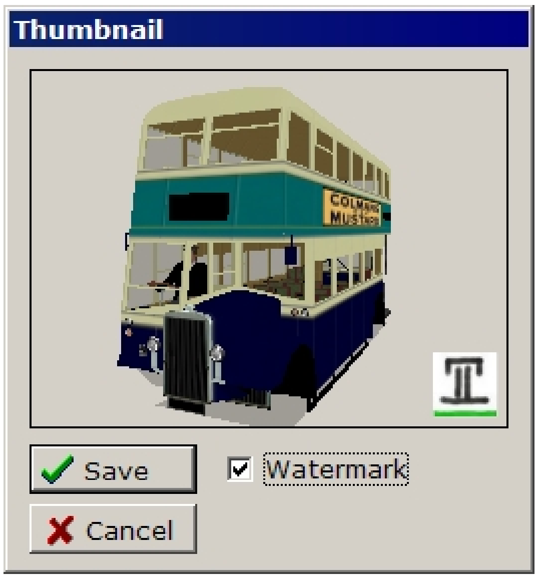
- To rotate the model click and drag with the left mouse button or use the Cursor Keys. The centre of rotation will be the centre of the object's bounding box.

- To move the model within the viewport click and drag with the middle or wheel button or use Control+Cursor Keys.

- To zoom in or out rotate the mouse wheel or use Shift+Up and Shift+Down.

**Maximise / Restore Viewer**
This button allows the Mesh Viewer to be enlarged to occupy the entire right hand half of the application workspace.

**Make Thumbnail**



This is similar to the corresponding Image Viewer command except that it works on the selected mesh. The command opens a dialogue to allow a 240 x 180 thumbnail image to be created from the current mesh view. If the file *watermark.tga* exists in the *...\AssetX\bin* directory this can be added to the thumbnail as an overlay. *Watermark.tga* should be a 240 x 180 x 32 bit tga image with an appropriate alpha channel. You can use the Image Editor Alpha from Image command to create a suitable file.

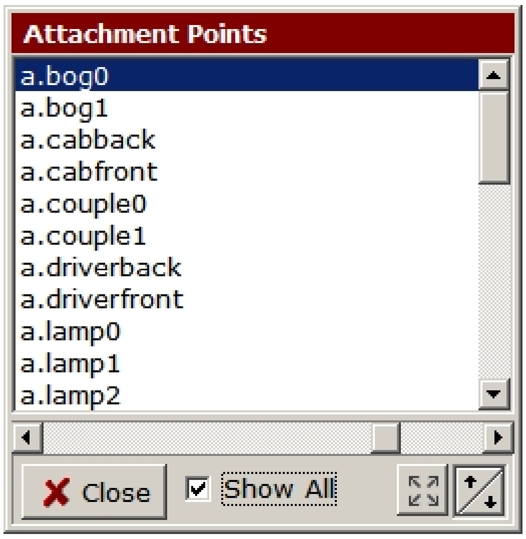
### Mesh Information

Opens an alert panel to display the dimensions of the mesh bounding box (the maximum length, width and height of the mesh) together with the mesh polycount.

### View as Text

Opens a text window containing technical data for the loaded mesh.

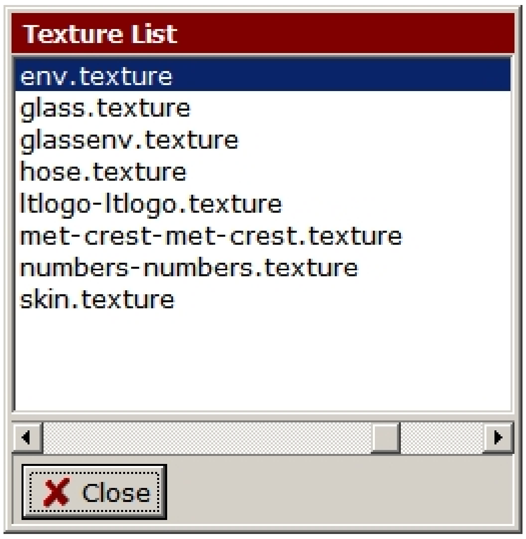
### Attachment Points



a.bog0
a.bog1
a.cabback
a.cabfront
a.couple0
a.couple1
a.driverback
a.driverfront
a.lamp0
a.lamp1
a.lamp2

Opens a text window to display a list of attachment points contained in the mesh. There are controls at the bottom of the window to adjust the line colour of the display and the size of the attachment point symbols.

### Texture List



Opens a text window to display the names of all the textures used by the mesh. The selected texture will be displayed in the viewer pane (unless this is a reflection or normal map)

**Display Guides**
Toggles display of the object's bounding box and the positive X, Y and Z axes of the model space origin. This is especially useful when examining invisible meshes.

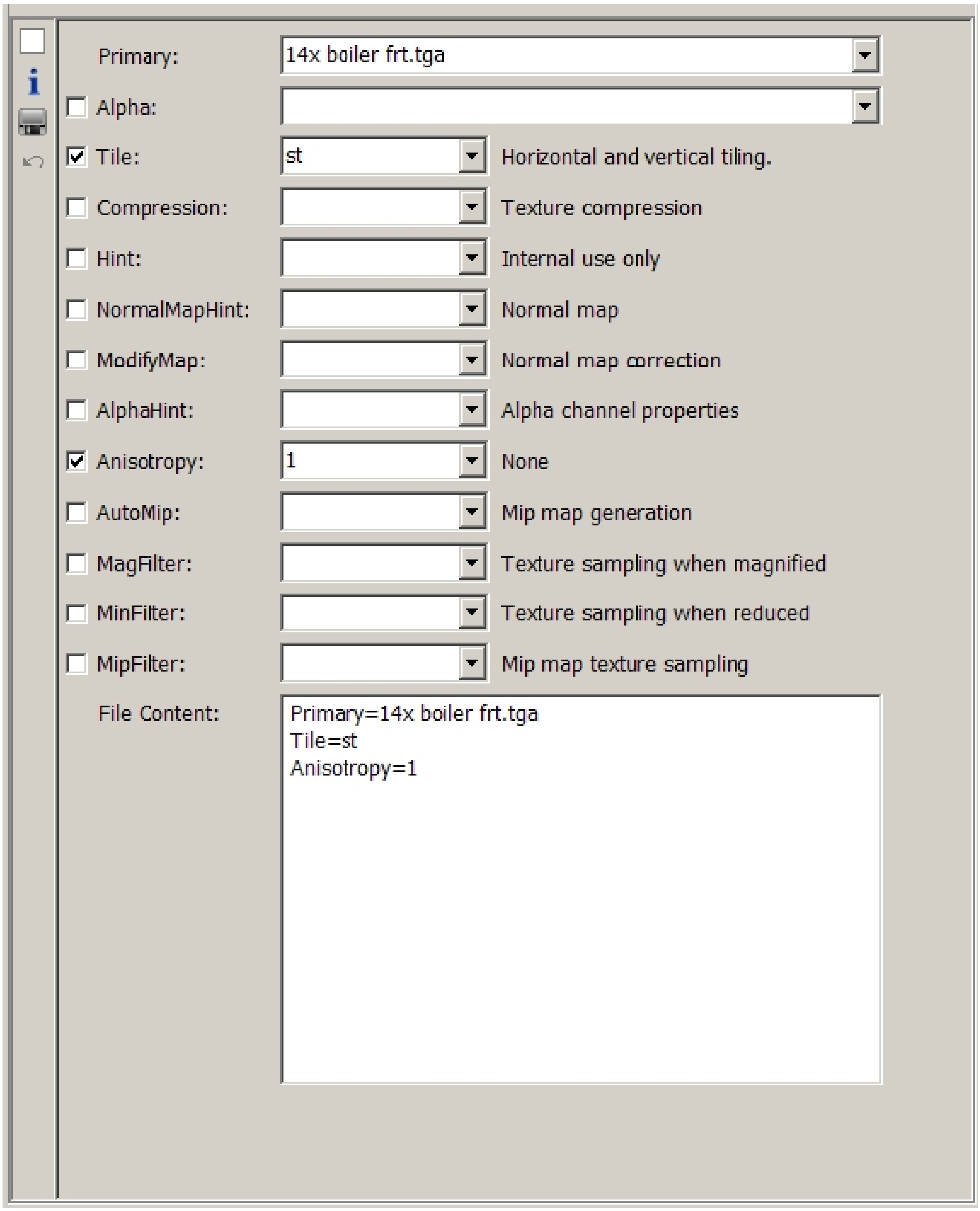
**Show Bumps**
If the mesh includes any textures which reference a normal map, this command turns off the texture rendering and displays the bump maps only using greyscale rendering.

**Background Colour**
Changes the viewer background colour.

**Preset Views**
Selects predefined top, left, right, front or back views respectively. These are perspective views aligned on the centre of the mesh bounding box.

**Zoom to Bounds**
Zooms in or out such that the bounding box of the mesh in the current view approximately fills the viewport. Few meshes entirely occupy their bounding box so this will usually leave plenty of space around the margins, you can use the mouse wheel to zoom in closer still. This command does not alter the angle of view.

## Texture File Editor



| | | | |
|---|---|---|---|
| | Primary: | 14x boiler frt.tga | |
| [ ] | Alpha: | | |
| [✓] | Tile: | st | Horizontal and vertical tiling. |
| [ ] | Compression: | | Texture compression |
| [ ] | Hint: | | Internal use only |
| [ ] | NormalMapHint: | | Normal map |
| [ ] | ModifyMap: | | Normal map correction |
| [ ] | AlphaHint: | | Alpha channel properties |
| [✓] | Anisotropy: | 1 | None |
| [ ] | AutoMip: | | Mip map generation |
| [ ] | MagFilter: | | Texture sampling when magnified |
| [ ] | MinFilter: | | Texture sampling when reduced |
| [ ] | MipFilter: | | Mip map texture sampling |

File Content:

```
Primary=14x boiler frt.tga
Tile=st
Anisotropy=1
```

The texture file editor will be opened whenever you select an *image.texture.txt* file either from the File List pane or from a *texture* tag in the Outline. This screen allows the parameters of the texture to be amended in detail.

To modify binary textures, those with an *image.texture* filename you will need to extract them and create an *image.texture.txt* file using Images2TGA prior to opening the editor.

Selecting a checkbox will add the default option of the associated parameter to the file, or remove it if is already present. You can amend the option required using the combo boxes. The Primary and Alpha combo boxes will contain a list of all the files found in the asset, you can also type a filename directly into these two boxes.

To the right are brief notes on the purpose of the parameters and on the meaning of the currently selected option.

The file content window will show the current content of the file and will update as you add and remove options.

### Maximise/Restore

Increases the size of the Editor such that it will occupy the entire height of the application screen, a second press and restores the default state.

### Info

Opens the WIKI page related to texture files in your default browser.

### Save File [^S]

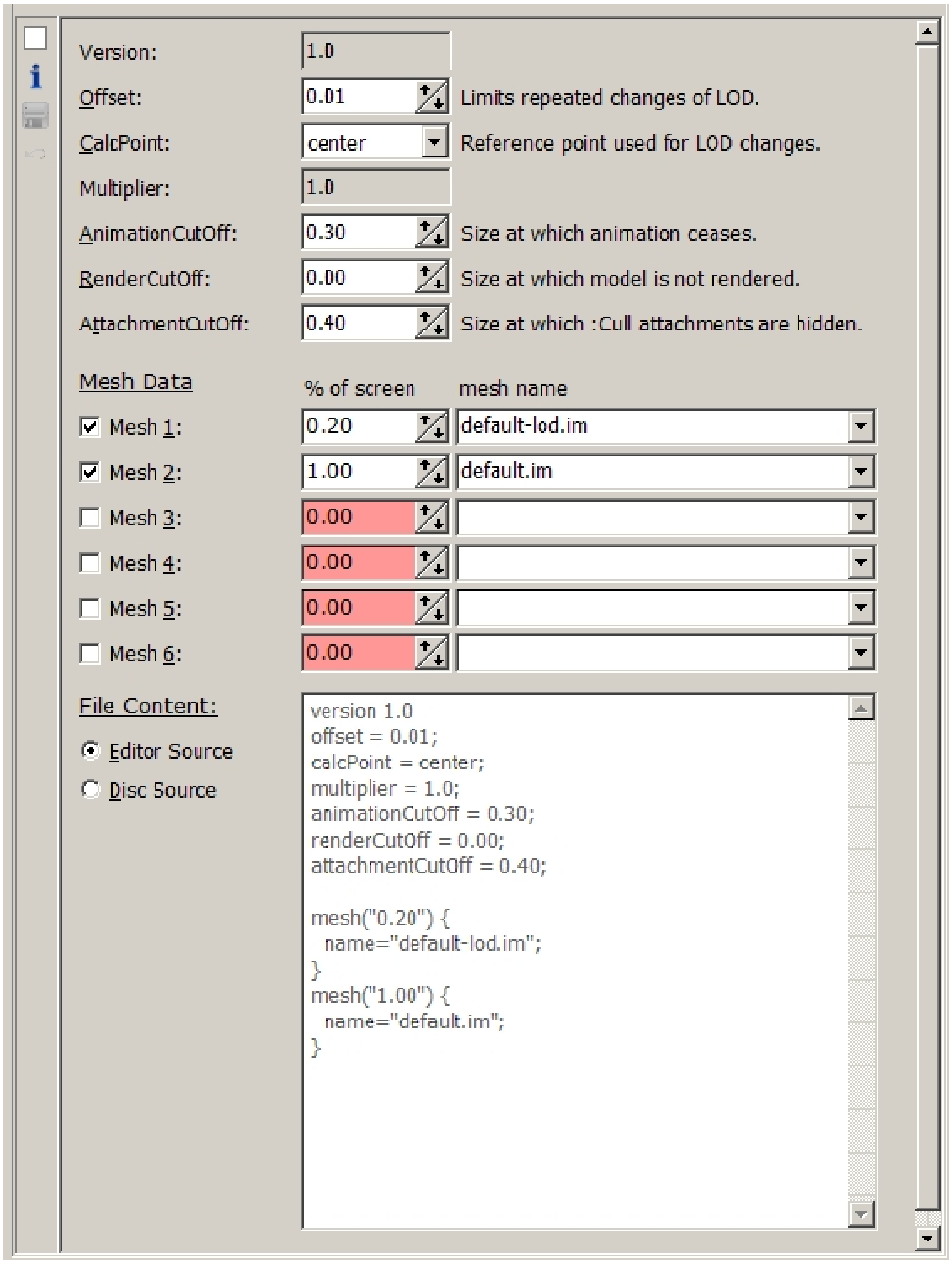
Saves the open file without prompting for a filename.

### Reload File

Reloads the file from disc abandoning any changes you might have made.

## LOD Mesh File Editor

Previous Top Next



| | | |
|---|---|---|
| Version: | 1.0 | |
| Offset: | 0.01 | Limits repeated changes of LOD. |
| CalcPoint: | center | Reference point used for LOD changes. |
| Multiplier: | 1.0 | |
| AnimationCutOff: | 0.30 | Size at which animation ceases. |
| RenderCutOff: | 0.00 | Size at which model is not rendered. |
| AttachmentCutOff: | 0.40 | Size at which :Cull attachments are hidden. |

| Mesh Data | % of screen | mesh name |
|---|---|---|
| ✓ Mesh 1: | 0.20 | default-lod.im |
| ✓ Mesh 2: | 1.00 | default.im |
| ☐ Mesh 3: | 0.00 | |
| ☐ Mesh 4: | 0.00 | |
| ☐ Mesh 5: | 0.00 | |
| ☐ Mesh 6: | 0.00 | |

File Content:

[✓ Editor Source] Disc Source

```
version 1.0
offset = 0.01;
calcPoint = center;
multiplier = 1.0;
animationCutOff = 0.30;
renderCutOff = 0.00;
attachmentCutOff = 0.40;

mesh("0.20") {
  name="default-lod.im";
}
mesh("1.00") {
  name="default.im";
}
```

The LOD file editor will be opened whenever you select a *mesh.lm.txt* file either from the File List pane or from a *mesh* tag in the Outline. This screen allows the parameters of the mesh to be amended in detail.

The LOD parameters can be edited directly and the relevant meshes can be added by selecting the mesh file from the drop down boxes in the Mesh Data section. To add a new mesh definition, select the mesh file, adjust its screen percentage value and tick the associated checkbox.

LOD files must be entered in order according to their size with the smallest LOD entered first. When this rule is not met the screen size box will be coloured and the mesh will not be added to the lm.txt file. To correct this you should adjust the mesh size until the box turns white.

To the right are brief notes on the purpose of the parameters and on the meaning of the currently selected option.

The file content window will show the current content of the file and will update as you add and remove options.

**Maximise/Restore**
Increases the size of the Editor such that it will occupy the entire height of the application screen, a second press and restores the default state.

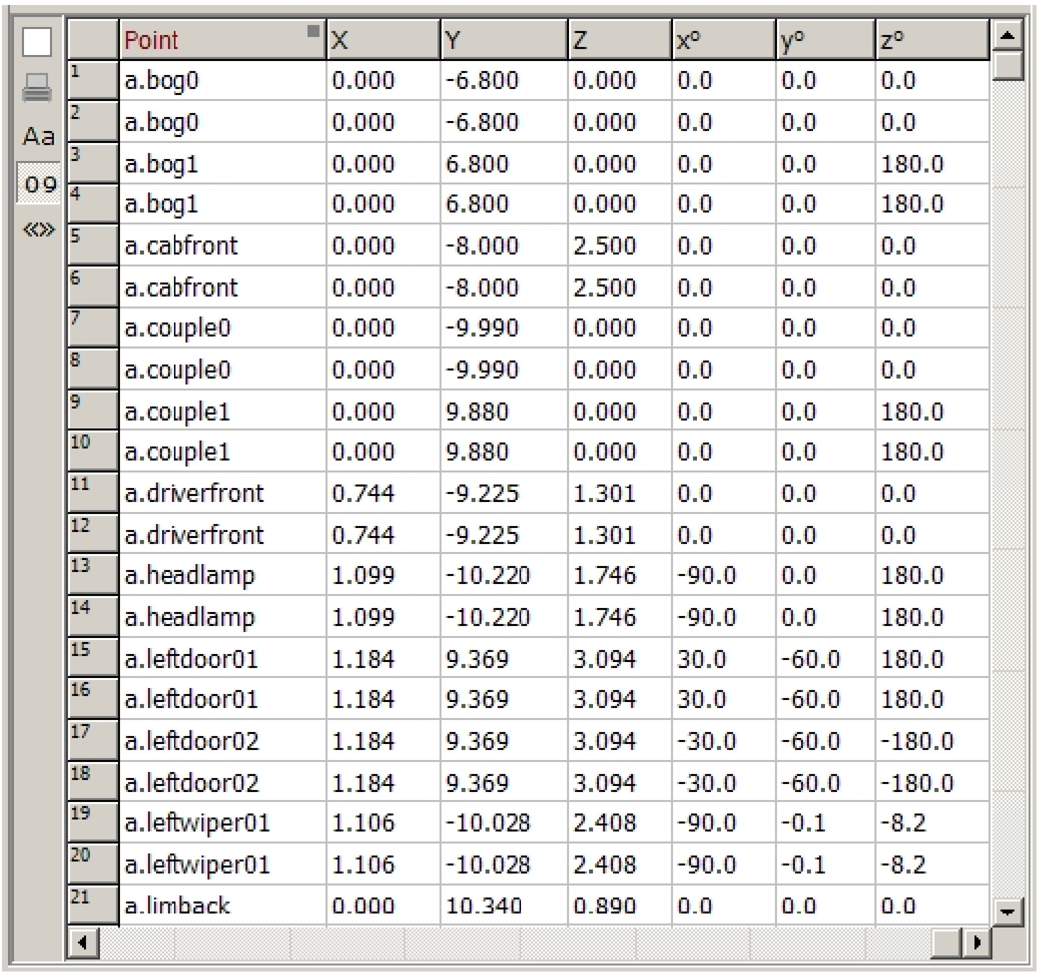
**Info**

Opens the WIKI page related to lm.txt files in your default browser.

**Save File [^S]**
Saves the open file without prompting for a filename.

**Reload File**
Reloads the file from disc abandoning any changes you might have made.

## Grid Viewer



| | Point | X | Y | Z | x° | y° | z° |
|---|---|---|---|---|---|---|---|
| 1 | a.bog0 | 0.000 | -6.800 | 0.000 | 0.0 | 0.0 | 0.0 |
| 2 | a.bog0 | 0.000 | -6.800 | 0.000 | 0.0 | 0.0 | 0.0 |
| 3 | a.bog1 | 0.000 | 6.800 | 0.000 | 0.0 | 0.0 | 180.0 |
| 4 | a.bog1 | 0.000 | 6.800 | 0.000 | 0.0 | 0.0 | 180.0 |
| 5 | a.cabfront | 0.000 | -8.000 | 2.500 | 0.0 | 0.0 | 0.0 |
| 6 | a.cabfront | 0.000 | -8.000 | 2.500 | 0.0 | 0.0 | 0.0 |
| 7 | a.couple0 | 0.000 | -9.990 | 0.000 | 0.0 | 0.0 | 0.0 |
| 8 | a.couple0 | 0.000 | -9.990 | 0.000 | 0.0 | 0.0 | 0.0 |
| 9 | a.couple1 | 0.000 | 9.880 | 0.000 | 0.0 | 0.0 | 180.0 |
| 10 | a.couple1 | 0.000 | 9.880 | 0.000 | 0.0 | 0.0 | 180.0 |
| 11 | a.driverfront | 0.744 | -9.225 | 1.301 | 0.0 | 0.0 | 0.0 |
| 12 | a.driverfront | 0.744 | -9.225 | 1.301 | 0.0 | 0.0 | 0.0 |
| 13 | a.headlamp | 1.099 | -10.220 | 1.746 | -90.0 | 0.0 | 180.0 |
| 14 | a.headlamp | 1.099 | -10.220 | 1.746 | -90.0 | 0.0 | 180.0 |
| 15 | a.leftdoor01 | 1.184 | 9.369 | 3.094 | 30.0 | -60.0 | 180.0 |
| 16 | a.leftdoor01 | 1.184 | 9.369 | 3.094 | 30.0 | -60.0 | 180.0 |
| 17 | a.leftdoor02 | 1.184 | 9.369 | 3.094 | -30.0 | -60.0 | -180.0 |
| 18 | a.leftdoor02 | 1.184 | 9.369 | 3.094 | -30.0 | -60.0 | -180.0 |
| 19 | a.leftwiper01 | 1.106 | -10.028 | 2.408 | -90.0 | -0.1 | -8.2 |
| 20 | a.leftwiper01 | 1.106 | -10.028 | 2.408 | -90.0 | -0.1 | -8.2 |
| 21 | a.limback | 0.000 | 10.340 | 0.890 | 0.0 | 0.0 | 0.0 |

The Grid Viewer is primarily used to display lists of attachment points in the asset and will be displayed whenever a helper point definition is selected in the Outline.

Clicking on the header text will sort the grid by that column, a second click will reverse the sort order. Dragging on the joint between two header cells will resize the columns.

The grid contents can also be exported to a file in **.csv* format, see File Menu for details.

**Maximise/Restore**
Increases the size of the Editor such that it will occupy the entire height of the application screen, a second press and restores the default state.

**Print**
Prints the current grid view respecting column widths and sort order.

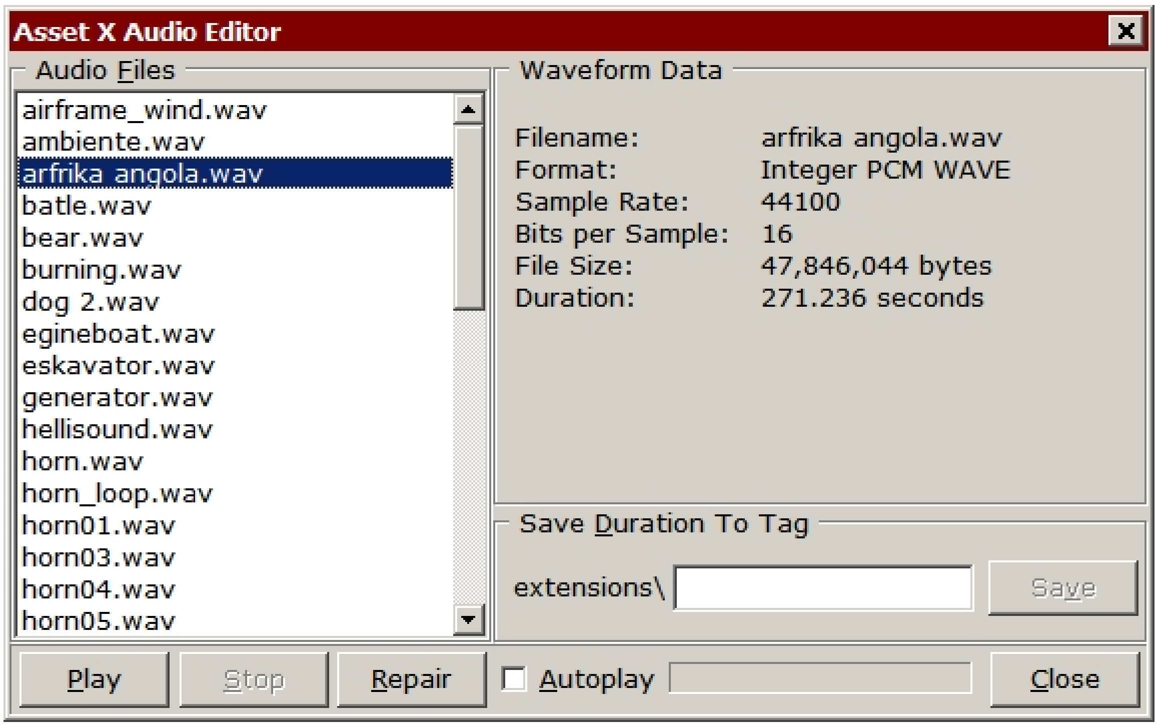
**Sort Mode**
Toggles between case sensitive and case insensitive sorting.

**Line Numbers**
Toggles display of line numbers in the left hand column.

**Autosize**
Resizes the width of each grid column to fit its contents.

## Audio Editor

Previous Top Next



Asset X Audio Editor

Audio Files

airframe_wind.wav
ambiente.wav
arfrika angola.wav
batle.wav
bear.wav
burning.wav
dog 2.wav
egineboat.wav
eskavator.wav
generator.wav
hellisound.wav
horn.wav
horn_loop.wav
horn01.wav
horn03.wav
horn04.wav
horn05.wav

Waveform Data

| | |
|---|---|
| Filename: | arfrika angola.wav |
| Format: | Integer PCM WAVE |
| Sample Rate: | 44100 |
| Bits per Sample: | 16 |
| File Size: | 47,846,044 bytes |
| Duration: | 271.236 seconds |

Save Duration To Tag

extensions\

The Audio Editor can be accessed by selecting an Audio File (*.wav or *.mp3) in the File List, right-clicking and selecting Audio Editor from the top of the context menu.

**Audio Files and Waveform Data**
The list box displays all of the audio files in the current asset, format data for the highlighted file is displayed on the right hand side of the window.

**Play, Stop and Autoplay**
These buttons control playback of the selected file. If Autoplay is checked the player will cycle through each item in the list, playing the next file as soon as the previous item has completed.

**Repair**
This re-encodes the file as an uncompressed PCM waveform and writes it back to disc. If the current format is incompatible with the game it will be converted. Mp3 files are converted to *.wav format and renamed by this command.

Where the sample frequency is not 11025, 22050 or 44100 the data is resampled to one of these rates and the bits per sample is changed to 16 if it currently has a different value. In some cases conversion from a compressed format will increase the file size dramatically.

**Save Duration to Tag**
This provides a mechanism for saving the duration of the selected file to a tag in config.txt for use by scripts which need to synchronise sound files. Initially the edit box will contain a default based on the filename of the selected file. Accept this or enter a different identifier and press [Save] The tag will be added to the extensions table.

## Main Menu Reference

Previous Top Next

The Main Menu contains commands which are relevant to the current state of the application and is split into the following sections:.

File Menu
Asset Menu
Diagnostics Menu
Trainz-Build Menu
Repair Menu
Scripts Menu
Tools Menu
Options Menu
Help Menu
Custom Menus

The menu commands available will vary depending on the current context.

## File Menu

Previous Top Next

**Add Assets [^O]**
Opens a Browse for Folder dialogue from which you can select a source folder which will be scanned for assets. Any assets found which are not already loaded will be added to the Asset List.

**Save Image As**
Allows images which are currently loaded into the image editor to be saved as TRS compatible **.jpg, *.tga* or **.bmp* files, irrespective of the original file format.

Since AssetX can load and display many image file types which are not supported by TRS, you can use this command to convert files to any of the supported image types.

**Create New Image**

Creates a new 8 x 8 pixel image to replace files which are commonly missing from assets. The blue value of the lower left pixel is adjusted marginally to pass CMP uniform colour checking. See also Make Non Uniform.

**Thumbnail From Image**
Opens the thumbnail tool (see Imaging Commands)

**Save Text File As**
Allows any data in the text editor to be saved to a plain text file.

**Save Table As**
Allows any data in the table viewer to be saved to **.csv* format, compatible with most spreadsheet software, including Microsoft Excel.

**Clear Tagged Folders, Clear Untagged Folders**
Removes tagged or untagged asset folders respectively from the Asset List. Neither of these commands will remove the currently selected asset. This simply unloads the assets from the application, it does not affect the files on disc.

**Clear All Folders**
Clears the Asset List, including the currently selected asset.

**Replicate to Checked Assets [^R]**
Allows selected data in the currently open config to be copied to all assets in the asset list which are currently tagged. See the Replication topic for further details.

**Paste Files to Tagged Assets**
Allows a file or files which have been copied from an asset to be pasted to all currently tagged folders in the Asset list.

**Backup Asset, Backup File**
Creates a backup of the selected file or asset.

**Restore Asset, Restore File**
If any backups are available, restores the selected file or folder.

## Asset Menu

Previous Top Next

**New Asset**
Opens a dialogue which allows you to create a new asset folder and a minimal config.txt. You will be asked for a kuid, asset kind, folder location and folder name.

You cannot use AssetX to create a new asset in the TRS editing folders but you can create a new asset in *TRS2004\Custom.*

**Clone Asset**
Creates a copy of the current asset under your own kuid. The command calls an additional dialogue with options allowing you to specify the kuid, username and description of the cloned asset. You can define the default kuid for cloned assets in Options/Preferences, the number will be incremented each time it is used.

You will also be asked for a folder location (anywhere on disc) and a new folder name to which the asset files will be copied. The location must already exist and must not be an Auran\editing folder or the AssetX backup folder. The location and folder name together make up the path to the new asset folder and must be unique.

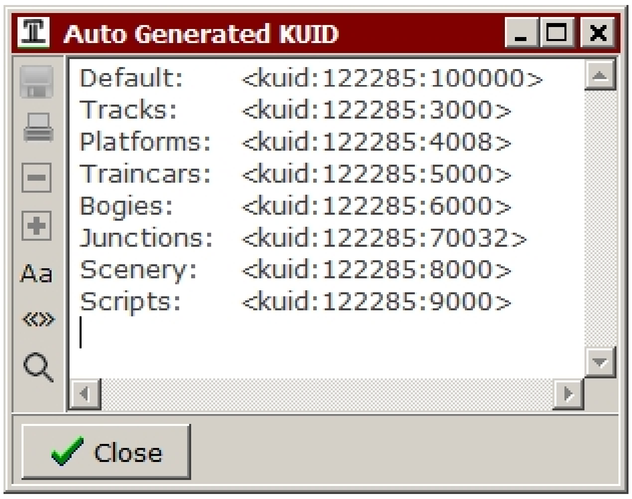
Clicking OK in this dialogue will result in the asset being cloned to the new location and opened in AssetX. The kuid, username and description tags will be updated and the amended *config.txt* saved to disc.

When a cloned asset is first created it will always be opened in AssetX, regardless of its location on disc. You should be aware that if the disc location is not a home folder the cloned asset will not be loaded in subsequent sessions.

AssetX cannot create a cloned asset within the Auran editing folders. You will need to install the asset via CMP after it has been created.

**Increment Kuid**
This will only be enabled for your own assets. Selecting this item causes the revision level of the kuid tag to be increased. <kuid:123:456> would be increased to <kuid2:123:456:2> and <kuid2:123:456:2> would become <kuid2:123:456:3>

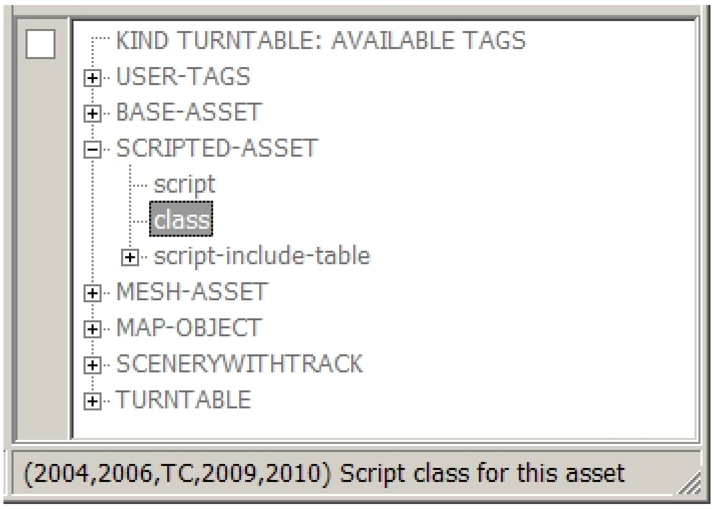
**Generate KUID Values**
Returns a list of the next available KUID for any number of user defined categories.



```
Default:     <kuid:122285:100000>
Tracks:      <kuid:122285:3000>
Platforms:   <kuid:122285:4008>
Traincars:   <kuid:122285:5000>
Bogies:      <kuid:122285:6000>
Junctions:   <kuid:122285:70032>
Scenery:     <kuid:122285:8000>
Scripts:     <kuid:122285:9000>
```

Provided that CMP 'knows' about all of your content using this utility will help to prevent duplication of IDs. After use the seed value used by CMP to assign numbers for routes and sessions will be reset to the default. See Options/Preferences for details of configuration.

**List Available Tags**
Opens a display in the viewer pane containing a list of tags and containers which can be used with the current asset kind:



```
KIND TURNTABLE: AVAILABLE TAGS
⊞ USER-TAGS
⊞ BASE-ASSET
⊟ SCRIPTED-ASSET
    script
    class
  ⊞ script-include-table
⊞ MESH-ASSET
⊞ MAP-OBJECT
⊞ SCENERYWITHTRACK
⊞ TURNTABLE
```

(2004,2006,TC,2009,2010) Script class for this asset

The display is in a tree format similar to the Outline and is categorised according to the way that the asset kind inherits its tag capability (see Inheritance) The status bar shows a help string for each tag together with details of its validity for the various versions of TRS.

**Add User Tags**
Adds or updates the user data tags in the Outline, see Options/UserData.

**Reload Home Folders [^F2]**
Loads each of the defined Home Folders.

**Custom and Editing Folders**
Loads all assets which are open for edit in the specified location. The availability of these commands will depend on which versions of TRS are installed on your machine.

## Trainz-Build Menu

Previous Top Next

**Trainz-Build**
The Trainz-Build menu defines which TRS build and version is currently in use by AssetX. This controls the behaviour of many commands and script functions. For instance if your Trainz-Build is set to 2.9 then commands which provide lists of tags (such as Insert Tag) will include tags which are valid for TRS2009.

When updating assets to a particular build it is important that the Trainz-Build menu reflects this since many of the repair functions behave differently for different builds.

By default the AssetX trainz-build will either be set permanently to your currently installed build or will change to reflect the *trainz-build* tag in the currently selected asset. This can be changed via Options / Paths & Files.

You can also set the build explicitly by selecting the appropriate value from the menu.

## Diagnostics Menu

Previous Top Next

Tools to assist in checking the current asset for possible incompatibilities. These tools do not use the same methods as CMP to verify files and tags (although they will find some errors that CMP misses) As a result there is no guarantee that an asset passing these tests will install successfully.

Most of these commands create a log file containing a report of the outcome of the command. Depending on settings made in Options/Preferences the log will be displayed once the command has finished processing.

**View Log**
Displays the current log file.

**Clear Log**
Clears the current log file.

**Validate Tags (3.7)**
Scans the current asset and checks the syntax and content of all of the tags and containers in config.txt using the trainz-build value shown in parentheses.

The outcome of the scan will be noted in the log file against each tag and the tag itself will be marked in red.

**Clear Markers**
Clears markers set by the Validation commands.

**Verify Attachments...**
Checks that attachments referenced in config.txt can be found in the correct mesh. In the case of traincar assets this command also checks that *a.limback* and *a.limfront* are present and the the names of *a.bog#* points are consistent with the bogeys container.

**Verify Files...**
Checks the asset for missing files and for files which are not required by tags in config.txt. This needs to be used with caution since unreferenced files may still be required by scripts, html pages or in some cases by external assets.

**Verify Textures...**
Checks *.texture and *.texture.txt files and advises where binary textures are present and other texture errors such as missing image files.

**Verify All...**
Carries out all of the above checks.

In cases where the selected asset is an alias, uses the asset-filename tag or contains mesh-library references, some types of check cannot be made.  The log window will warn you when this is the case.

## Repair Menu

Previous Top Next

These commands provide automated fixes and updates for many of the most common configuration issues encountered when updating content or when using assets written for previous version of TRS.

The commands operate on the Outline and changes are not saved to *config.txt* until the user carries out an explicit save,

Some of these commands involve multiple modifications and need to be used with care.  The order of execution is also critical in some cases, in general you should start at the top of the menu and work downwards.

If the modifications do not give the required results you can use the Undo system to revert to the previous version of the Outline or you can restore files from backup.

**Repair Textures**
Checks the compressed texture files, texture.txt and image files in the asset and carries out a series of automatic repairs.  Following completion you may be presented with a list of redundant or unreferenced files with the option to delete them.  See the matching script function -RepairTextures for more detail.

**Repair Audio Files**
Resamples any audio files in the asset to match common frequency and bit-rate settings and re-encodes the files to an uncompressed PCM wave format.

**Update Trainz-Build**
Updates the *trainz-build* tag to match the current AssetX version and build.

**Fix Legacy KUIDs**
Converts certain outdated *kuid* references to the correct format.

**Update KUID Table**

Scans and updates the asset *kuid-table* container.  For further details of this routine see - UpdateKUIDTable.

**Create Mesh Table**
Converts pre TRS2004 assets without mesh-tables to the newer *mesh-table* format. This is only possible for relatively simple assets.  You may be prompted to delete redundant or duplicate meshes.

**Create Bogie Table**
Converts multiple *bogey, bogey-#* tags to the newer bogey table format.

**Create Thumbnail Table**
Creates a *thumbnails* container from included thumbnail and *art_icon* image files.

**Update Camera List**
Applies to kind interior only and corrects common errors in the *cameradefault* tag and the *cameralist* container.

**Update Asset-Filename**
Creates new tags to replace the obsolete *asset-filename* tag. Since *asset-filename* has multiple uses, this often involves multiple changes to the Outline.

**Update User Tags**
Updates author data for your own assets, according to the data saved in the Options/User Data dialogue.

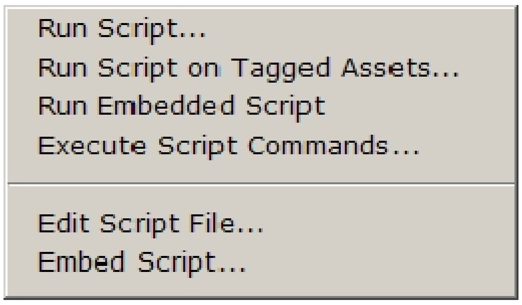
**Update Category-Era**
Consolidates multiple *category-era-#* tags into the newer *category-era* list format.

**Update Category-Region**
Consolidates multiple *category-region-#* tags into the newer *category-region* list format.

**Update Username Tags**
Updates or removes obsolete *name* tags, including localised variants.

**Delete Empty Tags**
Removes tags and containers which have no data attached.

**Delete Obsolete Tags**
Removes tags which are invalid for the current build or invalid in their current context.



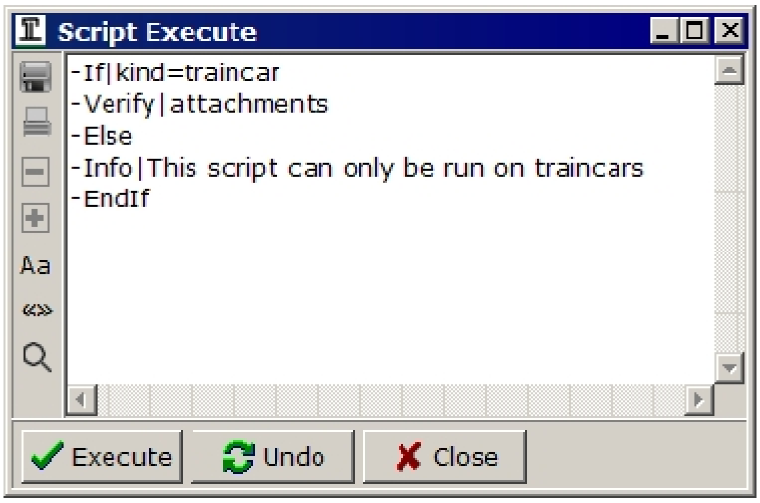
Refer to the Scripts section for more details of how to write and use Asset·X scripts.

**Run Script**
Provides a File/Open dialogue allowing the user to browse for a script file to run on the selected Outline.

**Run Script on Tagged Assets**
Provides a File/Open dialogue allowing the user to browse for a script file to run in Batch Mode on all assets tagged in the Asset List. See Batch Mode Scripts for further details.

**Run Embedded Script**
Opens a sub menu containing any embedded scripts in the current asset config. Embedded scripts each occupy a sub container at the tag address *\extensions\assetx-122285\scripts\*
See Script Files for further details.

**Execute Script Commands**
Opens a text editor which allows scripts or script extracts to be written and tested.



```
-If|kind=traincar
-Verify|attachments
-Else
-Info|This script can only be run on traincars
-EndIf
```

Pressing *Execute* will run the script in real time on the currently open asset and display the results without closing the editor. Pressing *Undo* will restore the Outline to the state that it was in before the script was run.

Once you are satisfied that the script is operating correctly you can use the *Save File* button to save it under a new name.

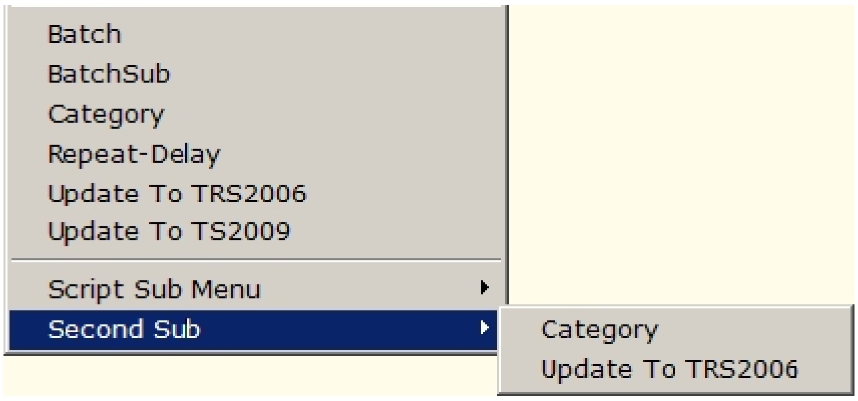
The contents of the window are saved between sessions.

Note that whilst *Undo* will revert any changes made to the Outline it will not revert the results of commands which copy or delete files. You should use the Backup facility if you plan to do this.

**Edit Script File**
Creates or opens a script file for editing in a simple popup editor.

**Embed Script**
Provides a File/Open dialogue allowing the user to select a script which will be embedded in the current asset.

**User Scripts**



The lower section of this menu contains a menu item for any script file which is found in the *...\AssetX\scripts* folder.  Selecting one of these items will cause the referenced script to be run on the current asset.

You can add sub menus to the end of this menu by creating sub folders within the *...\AssetX \scripts* folder.  Any sub folders found will create a sub menu.  Sub menus cannot be nested.

When creating user script menus AssetX ignores files and folders whose name starts with a lower case letter.  This allows you to create folders for sub routines which will not be visible in the menu.

## Tools Menu

Previous Top Next

A selection of miscellaneous tools.

**Add Groundplanes**

MODERN TRANSPORT
COMPANY
LEEDS
THE GREAT GRIMSBY
COAL SALT
AND TANNING CO LTD
MODERN TRANSPORT
COMPANY
LEEDS

Adds an ambient ground shadow to traincar assets by creating a made to measure rendered plane set at rail level.  This does not affect the asset's shadow mesh.

**Add Passenger Queue**
Adds a passenger queue to traincars.  This enables the carriage to collect and deliver passengers at stations but it does not provide any visible passengers in the carriage itself.

**PM2IM (Trawl Mode)**
Runs PEV's PM2IM utility in trawl mode on the selected asset. Refer to the PM2IM help file for further details. Please note that you require version 1.19 or later of PM2IM to interface correctly with AssetX.

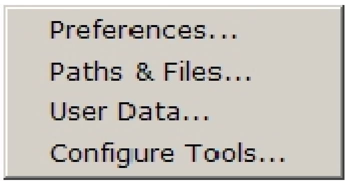
**Images2TGA (Trawl Mode)**
Runs PEV's Images2TGA utility in trawl mode on the selected asset. Refer to the Images2TGA help file for further details. Please note that you require version 1.08 or later of Images2TGA to interface correctly with Asset·X.

**Open Asset in Explorer**
Opens the current asset folder in Windows Explorer.

**User Defined Commands**
User defined commands can be added to the end of this menu. See Tool Definitions for further details.

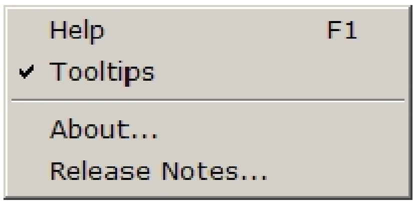
## Options & Help Menus

Previous Top Next



Provides direct access to various Options dialogues:

- Options/Preferences
- Options/Paths & Files
- Options/User Data
- Options/Tool Definition



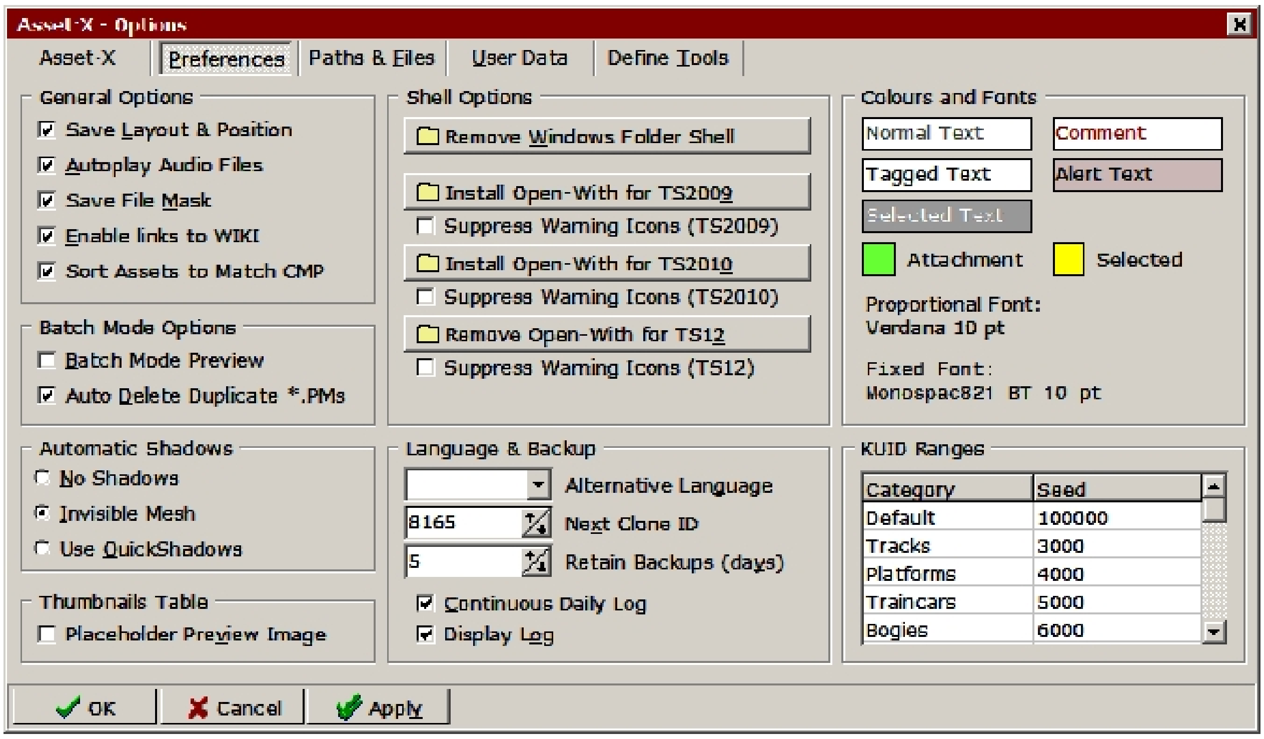
**Help**
Opens this file.

**Tooltips**
Enables or disables mouseover hints on tool buttons.

**Release Notes**
Provides access to a text file giving details of the current release including known limitations and restrictions.

**User Defined Items**
User defined commands can be added to the end of this menu. See Tool Definitions for further details.

## Preferences


Asset·X - Options

General Options
- [✓] Save Layout & Position
- [✓] Autoplay Audio Files
- [✓] Save File Mask
- [✓] Enable links to WIKI
- [✓] Sort Assets to Match CMP

Batch Mode Options
- [ ] Batch Mode Preview
- [✓] Auto Delete Duplicate *.PMs

Automatic Shadows
- ○ No Shadows
- ● Invisible Mesh
- ○ Use QuickShadows

Thumbnails Table
- [ ] Placeholder Preview Image

Shell Options
- Remove Windows Folder Shell
- Install Open-With for TS2009
- [ ] Suppress Warning Icons (TS2009)
- Install Open-With for TS2010
- [ ] Suppress Warning Icons (TS2010)
- Remove Open-With for TS12
- [ ] Suppress Warning Icons (TS12)

Language & Backup
- Alternative Language: 
- Next Clone ID: 8165
- Retain Backups (days): 5
- [✓] Continuous Daily Log
- [✓] Display Log

Colours and Fonts
- Normal Text
- Comment
- Tagged Text
- Alert Text
- Selected Text
- 🟩 Attachment
- 🟨 Selected

Proportional Font:
Verdana 10 pt

Fixed Font:
Monospac821 BT 10 pt

KUID Ranges

| Category | Seed |
|---|---|
| Default | 100000 |
| Tracks | 3000 |
| Platforms | 4000 |
| Traincars | 5000 |
| Bogies | 6000 |

### General Options

**Save Layout & Position**
Instructs Asset·X to remember its screen and window sizes and positions between sessions.

**Autoplay Audio Files**
If this is checked any audio files in the File list can be played back simply by selecting them, otherwise you will need to select *Play Audio* from the context menu.

**Save File Mask**
Checking this option causes Asset·X to remember the file filter mask in the file list between sessions.

**Enable links to WIKI**
Enables or disables direct web links from the outline right click menu to wiki pages covering the selected item.

**Sort Assets to Match CMP**
Allows the sort button in the Asset list to match the sort order used in CMP.

### Batch Mode Options

**Batch Mode Preview**
When this item is checked Asset·X will show the result of a script or a tag replication in a preview window before proceeding with the operation giving the user the opportunity to confirm that the result will be as expected. When it is unchecked the script will be applied to all of the tagged assets without further confirmation.

**Auto Delete Duplicate *.PMs**
Using the PM2IM tool will normally create a new *.IM mesh without deleting the original *.PM file. Checking this option will cause the original file to be automatically deleted when the tool is run through AssetX.

## Automatic Shadows

Instructs AssetX what to do about shadow files when creating mesh-tables for old assets:

**No Shadows**
Shadow files will not be created, in which case the user will need to deal with this issue manually.

**Invisible Mesh**
An invisible shadow will be created which will satisfy any TRS requirement for a shadow but will not actually create a shadow file in game.

**Use QuickShadows**
A shadow mesh will be created using the QuickShadows tool.

## Thumbnails

**Placeholder Preview Image**
If this option is checked the *Update Thumbnail Table* command will provide a dummy preview by copying *$screenshot$.jpg* from the *AssetX\bin* folder If the option is unchecked no thumbnails table will be produced unless the asset already contains a suitable preview image.

## Shell Options

**Windows Folder Shell**
Pressing this button will cause Asset·X to install or remove a windows shell association which will add an *Open with AssetX* option to the right click menu for folders in Windows Explorer.

**Open With in TS2009/2010/2012**
These buttons will install or remove handlers to add *Open with AssetX* commands to the right click menus in CMP for TS2009 or TS2010 respectively.

If it does not already exist, installing these options will create a new folder at *...\Auran \TS????\bin\CMPData\tools\OpenWith* and will add two files entitled *AssetX.exe* and *AssetX.ini*. Removing the handler deletes these files but will not affect any other *OpenWith* handlers.

Due to limitations in the way that CMP handles external commands AssetX will not operate correctly if it is installed into CMP by any other means.

**Suppress Warning Icons in 2009, 2010 and 2012**
Enables or disables warning icons from being displayed in TS2009 and TS2010 versions of CMP. This is a fully reversible modification and does not prevent CMP from searching for assets with warnings or from displaying details of the warning messages.

## Language & Backup

**Alternative Language**
This box can be set to any of the country codes representing localised versions of TRS. If the box is set to 'de' then the English and German versions of localised tags such as *username-de* and *description-de* will be opened at the same time whenever either tag is called up for editing.

**Next Clone ID**
The next asset ID to be used for cloned assets. If your trainz user ID is 1234 and this box reads '90000' then the next time you clone an asset the kuid will be <kuid:1234:90000>. After a clone operation the number is incremented by one.

**Retain Backups**
Specifies the maximum age of backup files which will be retained by the application's Backup system. Backups older than this will be moved to the Recycle Bin whenever Asset·X is started.

**Continuous Daily Log**
If this box is unchecked every command which creates a log file will overwrite any existing log.

If the box is checked log output will be appended to the existing log file. A new log will be created on a daily basis and old logs deleted after the number of days specified by the Retain Backups option.

**Display Log**
If this box is checked the log file will be opened and displayed whenever new content is added. The log can always be opened using the *Diagnostics/Display Log* command.

## Colours & Fonts

Use these controls to alter the appearance of the application. In some cases you might need to restart the programme to see the effects.

## KUID Ranges

This table is used to define a series of categories and seed values used to define KUID numbers to be assigned to them. Calling the Asset Menu command *Generate KUID Values* will obtain the next available kuid in each of these ranges.

You can define as many categories as you wish but one of them must be named 'Default'. This is used to reset CMP after querying it for values. The default value is used by CMP to generate numbers for routes and sessions and the default is 100000 for new installations.

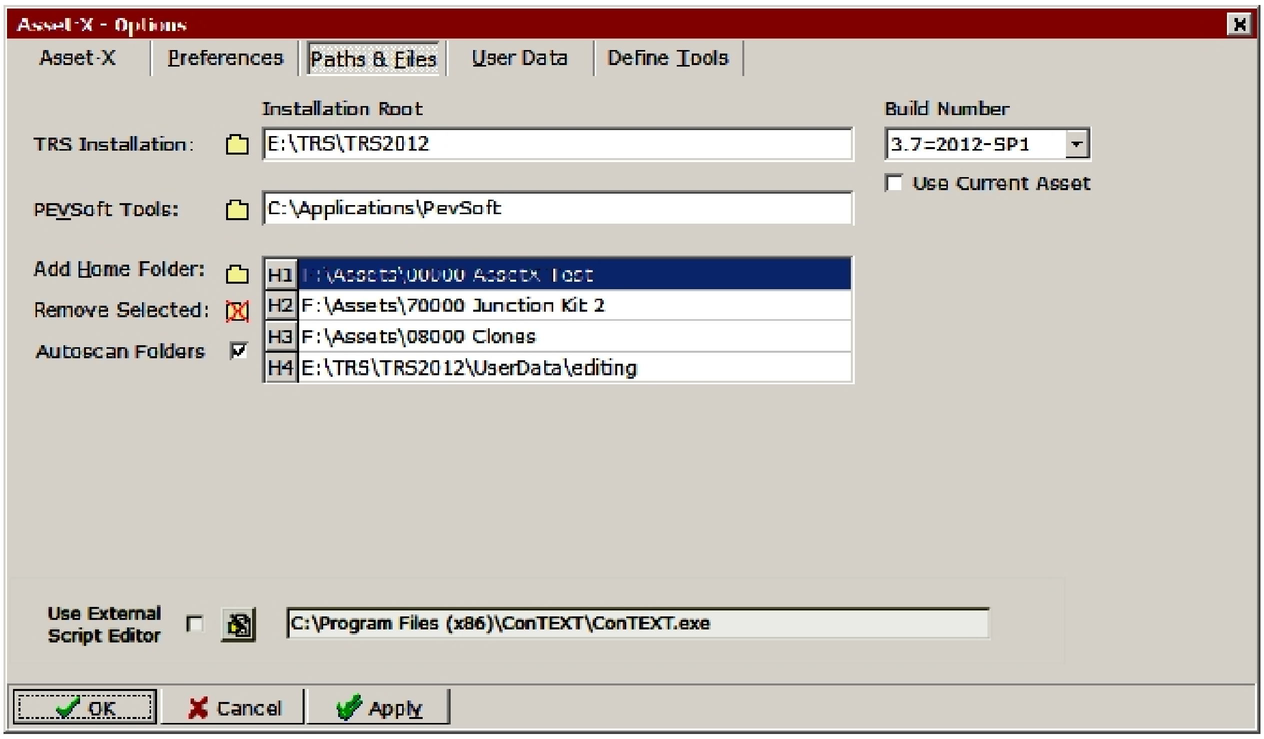
## Paths & Files

Previous Top Next

The installation dialogue has been simplified for this version and no longer provides space for multiple TRS installations. You should enter data only for your most recent installation.

You will be prompted to complete this dialogue when you install the application. You should enter the relevant paths and trainz-build correctly.

Note that the build must be entered as well as the installation path.



Installation Root

TRS Installation: E:\TRS\TRS2012

PEVSoft Tools: C:\Applications\PevSoft

Build Number: 3.7=2012-SP1

Use Current Asset [ ]

Add Home Folder:

Remove Selected:

Autoscan Folders [✓]

H1 F:\Assets\00000 AssetX Test
H2 F:\Assets\70000 Junction Kit 2
H3 F:\Assets\08000 Clones
H4 E:\TRS\TRS2012\UserData\editing

Use External Script Editor [ ] C:\Program Files (x86)\ConTEXT\ConTEXT.exe

**Installation Root**
Enter the root path of your latest version of TRS.

**Build Number**
Enter the trainz-build number for this installation, you should amend this as necessary when you apply service packs.

**Use Current Asset**
Check this box if you want AssetX to change the value of its build variable to match that of the asset which is currently displayed.

**PEVSoft**
Defines the root path of Peter Villaume's tools for TRS. An entry here is not essential but it will reduce the amount of typing necessary when you are setting Asset·X up to interface with these applications.

**Home Folders**
You can define any number of folders here. Each of these folders will be scanned at start up and any assets found will be added to the Asset List. To add folders click the Add Path button, to remove select a folder in the list box and press the *Remove Selected* button.

There is no limit to the number of folders that can be entered. The first five folders will have filter buttons assigned in the asset list allowing their contents to be displayed or hidden. See Asset List for details of this function.

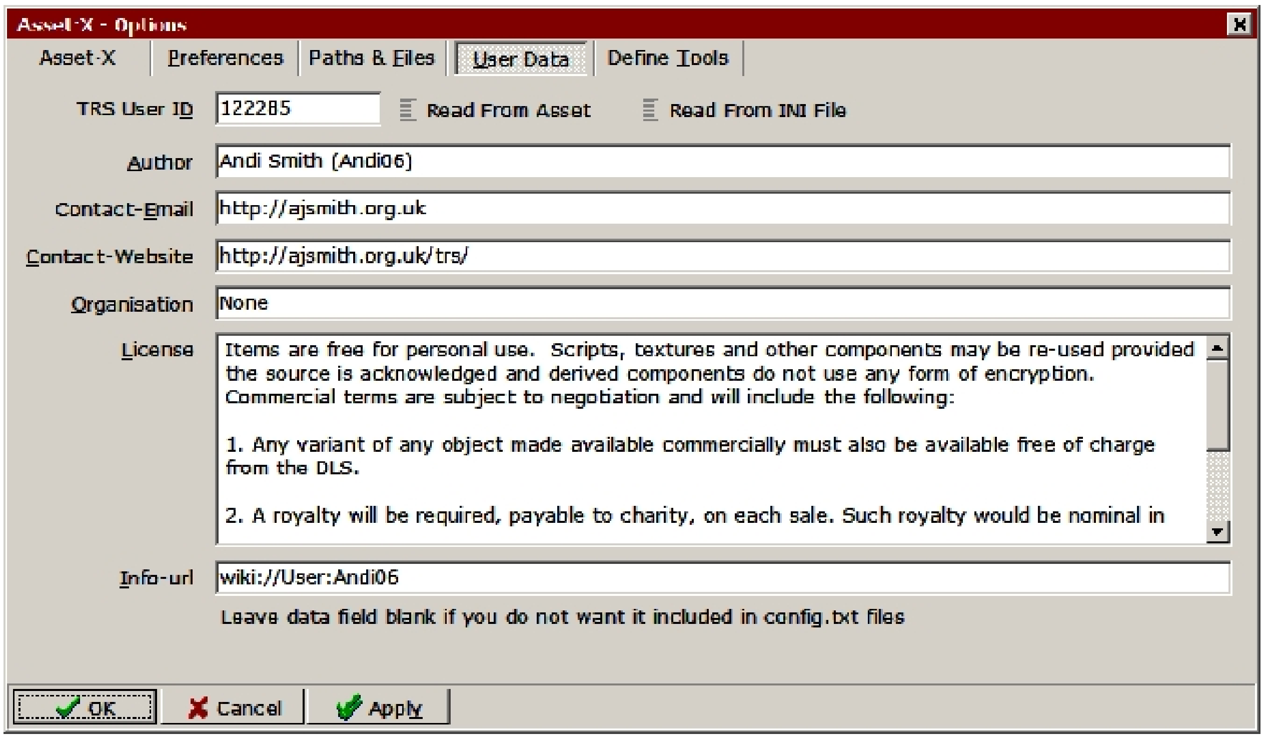
The order of the folders can be changed by dragging the grey button to the left of the folder path up or down.

**External Script Editor**
You now have the option of using an code editor such as ConTEXT for editing AssetX macro scripts. You can choose this option by clicking on the *User External Script Editor* check box. Then click on the file find button next to it and find the editor with the file open dialogue. The file path will be displayed in the External Editor box.

**Note** that we have provided Context highlighter files for AssetX *.axm scripts and *.gs Trainz scripts.

## User Data

Previous Top Next



TRS User ID: 122285

Author: Andi Smith (Andi06)

Contact-Email: http://ajsmith.org.uk

Contact-Website: http://ajsmith.org.uk/trs/

Organisation: None

License: Items are free for personal use. Scripts, textures and other components may be re-used provided the source is acknowledged and derived components do not use any form of encryption. Commercial terms are subject to negotiation and will include the following:

1. Any variant of any object made available commercially must also be available free of charge from the DLS.

2. A royalty will be required, payable to charity, on each sale. Such royalty would be nominal in

Info-url: wiki://User:Andi06

Leave data field blank if you do not want it included in config.txt files

These entries are used by AssetX to add or update user data for your own assets.

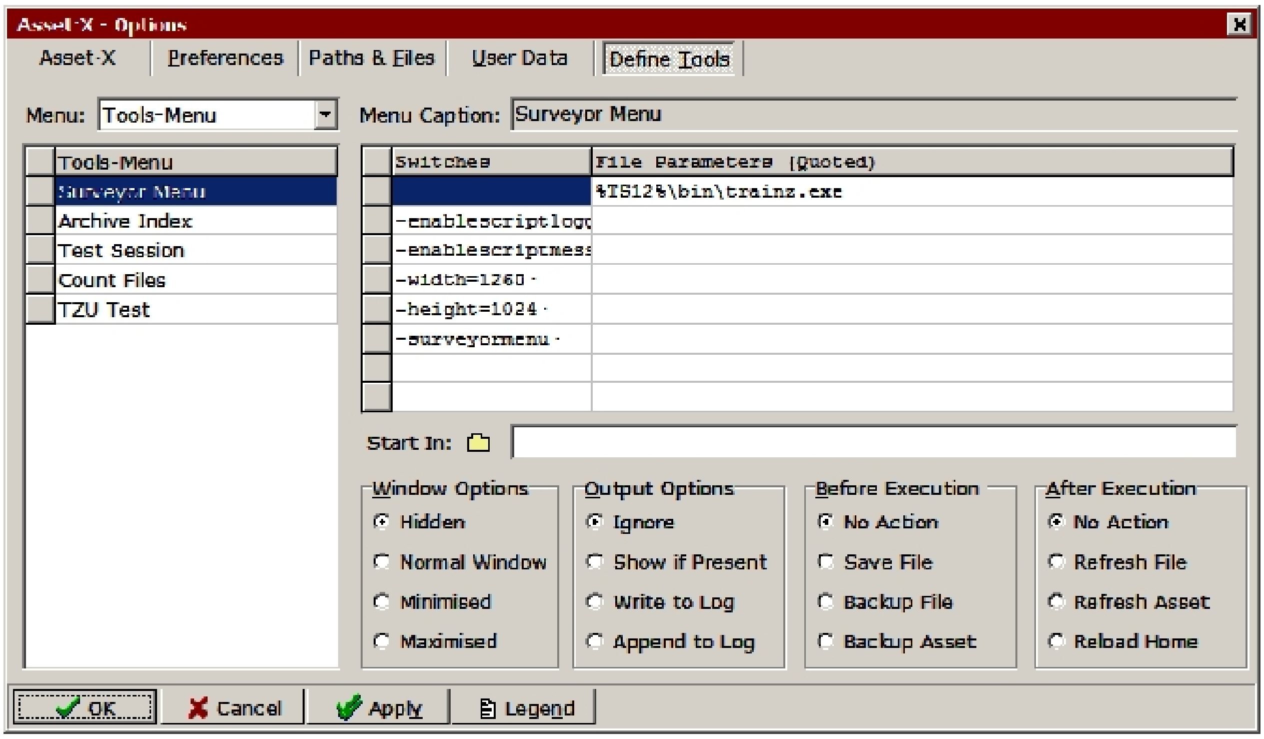
**TRS User ID**
Enter your TRS User ID here, without it you will not be able to edit all aspects of your own content.

**Read From Asset, Read from INI File**
These buttons can be used to fill the boxes from data contained in the current asset or the AssetX.ini file respectively. The values will then become the default for the menu command *Repair/Update User Tags* and the script function -UpdateUserData.

**Author, Contact-Email, Contact-Website, Organisation, License, Info-URL**
Current default values for the standard user tags. If you want to omit some or all of these tags from *config.txt*, simple leave the corresponding entries blank.

## Tool Definition

Previous Top Next



Asset·X - Options

Asset·X | Preferences | Paths & Files | User Data | Define Tools

Menu: Tools-Menu

Menu Caption: Surveyor Menu

| Tools-Menu |
|---|
| Surveyor Menu |
| Archive Index |
| Test Session |
| Count Files |
| TZU Test |

| Switches | File Parameters (Quoted) |
|---|---|
| | %TS12%\bin\trainz.exe |
| -enablescriptlogg | |
| -enablescriptmess | |
| -width=1260 · | |
| -height=1024 · | |
| -surveyormenu · | |
| | |
| | |

Start In:

| Window Options | Output Options | Before Execution | After Execution |
|---|---|---|---|
| [✓ Hidden] | [✓ Ignore] | [✓ No Action] | [✓ No Action] |
| Normal Window | Show if Present | Save File | Refresh File |
| Minimised | Write to Log | Backup File | Refresh Asset |
| Maximised | Append to Log | Backup Asset | Reload Home |

OK | Cancel | Apply | Legend

**Configuring Custom Tools**

To define a tool you will need to enter a Menu Caption, the necessary command line data, and various options which specify how the tool should be run and what should be done by Asset·X before and after execution.

**Menu Selector**
Allows selection of various categories of command which determine the means of accessing tools, as described in the table below.

| Category | Means of Access |
|---|---|
| Tools-Menu | Commands defined in this section will be added to the bottom of the main menu Tools section, separated by a spacer bar. |
| Documents | These commands will be included in a Documents submenu of the main menu Tools section. If this section is empty the Documents submenu will not appear. |
| Help | Commands defined in this section will be added to the bottom of the main menu Help section, separated by a spacer bar. |
| CustomA, CustomB, CustomC, CustomD, CustomE | These can be entirely user-defined. The menu title can be edited and the user menus will appear on the main menu bar between Tools and Options. If any of the sections are empty the corresponding menu will not appear. |
| *.* | These will appear for every file in the right-click context menu in the file list |

| | |
|---|---|
| | pane. There is a limit of 6 items. |
| *.ext | These will also appear in the File List Context menu but only if the menu title matches the selected file specification. Again there is a limit of 6 items per filetype. These menus can be removed, or new ones added by manually editing the INI file. |

Selecting one of these items will open the current contents of that menu for editing.

**Menu Editor**

The menu editor is a list of items for the selected section. Items can be reordered by dragging the buttons in the left hand column up or down the screen. The grid itself is read-only and can be edited by means of a right click context menu with these command options:

| Command | Action |
|---|---|
| Edit Menu Title: | This applies to the custom menu items only and allows you to change the text which will appear in the main menu bar from CustomA to a Title of your own choice. |
| Clear Menu: | Removes all of the current items and (except for Help and Tools) will prevent the menu categories from appearing in the application. |
| Insert Item: | Creates a new menu command above the selected item. |
| Rename Item: | Allows the selected command to be renamed. |
| Insert Divider: | Inserts a separator bar above the selected item. This is not available for the filetype sections. |
| Cut, Copy & Paste: | Copies or Pastes the menu item, including the entire command definition. |
| Delete: | Removes the selected item. |

**Menu Caption:** Displays the caption which will be used to identify the current tool.

**Switches & File Parameters**
The main part of the screen contains a table of switches and parameters. The left hand column is used to enter command line switches, such as *-compile* or *inFile*. These will be sent to the output unchanged.

The right hand column is used for filenames of executables which are to be called or for html links or documents. These will always be wrapped in quotes before processing.

Use the special command sequence *RunAXM path\scriptfile.axm* to assign an Asset·X script to a menu.

You can build up your command by entering text, macros or path and filenames into the grid cells. It is not necessary to fill every cell.

It is usually important to get the right number of spaces in the right place. Within the grid, space characters will show up as a small dot to help you to see where they will be added, but they will be appended to the command line as a space character.

When the tool is parsed by the game the left and right hand cells are concatenated without any intervening spaces (unless you have typed spaces into the cell) Empty cells are ignored. Each row of the grid is then added on in turn with a space inserted between the rows.

See Tool Examples for further details of options.

**Start in:**
Use the 'Start in Folder' box if the tool needs to be run in a specific working directory. This is rarely necessary.

**Window Options:**
These options define whether the Tool will be run in a normal, *minimised* or *maximised* window. You can also run tools in *hidden* windows but you should not do this until you are absolutely sure that they are working correctly.

**Log Options:**
Various alternative methods for dealing with text output from tools. This can be ignored, written into the Text Viewer window or saved to a log file which will be opened when the process has completed. Using *Append to Log* will allow you to keep a continuous record of command output in a text file *...\AssetX\log\AssetX.log*. Not all programmes supply any text output.

**Before Execution:**
This panel allows you to specify options for saving files and backing up before the tools are executed.

**After Execution:**
Provides options for post processing, these are normally used to refresh files or assets which may have changed as a result of running the tool.

**Legend:**
Provides a summary of available shortcut macros which you can use to keep your tools portable (which will enable you to share them with other users who may have their programmes in different disc locations)

**Right Click Context Menu**
You can access this menu by pressing the right mouse button when the cursor is over the grid, these are the available commands:

| Command | Action |
|---|---|
| Clear Parameters | Clears the entire parameter table. |
| Cut,<br>Copy,<br>Paste, | Standard Windows clipboard actions. |

| | |
|---|---|
| Delete,<br>Select All | |
| Macros | A sub menu which enables you to add any of the Asset·X predefined macros. |
| Browse for File,<br>Browse for Folder | Enables you to select a file or folder using a standard Windows dialogue. The selected item will be placed into the current cell of the grid. |
| Edit Command Script | If the current cell refers to a command or batch file (**.CMD* or **.BAT*) this menu option will open that file for editing. If the command or batch file doesn't exist it will be created as an empty file, you will need to ensure that your macro points to the correct path. |
| Preview Macro... | Evaluates any macros in the current grid cell and displays the result in a message box. |
| Preview Command Line... | Evaluates the entire command line and displays the result in a message box. |

## Tool Examples

Previous Top Next

The following examples demonstrate how to configure various kinds of user defined tool for use with AssetX, including how to enter parameters and which settings to use.  Some tools are predefined in the default installation, you can study these as additional examples.

**Simple Executables**
The simplest form of tool definition, which simply calls the executable file without any additional parameters.

| Switches | File Parameters (Quoted) |
|---|---|
| | %systemroot%\system32\calc.exe |
| | |
| Show Window | Normal Window |
| Capture Output | Ignore |
| Before Action | None |
| After Action | None |

**Note:** On 64 bit installations Windows redirects calls to the *...\Windows\system32\* folder to another folder *...\Windows\sysWOW64\* where 64 bit versions of the operating system utilities are stored.  In some cases a 64 bit version is not available and this will cause the tool to fail, one example being the built-in screen capture utility SnippingTool.exe.  The syntax *%systemroot%\SysNative\SnippingTool.exe* should be used to call this utility.

**External Editors**

An external image editor.  The executable is called with the current file as a parameter. Since this action may result in changes to the image, the current file is refreshed when the command returns.

| Switches | File Parameters (Quoted) |
|---|---|
| | %programfiles%\PSP\PSP.exe |
| | %pfe% |
| | |
| Show Window | Normal Window |
| Capture Output | Ignore |
| Before Action | None |
| After Action | Refresh Selected File |

**Executables with Multiple Parameters**
This tool calls *trainzutil.exe* to compile a script and uses various parameter strings to set include file directories and paths for temporary files. Finally it passes the path and filename of the current file in the parameter *%pfe%*.

Before Action is set to save the file before compiling.  *Trainzutil* is set up to run in a minimised window and to show its output in a message box once the command has completed running.

| Switches | File Parameters (Quoted) |
|---|---|
| | %trainz%\bin\trainzutil.exe |
| -i | %trainz%\scripts |
| -b | %temp% |
| -p | %temp% |
| | %pfe% |
| | |
| Show Window | Minimised |
| Capture Output | Show in Message Box |
| Before Action | Save Selected File |
| After Action | None |

**Documents**
Assuming that you have an appropriate viewer application installed all that is necessary to open a document is to insert the document's path and filename.

| Switches | File Parameters (Quoted) |
|---|---|
| | D:\Software\TRS\Documents\CCGTC.pdf |
| | |
| Show Window | Normal Window |
| Capture Output | Ignore |
| Before Action | None |
| After Action | None |

**Internet Links**
Similarly for internet links simply enter the web address. The link will be opened in your default browser.  The http:// prefix is required.

| Switches | File Parameters (Quoted) |
|---|---|

| | |
|---|---|
| | http://forums.auran.com/trainz/index.php |
| | |
| Show Window | Normal Window |
| Capture Output | Ignore |
| Before Action | None |
| After Action | None |

**Assigning a Script to a Menu Command**
This syntax allows you to allocate a customised script to a menu item. You must provide the full path and filename of the script (which can be anywhere on disc) as the file parameter.

| Switches | File Parameters (Quoted) |
|---|---|
| RunAXM | %scripts%\sub\Clone & Update.axm |
| | |
| Show Window | Normal Window |
| Capture Output | Ignore (script output will appear in the Alert bar as usual) |
| Before Action | None |
| After Action | None |

As an alternative to referencing a script file you can type script commands directly into the File Parameters column. The data grid does not allow vertical scrolling so you will be limited by the number of lines available. Here is a simple example to count the number of files in the current asset:-

| Switches | File Parameters (Quoted) |
|---|---|
| RunAXM | -Foreach\|File\|*.* |
| | -EndFor |
| | -MessageBox\|Asset·has·^T·files |
| | |
| Show Window | Normal Window |
| Capture Output | Ignore (script output will appear in the Alert bar as usual) |
| Before Action | None |
| After Action | None |

## Backups

Previous Top Next

Backup facilities provided by the application are intended to provide a temporary location for safety copies of assets that you are currently working with. You should not rely on them as a substitute for a proper backup regime.

Asset·X has facilities for backing up individual files or entire assets. The backup can be initiated by selecting *File/Backup File* or *File/Backup Asset* from the main menu or you can use the script functions -BackupFile and -BackupAsset. Restoring a backup can only be performed manually, using *File/Restore File* or *File/Restore Asset*.

Backing up a file or asset results in the selected item being copied to the backup folder. The files are not compressed or altered in any way but they are renamed to reflect the original source and the date and time of the backup. The naming conventions used are:

| Item | Naming Convention | |
|---|---|---|
| Assets: | YYYYMMDDHHMMSS¶Drive¦¦Path¶ | |
| Files: | YYYYMMDDHHMMSS¶Drive¦¦Path¶Filename.ext | |
| where | YYYYMMDD<br>HHMMSS<br>Drive<br>Path<br>Filename.ext | Date of backup (Year, month & day)<br>Time of Backup (Hours, minutes & seconds)<br>Original Disc Drive<br>Original Path<br>Original Filename |

When files or assets are restored the following procedure is followed:

1. If a backup is available then the current file or asset is deleted

2. The latest available backup is copied back to the original location and its original name restored.

3. The backup itself is then deleted, any older backups are retained.

Repeated calls to *File/Restore File* or *File/Restore Asset* will therefore reinstate successively older backups as long as they remain available to the programme.

The option to Retain Backups (see Options/Preferences) defines the maximum number of days for which backups will be retained. Any backups which are older than this are moved from the backup folder to the Windows recycle bin during Asset·X loading.

If you need to restore backups from the Recycle Bin you should do so while Asset·X is running and you should use them within the current session, this is because the backup folder is purged of outdated files during programme initialisation.

The default backup location, which is also used to store undo files, is *...\AssetX\backups*. This can be changed by editing the INI file entry *Paths,Backup* (which is empty by default) to define an alternative path. If you modify this location your new path should not be in any of the TRS *custom* or *editing* folders.

## Scripts

Previous Top Next

Asset·X includes a simple script language to allow users to build asset manipulation tools which can be tailored to suit their particular needs. Scripts can be saved as plain text files or embedded into the *config.txt* of an asset, in either case scripts operate on the tree view of the currently loaded asset.

Because *config.txt* is displayed in a tree-view format the location of any item can be specified in folder\file format in the same way that the location of a file on disc is displayed. This notation is referred to as the data's tag address.

For example the mesh tag in an asset's default mesh would have the tag address: *config.txt\mesh-table\default\mesh.*

An address cannot include tabs or spaces, starts from the first level at *config.txt*, and must use backslashes as a path separator. To cut down on typing you may omit the starting text *config.txt* or use a leading backslash instead. When this syntax is used the tag address above can be abbreviated to: *\mesh-table\default\mesh* or just *mesh-table\default\mesh*

Scripts are read and executed one line at a time. Any leading spaces or tab characters are ignored and any empty line or any line starting with a semi colon ( ; ) is treated as a comment and ignored by the interpreter.

From version 3 comments may be appended as an additional parameter to most commands.

Script commands always start with a hyphen but are not case sensitive. Mixed upper and lowercase strings are easier for us humans to understand. The script interpreter however, converts everything to lowercase before processing.

Empty lines, comments and any amount of white space are permitted but any other character at the start of a line will cause the script to fail.

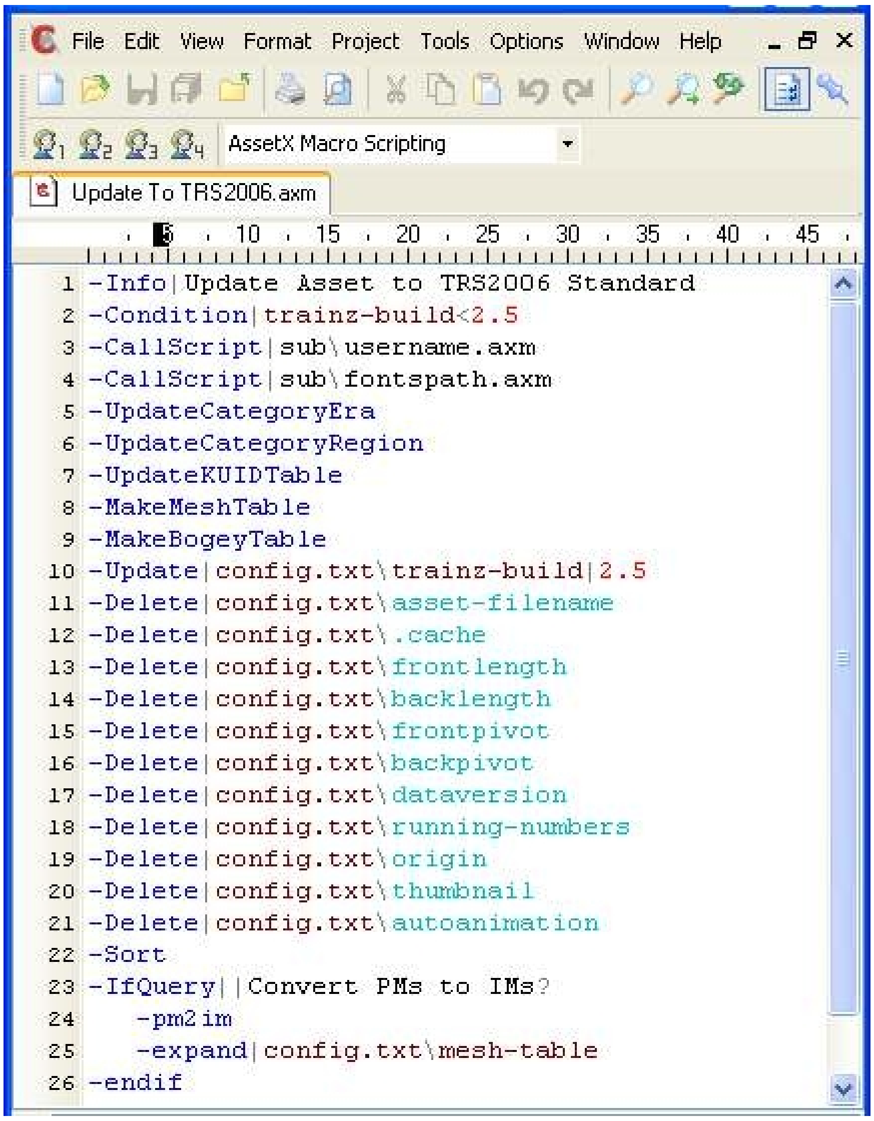
Asset·X scripts support several variations of *If/Else/EndIf* and simple *Case* and *ForEach* functions for decision making and includes support for user input via message boxes, there is no provision of while or repeat contructs.

Script commands should not include single or double quotation marks.

*ConTEXT* is a good freeware editor available on the internet. Its highlighter parsing is useful for checking script files.

A ConTEXT editor highlighter file for **.axm* files is provided to assist with writing scripts. Additionally a highlighter for Trainz Script is provided. The files, "assetx script.chl" and "trainz_script.chl" can be found in the AssetX bin folder. Copy these to the ConTEXT highlighter folder.

The image below shows an AssetX Script opened in ConTEXT.



```
-Info|Update Asset to TRS2006 Standard
-Condition|trainz-build<2.5
-CallScript|sub\username.axm
-CallScript|sub\fontspath.axm
-UpdateCategoryEra
-UpdateCategoryRegion
-UpdateKUIDTable
-MakeMeshTable
-MakeBogeyTable
-Update|config.txt\trainz-build|2.5
-Delete|config.txt\asset-filename
-Delete|config.txt\.cache
-Delete|config.txt\frontlength
-Delete|config.txt\backlength
-Delete|config.txt\frontpivot
-Delete|config.txt\backpivot
-Delete|config.txt\dataversion
-Delete|config.txt\running-numbers
-Delete|config.txt\origin
-Delete|config.txt\thumbnail
-Delete|config.txt\autoanimation
-Sort
-IfQuery||Convert PMs to IMs?
   -pm2im
   -expand|config.txt\mesh-table
-endif
```

## Script Files

Previous Top Next

There are two ways to save scripts for future use:

**Script Files**
Scripts should be plain text files saved with the extension *.axm*. If a script is saved into the *AssetX\scripts* folder it's name (extension omitted) will be added to the main scripts menu and can be executed by selecting the matching item from the menu. Scripts saved elsewhere can be executed using the *Scripts\Open Script...* item in the same menu.

Debugging of scripts must be performed within AssetX, you might find the *Scripts/Execute Script Commands* menu item useful in this context. To assist with debugging an interpreter syntax message will be displayed in the configuration pane window showing any instruction that could not be executed with the name of the file and its line number. Sub-routines should be debugged as stand-alone scripts before integrating them using the *-CallScript* instruction.

When using *-Callscript* you must ensure that all script file names are unique, even if the files are in separate folders, this is to remove any possibility of recursive script calls.

**Embedded Scripts**
Scripts can also be stored for use in specific assets by embedding them into the asset's *config.txt* file at tag address *\extensions\assetx-122285\scripts\scriptname*, where *scriptname* is the name of the script (space characters are not permitted in embedded script names).

In this case the script can be executed via the *Scripts\Run Embedded Script...* menu item or by a Right Click on *scriptname*. The format of embedded scripts is slightly different since it includes line numbers and quote characters around the commands (these items are stripped out before processing). The easiest way to embed a script is to create and test it as a text file and then use the main menu command *Scripts\Embed Script...* to add it to an asset's config.txt file.

## Parameters & Data

Previous Top Next

Some commands require additional parameters or data. A pipe character ( | ), normally shifted backslash on English language keyboards, serves as a separator between the command itself and any following fields.

So for example if you wish to delete a mesh reference in the mesh-table the script instruction would be: *-Delete|config.txt\mesh-table\default\mesh*

Now suppose that you wish to set a new value for a tag. Again using the mesh-table data, you might wish to amend the auto-create value:
*-Update|config.txt\mesh-table\default\auto-create|1*

Again note the vertical pipe symbol between the address and its new data. The Update instruction will change an existing tag or create a new tag if it does not already exist at the address specified. Hence it is used to make new entries as well as to alter existing tags.

Note that the Asset·X script language does not use or recognise either single or double quotation marks.

Parameters which include filenames and paths and certain other constants can make use of the wildcards (* and ?) and of Asset·X macros which are shortcuts wrapped in % signs:

This command sequence for instance would delete all of the current backup files.
*-DeleteFiles|%backup%*.**

## Operators

Previous Top Next

To allow branches in execution some commands require conditions as parameters.  A condition is an expression which compares the value of specific tags with a test value and returns true if the comparison is met and false otherwise. The following tests are available in Asset·X scripts.

Most operators work with the value of a tag, since container labels do not have a value only the @ operator can be used with containers.

| Operator | Example | Description |
|---|---|---|

| | | |
|---|---|---|
| = | *If\|trainzbuild=2.4* | Returns true if the tag trainz-build has the value 2.4. |
| ! | *–If\|trainz-build!2.4* | Returns true if trainz-build is not equal to 2.4. |
| < | *–If\|trainz-build<2.4* | Returns true if trainz-build is earlier than 2.4. |
| > | *If\|trainzbuild>2.4* | Returns true if trainz-build is later than 2.4. |
| @ | *-If\|trainz-build@* | Returns true if trainz-build exists. This operator does not require a comparison. This operator works equally well with containers as well as tags. |
| # | *-If\|trainz-build#* | Returns true if trainz-build exists and has a numeric value. This operator does not require a comparison. |
| $ | *-If\|trainz-build$* | Returns true if trainz-build exists and has a non numeric value. For the purposes of this operator float and integer lists are treated as non numeric. This operator does not require a comparison. |

The interpreter always attempts a numeric comparison first. If one or both of the operands cannot be converted to a number an alphabetic comparison is carried out.

If the comparison argument is omitted it will be interpreted as nil. This allows you to test for the existence of a tag by omitting the second argument:

*-If|trainz-build!* returns true if trainz-build exists (if it is not equal to nil).

*-If|trainz-build=* returns true if the tag does not exist (if it is equal to nil).

You can use *–Else* to implement negative conditions:

*-If|kind=traincar*
*-Else*
- commands placed here will only be executed if the asset is not a traincar.
*-EndIf*

Alternatively you can use *-IfNot* to achieve the same result in a different order:

*-IfNot|kind=traincar*
- commands placed here will only be executed if the asset is not a traincar.
*-Else*
*-EndIf*

Similar constructions can be used to perform tasks where a tag is greater than or equal to ( >= )

*-If|trainz-build<3.0*
*-Else*
- commands placed here will we executed if trainz-build is equal to or later than 3.0
*-EndIf*

The # operator can be used to check whether a value exists and contains a numeric value. The $ operator returns true for non numerics, including lists and can be used in a similar way.

*-If|repeat-delay#*
*-Info|repeat-delay exists and contains a number*
*-Else*
*-Info|repeat-delay does not exist or is a string (or a numeric list)*
*-EndIf*

You can use *-IfNot* to reverse the interpretation of a test:

*-IfNot|mesh-table@*
- commands placed here will only operate if a mesh-table does not exist
*-EndIf*

**String Wild Cards**
Most comparisons for string tests the use of the * and ? wild cards.

* is used to represent zero or more characters.

? is used to represent one character only

- *-If|username=fred** will test TRUE if the username STARTS with "fred".
- *-If|username=*fred* will test TRUE if the username ENDS with "fred".
- *-If|username=*fred** will test TRUE if the username CONTAINS the string "fred".
- *-If|(exists)????e.tga* will test TRUE if the current asset CONTAINS any tga files with five character names ending in e.

In some cases wildcards are not permitted, the script interpreter will issue an error message where this is the case.

-IfMatches allows wildcard tests on Script Variables as well as tag values:

*-IfMatches|?rain*|kind*

returns true if kind is a traincar

*-IfMatches|*York*|^0*

returns true if the variable ^0 contains the text 'New York City'  (or just 'York')

**Notes**

Wildcard behaviour has changed for the current version of AssetX.  In most cases this will not affect existing scripts but there may be exceptions.

## File Searches

Previous Top Next

The operator *(exists)* is used to test for the existence of a file or files on disc. The file specification can include * or ? wildcards and can also include  Asset·X Macros.

If the file specification evaluates to include a drive specifier then the computer's file system will be searched, if not the search will be limited to the current asset folder.

Note the underlined special syntax which allows you to search sub-folders in an asset.

| Example | Scope of Search |
|---|---|
| *-If\|(exists)C:\TRS.BAT* | Searches for TRS.BAT in the root directory of drive C. |
| *-If\|(exists)C:\*.BAT* | Any *.BAT file in the root of drive C. |
| *-If\|(exists)%bin%\*.AXM* | Any *.AXM file in the *...\AssetX\bin* folder. (*%bin%* includes a drive specification). |
| *-If\|(exists)*.WAV* | Any *.WAV file <u>anywhere in the asset</u>. |
| *-If\|(exists)\*.WAV* | Any *.WAV file in the root folder of the asset. |
| *-If\|(exists)\SOUND\*.WAV* | Any *.WAV file in the asset's \SOUND folder. |

A similar syntax applies to other commands which handle files, for instance:
*-DeleteFiles|*.texture* deletes all binary texture files in the current asset, including sub-folders.

You can also check for the non existence of a file using this construct:

*-If|(exists)config.txt*
*-Else*
*-Info|This asset will not work*
*-EndIf*

## Inheritance

Previous Top Next

The operator *(inherits)* checks the current asset kind and returns true if the asset includes support for the appropriate set of tags. This can be used to determine whether the current asset supports scripting or product queues for instance.

| Example | Description |
|---|---|
| *-If\|(inherits)scripted-asset* | Returns true if the asset has scripting |

| | |
|---|---|
| | capabilities. |
| *-If\|(inherits)mesh-asset* | Returns true if the asset includes mesh-table support. |
| *-If\|(inherits)industry* | Returns true if the asset supports processes. |

A full definition of asset ancestry parameters can be found in the data file *...\AssetX\bin\tags.xml*. You can also view inheritance data using the main menu command *Asset/List Available Tags*. Some of the most useful parameters are listed below:

| Inheritance Specifier | Description |
|---|---|
| Scripted-Asset | Asset can be scripted. |
| Mesh-Asset | Asset has mesh-table support. |
| Map-Object | Asset supports product queues and smoke blocks. |
| SceneryWithTrack | Asset supports attached track and scenery triggers. |
| Industry | Asset supports processes. |

## Script Variables

Previous Top Next

**Script Variables**

Script variables provide a mechanism to pass data to scripts.

The range and type of the variables which are available are set out in the table below. It is also possible for a script author to manipulate variables using the Save, Restore, and Combine functions.

To make use of a variable you insert a circumflex (**^**) symbol followed by the appropriate single numeric or alphabetic character taken from the reference table below. The script engine will replace this character pair with the contents of the variable, if the variable is empty it will simply be removed.

**Examples**

```
-Foreach|File|*.im
  -Update|mesh-table\^I\mesh|^F
  -Update|mesh-table\^I\auto-create|1
  -If|(exists)^f.kin
    -Update|mesh-table\^I\anim|^f.kin
  -EndIf
-EndFor
```

The code above will take all the *.im meshes found in the current asset and create a mesh table entry with mesh and auto-create tags for each file, animation tags will also be added if a corresponding *.kin file exists.

*-If|kuid=*
*-Update|kuid|^K*
*-EndIf*

If no kuid tag exists this code will add the tag using the next available value specified in Options/Preferences.

*-Save|0|username*
*-Update|username-^L|^0 (to be translated)*

Assuming that you have set '*de*' as an alternative language, this copies the value of *username* into *username-de* with a note appended.

**Table of Available Script Variables**

| Variable | Contents | Limitations |
|---|---|---|
| ^0 .. ^9 | The current contents of the AssetX memories. These are created and manipulated by Save, Combine, Compute, INIRead, Tokens and Restore | These values will be empty until a value has been entered one of the functions listed. |
| ^D | The value of the current item path including trailing path delimiter. This variable returns a path which is relative to the file system and includes the drive specifier. | Only available within a ForEach block. |
| ^d | The value of the current item path including trailing path delimiter. This variable returns an asset relative path. | Only available within a ForEach block. |
| ^F | The value of the current item as a filename including an asset relative path if necessary. | Only available within a ForEach block. |
| ^f | The same as ^F but with the file extension omitted. | Only available within a ForEach block. |
| ^I | The index of the current list pointer. | Only available within a ForEach block. |
| ^K | Next available clone KUID | Empty unless set via the Options pages. |
| ^L | Alternative language | Empty unless set via the Options pages. |

| | | |
|---|---|---|
| ^P | The name of the last point read using the PointRead function. | Empty if PointRead has not been called in the current script. |
| ^T | The number of items in the last list generated by the Foreach command. | Empty unless Foreach has been called in the current script. |
| ^V | The value of the current list item, the information returned will depend on the nature of the Foreach parameters. | Only available within a Foreach block. |
| ^X, ^Y, ^Z | The X, Y and Z position values of the last point read using the PointRead function. | Must be entered in upper case characters. |
| ^x, ^y, ^z | The x, y and z rotation values of the last point read using the PointRead function. | Must be entered in lower case characters. |

**Notes**

Variables are evaluated before script commands are called. It is therefore possible to use them in almost any parameter to almost any function. The script author must ensure that the variable will evaluate to the correct data type for the function being used.

If the variable being called has no data at the point that the call is made it will be removed from the script parameter. This may result in an empty parameter where the function being called requires valid data. The result of this will depend on the function being called.

Each script has its own saved data and this is not passed on to any nested script invoked using the CallScript mechanism. If you need to pass data between scripts you should use the SetEnv function to create environment variables.

Strings held in any of the script variable memory locations are limited to 255 characters in length. Any additional data will be truncated.

Script variables cannot be used in the parameters of the Save or Restore functions.

**See Also**

-Combine
-Restore
-Save
-SetEnv

## Batch Mode Scripts

Previous Top Next

Scripts can be run in batch mode, that is a single script can be executed on a selection of assets in a single command.  To do this:

- Tag any number of assets in the asset list pane, the current asset can be included.
- From the main menu select *Scripts/Run Script on Tagged Assets*, this will present a file dialogue allowing you to select the script file to execute.  Double click on the script file or select it and press [*Open*]
- You will then see a Preview dialogue showing the results of running the script on the first asset in your list.  This is similar to the Preview used in the Replicate command.  The preview can be turned off via *Options/Preferences*.
- The *Preview Next File* checkbox is used to control whether or not  subsequent assets in your selection will be presented for Preview.  If you uncheck this box subsequent assets will be processed with being displayed in the Preview.
- If you are happy with the results, press *Accept* and the changes which the script has made will be saved to file, if not press *Reject* to prevent changes to the current asset. If *Preview Next File* is checked the next file will be previewed for a decision, if it is unchecked your decision will be applied to all the remaining files.
- When Asset·X has completed the batch run a report window will be displayed showing the time taken for the command and a list of assets which have been processed together with any messages generated by the script for each asset. You can save this log to disc if you wish.
- Assets with faulty *config.txt* files or assets with missing or faulty *kind* tags cannot be processed in Batch Mode.  Where such assets are found they will be skipped and the log will be annotated to show that they have not been processed.

Asset·X will backup the *config.txt* of each tagged asset which enables you to select the asset and restore its previous state if anything has gone wrong.  If you are running scripts on large numbers of assets you might want to reduce the number of files in the backup folder by reducing the number of days they are retained.  You can do this via the Options/ Preferences dialogue.

Certain script methods behave slightly differently in batch mode.  For details see the individual pages for each command.

The Undo system does not operate during batch mode scripts.

## Run Once Scripts. (advanced users only)

Previous Top Next

Asset·X uses an INI file to store its configuration data. Upgrades and Tool additions that add to the INI file can be done by with an Asset·X script named "RunOnce.axm". Such scripts use special scripting commands to update the INI file, in addition to the normal script commands listed in this help file.

When placed in the Asset·X scripts folder, a RunOnce.axm script will be executed when Asset·X is launched and then the script is deleted. Hence it only runs once.

Most users will be unaware that all Asset·X upgrades and tool implementations have included RunOnce.axm scripts. They are a painless way to alter or add features.

The script INI file instruction are:-

| Instruction | Purpose |
|---|---|

| | |
|---|---|
| -INIAdd\|Section\|Key\|Value | Adds Key=Value to Section, creating Section if necessary |
| -INIAdd\|Section\|\|Value | (empty second parameter) Adds Value to the end of a numbered list, creating a new numeric Key, Section is created if necessary. Unfortunately there is no way to tell if a Section is treated as a numbered list without access to the code. |
| -INIAdd\|Section | Creates a new empty Section if it doesn't already exist |
| -INIDelete\|Section\|Key | Deletes Key |
| -INIDelete\|Section\|* | Deletes all Keys but keeps the Section header (clears the section) |
| -INIDelete\|Section | Deletes all Keys and removes the Section header |

Example
*-INIAdd¦*jpg¦¦Add Watermark¦¦"%PEV%\Images2TGA\Images2TGA.exe"¶-watermark·-close·"%pfe%"¶watermark_CRG1.tga¦1¦0¦0¦0¦*

The data above can be copy-pasted from the AssetX.INI file after getting the tool to work as desired.

## Script Command Listing

Previous Top Next

An alphabetic listing of AXM script instructions.

| Script Instruction | Purpose |
|---|---|
| -BackupAsset | Create a backup of the current or specified asset. |
| -BackupFile | Create a backup of the current config, or of any specified file. |
| -Break | Terminates execution of a -Foreach loop. |
| -CallScript | Call another script as a sub-routine, no return values are supported. |
| -Case | Transfers script execution to a new line based on selection from a list of options. |
| -Category | Adds and removes items from tags such as category-keyword. |
| -Clear | Removes empty and invalid tags and containers from config.txt |
| -Condition | Displays a message requiring a Yes/No response. Yes allows execution to continue, no terminates the script. |

| | |
|---|---|
| -Combine | Used in conjunction with -Save and -Restore, this command allows string concatenation of saved variables. |
| -Comment | Marks a tag or container as a comment. |
| -CopyFile | Copies a file from a specified source to a specified destination, renaming the file if appropriate. |
| -CopyFromAsset | Used in an *-IfAsset/-EndIf* block.  Copies files from another asset if that asset is currently loaded in Asset·X |
| -Delete | Deletes the specified tag or container. |
| -DeleteFiles | Deletes the specified files from disc. |
| -EndCase | Terminates a *-Case* statement. |
| -EndFor | Terminates a *-ForEach* loop. |
| -EndSelect | Terminates a *-Select* statement. |
| -Exchange | Searches for an existing tag with a specified value.  If the tag is found and the value matches it will be deleted and replaced with an alternative. |
| -Exit | Unconditionally terminates script execution. |
| -Expand | Expands the tree view of a container to display its contents. |
| -FixKUIDs | Updates legacy Auran KUID entries, normally found in content which predates TRS2004, to standard format. |
| -ForEach | Carries out instructions for every item in a list defined by the function parameters. |
| -If -IfNot -Else -Endif | Condition control statements. If the condition is true the following code will be executed up to the next *-Else* of *-EndIf* statement. |
| -IfAsset | Tests whether the asset represented by a parameter is currently loaded into AssetX. If the asset is loaded its path is remembered for use by subsequent *–CopyFromAsset* calls. |
| -IfMatches | Tests a tag or string value for a wildcard match. |
| -IfQuery | A conditional test with additional Yes/No query; |

| | |
|---|---|
| | if the condition is true, then display a message requiring a Yes or No answer. If Yes, execute the code up to the next *-Else* or *-EndIf* statement. |
| -IfThumbnail | Tests whether the named file is a 240x180 pixel JPG thumbnail. Returns true if file is a thumbnail and executes any code up to the next *-Else* or *-EndIf* statement. |
| -IgnoreCase | Sets a flag instructing a script that the next function should ignore the case of string parameters. |
| -Images2TGA | Executes PEV's Images2TGA utility to convert any Auran binary textures to uncompressed Targa images. |
| -IncrementKUID | Increases the build version of the selected asset. |
| -Info | Displays a message in the Alert area of the configuration pane. |
| -INIRead | Reads a value from AssetX.ini. |
| -InstallTool | Loads a new tool definition into the Asset·X tool menu. |
| -MakeBogeyTable | A special instruction for updating early content bogey definitions to those suitable for TRS2006 and later. |
| -MakeMeshTable | A special instruction for building a mesh table dependent on the meshes present in the asset. |
| -MakeNormalMap | Creates a normal map from the specified file. |
| -MakeTGA | Creates commonly used single colour image files. |
| -MakeThumbnailTable | Creates or updates the thumbnail container for the current asset. |
| -MenuExecute | Launches any external tool connected to an AssetX menu. |
| -MessageBox | Shows a message in a window with an OK button to dismiss the dialogue. |
| -PM2IM | Runs PEV's PM2IM utility and alters mesh names in config.txt to point to IM files. |
| -PointRead | Reads the position and rotation of an attachment point in the current asset and saves the recovered values to memory. |

| | |
|---|---|
| -PointPosition | Takes the X, Y and Z co-ordinates of a saved point and writes them back to config, with the option of simple arithmetical processing. |
| -PointRotation | Takes the X, Y and Z rotation components of a saved point and writes them back to config, with the option of simple arithmetical processing. |
| -Promote | Moves tags or sub-containers out of their current container and places them one level higher in the Outline tree. |
| -Rename | Renames a tag without changing its value, can also effectively move a tag into or out of any specified container. |
| -RepairTextures | Carries out a variety of repairs to image and texture files. |
| -RepairWaves | Re-encodes or converts audio files for use in TRS and repairs existing files which the game is unable to load. |
| -Replace | Searches for an existing tag containing a specified substring value.  If the tag and substring are found a replacement string will be inserted. |
| -ReplaceAll | Performs a global search and replace on strings or partial strings within a config. |
| -Restore | Places the value stored in variable number 'n' (0 to 9) into tag address specified. Saved variables can be combined using the -Compute command. |
| -Save | Saves the data from the specified tag to the variable number 'n' (0 to 9) for future use. Data is saved as text with any quote marks removed. Saved variables can be combined using the -Compute command. |
| -SaveConfig | Performs an unconditional save of the current config. |
| -Select | Executes script based on selections made from a list of options. |
| -SetEnv | Sets and clears Windows environment variables. |
| -Silent | Enables or suppresses script error messages. |
| -Store | Saves an arbitrary value to a script memory location. |
| -Sort | Sorts the tags and containers in *config.txt* into |

| | |
|---|---|
| | the order nominated in a sample file. |
| -TagForDeletion | Marks a tag or container for deletion. |
| -Tokens | Splits a string into separate tokens using the characters from a second string as delimiters. |
| -Update | Places the data specified in the instruction into the specified tag address. Creates the tag if it does not exist. |
| -UpdateAssetFilename | Removes the asset-filename tag from *config.txt* in later versions where the tag is obsolete. Updates or replaces associated entries. |
| -UpdateCategoryEra | Converts old *category-era-0* tags with one era per tag to the TRS2006 onward *category-era* tags with all eras in a single list. |
| -UpdateCategoryRegion | Converts old *category-region-0* tags with one region per tag to the TRS2006 onward *category-region* tags with all regions in the one list. |
| -UpdateKUIDTable | A special instruction to scan *config.txt* for KUIDs used and create or re-populate the *kuid-table*. Also converts Trainz version 1 single number kuids to the later form. |
| -UpdateUserData | If the current asset is owned by the user, then his/her stored user data is added or updated. |
| -UpdateUsername | Deletes obsolete *name* tags (including localised versions such as *name-fr*) and replaces them with *username.* |
| -Write | Procedures for creating, opening, and writing to simple text files. |

## -BackupAsset

Previous Top Next

**Format:**

*-BackUpAsset|Asset*

Backup the specified asset folder, or backup the current asset if parameter is empty.

**Example**

```
-IfQuery||Backup this asset?
  -BackUpAsset
-EndIf
```

Asks if you want to backup, and if the response is "Yes" then performs a backup of the current asset. Refer to Backups for details of how this is done and how the asset can be restored.

**See Also**

-BackupFile

## -BackupFile

Previous Top Next

**Format:**

*-BackUpFile|Filename*

Backup the specified file, or backup the current config.txt if the parameter is empty.

**Examples**

```
-IfQuery||Backup this Image?
   -BackUpFile|env_metal.tga
-EndIf
```

Asks if you want to backup the image file, and if the response is "Yes" then creates a backup. Refer to Backups for details of how this is done and how the asset can be restored.

```
-IfQuery||Backup config.txt?
   -BackUpFile
-EndIf
```

Creates a backup of the current config.txt for the current asset.

**See Also**
-BackupAsset

**Notes**
The name part of the file specification can include wildcards but the path cannot, an error will be generated if this is attempted. However the contents of asset sub-folders are included by default:

| | |
|---|---|
| -Backup\|%p%\default.im | returns both \default.im and subfolder \default.im |
| -Backup\|%p%\body\default.im | returns files in the \body subfolder only |
| -Backup\|%p%*\default.im | returns an error message |

## -Break

Previous Top Next

**Format:**

*-Break*

This instruction exits unconditionally from a -Foreach loop and moves the execution point to the line following -EndFor.

**Example**

```
-Foreach|Reverse|%tetdata%\category-era.txt
  -If|category-era=^V
    -Break
  -Else
    -Category|category-era||^V|
  -EndIf
-EndFor
```

The code above strips the latest entries from a category tag and breaks from the loop as soon as the tag is reduced to a single entry.

The tag *category-era "1910s;1920s;1930s;1940s;1950s;1960s"* will be changed to *category-era "1910s"*

## -Category

Previous Top Next

**Format:**

*-Category|Tag|ItemsToAdd|ItemsToRemove*

This instruction allows script manipulation of tags which consist of a semi-colon separated list of items.  The tags which can be manipulated with this command are:

*category-keyword*
*category-era*
*category-region*
*custom-category-list*

**Tag**
defines the tag to be processed and is mandatory, both of the other parameters are optional and can be omitted if required.  For *category-keyword* and *category-region* any tags which do not match the predefined standard codes will be ignored.

**ItemsToAdd & ItemsToRemove**
Each of these parameters should comprise a semi-colon separated list of items.  Each of the included items will be added to or removed from the existing tag data in the target asset.

**Example**

*-Category|category-keyword|old;new|ancient;modern*

Adds the items old and new to the target asset (if they are not already present) and removes the items ancient and modern.

## -CallScript

Previous Top Next

**Format:**

*-CallScript|ScriptFile*

Calls the script specified by *ScriptFile* as a sub-routine of the calling script.

On completion of the sub-routine, the calling script will continue execution at the instruction following the *-CallScript* instruction.

With the exception of environment variables (see -SetEnv) no data can be sent to or returned from a called script.

You should be aware that there is nothing to prevent *-CallScript* commands from being nested. However if you write a script which calls itself, or which calls other scripts recursively you can set up a continuous endless loop. To help prevent this Asset·X tracks the name of each script as it is called and will not allow a script to run if it has the same name as one of its ancestors, even if the scripts are in different folders.

**Example**

*-Info|Update Asset to TRS2006 Standard*
*-Condition|trainz-build<2.5*
*-CallScript|sub\username.axm*
*-CallScript|sub\fontspath.axm*
*-UpdateCategoryEra*

**Note:**
The *ScriptFile* parameter can also contain a sub-folder name as shown. For this version of the program the scripts are stored in *...\AssetX\scripts*, so the scripts above are in the *...\AssetX\scripts\sub\* folder.

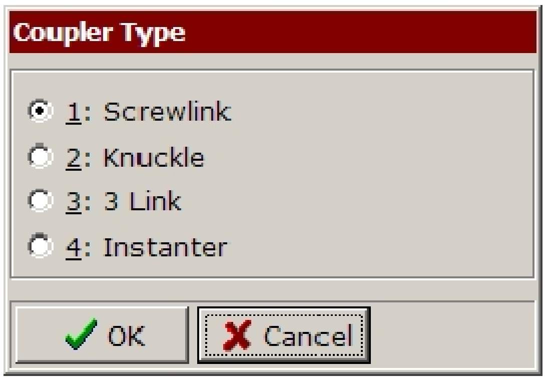
## -Case

Previous Top Next

**Format:**

*-Case|Option 1,Option 2,Option 3,...,Option n|Caption*

An interactive function which provides a dialogue box and requests a selection from the user.


Coupler Type

(●) 1: Screwlink
( ) 2: Knuckle
( ) 3: 3 Link
( ) 4: Instanter

**Example**

*-Case|Screwlink,Knuckle,3 Link,Instanter|Coupler Type*

The first parameter is a comma separated list of alternative options.  The second (which is optional) provides a title for the dialogue. The script code below gives an example of how this function can be used.

```
-Case|Screwlink,Knuckle,3 Link,Instanter|Coupler Type
; code here will be executed if the dialogue is cancelled, if there is no code
; in this section execution will continue after the -EndCase statement.
 -IfQuery||Are you sure you want to cancel
   -Info|Script cancelled
   -Exit
 -EndIf

-Case|1
; selecting 1: Screwlink will transfer execution to this point
; when the code has run execution will transfer to the line after -EndCase
 -CallScript|screwlink.axm

-Case|2
 -MessageBox|Doing a Knuckle

-Case|3
 -If|kind=track
   -MessageBox|Looking at a track
 -Else
   -MessageBox|Looking at something else
 -EndIf

-Case|4
 -MessageBox|Found Case 4

-EndCase
-Info|Script completed
```

**Notes:**

- A maximum of nine choices are allowed.
- Each option string must be at least two characters in length.
- You can use *-If -Else and -EndIf* in each section but the entire construct must be balanced and closed within the same section.
- You can also use *-CallScript*, in which case the child script will be run and execution will continue after completion. The nested script will not be able to access the case statement parameters.
- You cannot use nested *-Case* statements.
- There must be a matching *-EndCase* for every instance of a *-Case* block.

**See Also**

-EndCase

## -Clear

Previous Top Next

**Format:**

*-Clear|CommandString*

Removes the tags described in *CommandString* from the config.

*CommandString* is a comma separated list containing any of the following elements:

| Element | Description |
|---|---|
| *PmMeshes* | Scans for and removes any *.pm meshes where corresponding *.im meshes exist. These may be left behind by the use of PM2IM.<br><br>In single script mode you will be asked for confirmation unless you have selected *Auto Delete Duplicate *.PMs* in the *Options/Preferences* dialogue. In batch mode the meshes will only be deleted if this option is checked.<br><br>It is strongly recommended that the asset is backed up and checked for correct operation in game before this command is used. |
| *NullContainer* | Removes empty containers even if they are valid in the current context. |
| *NullString* | Removes empty string tags even if they are valid in the current context.<br><br>Take care with this option since it may cause problems with some scripts and with some mesh effects. |
| *NullData* | Removes empty non string tags even if they are valid in the current context. |
| *Invalid* | Removes tags and containers which are invalid in the current context. This will include both obsolete tags and tags which are invalid for other reasons, including incorrect spelling.<br><br>Some known items, such as *bluestar* tags, are known to cause problems if they are deleted. You will be asked for confirmation in single script mode. In batch mode these tags will not be removed but their existence will noted in the batch report. |

*-Clear* does not check whether individual tags can be updated or amended so it should only be called after individual upgrades have been performed by other script methods.

**Examples**

*-Clear|PmMeshes,NullContainer,NullString,NullData,Invalid*

Removes tags from all of the possible categories.

*-Clear|NullContainer*

Removes empty containers only.

**Notes**

Except for the removal of *.pm meshes this command can be undone.

## -Comment

Previous Top Next

**Format:**

*-Comment|Tag*

Marks the specified tag as a comment.  If the tag is a container all of its children will also be marked, continues if the tag is not found.

**Example**

*-Comment|config.txt\mesh-table*

## -Combine

Previous Top Next

**Format:**

*-Combine|n|Parameter String*

Combines a string value from saved variables and stores it into script variable number *n*.

**Example**

```
-Save|0|category-era-0
-Save|1|category-era-1
-Save|2|category-era-2
-Combine|3|^0;^1;^2
-Restore|3|category-era
```

Substitutes the current value of a saved variable for any number found in the parameter string.  Other characters in the parameter are not affected.

If the config contains these values:

```
category-era-0 1910s
category-era-1 1920s
category-era-2 1930s
```

Then the example above will produce a new tag:

*category-era 1910s;1920s;1930s*

Note that any non numeric value within the parameter string is retained and that only string substitutions are provided for.

The stored values are not cleared by the script engine. You should never assume that a variable will contain any particular value unless your script has set it to that value.

Note
The syntax of this command has changed to allow better support for Script Variables.

**See Also**

-Compute
-Restore
-Save

## -Compute

**Format:**

*-Compute|n|value1 [+,-,/,*] value2 .... [+,-,/,*] value#*

Allows simple arithmetical calculations based on the contents of Script Variables.

The first parameter specifies the memory location in which the calculated result will be stored. It should be in the range ^0 .. ^9.

The second parameter contains at least one item which evaluates to an integer or floating point value. This can either be a number or a variable which evaluates to a number. Optionally this can be followed by any number of additional data pairs, each of which must be a numeric operator (+, -, / or *) followed by a number. The parameter list can include spaces to make it more easily readable.

The function calculates a total from the data provided and stores the result in the memory location specified by the first parameter. This can be used as a script variable in further calculations or written to config.txt using the -Restore function

The calculation is performed left to right and there is no facility for parentheses or intermediate sub-totals. In some cases therefore it will be necessary to use multiple calls.

**Examples**

*-Compute|0|2 + 2 + 2 / 3*

Equivalent to (((2 + 2) + 2) / 3)
Stores the value 2 in memory location 0

*-Compute|0|2 + 2*
*-Compute|1|2 / 3*
*-Compute|3|^0 + ^1*

Equivalent to (2 + 2) + (2 / 3)
Stores the value 4.667 in memory location 3

**Notes**
The result is saved as an integer value if possible and, if not, as floating point rounded to 3 decimal places.

**See Also**

-Combine
-Restore
-Save

## -Condition

Previous Top Next

**Format:**

*-Condition|test*

Decides from the specified test whether to continue executing the script or to terminate.

The test must be ONE of the following:

| Script Test & Usage | Description |
|---|---|
| *tag=value* | The script will continue if the specified tag exists and has the specified value. |
| *tag!value* | The script will continue if the specified tag exists but does not have the specified value. |
| *tag<value* | The script will continue if the tag's contents are less than the specified value. |
| *tag>value* | The script will continue if the tag's contents are greater than the specified value. |
| *(exists)filespec* | The script will continue if any file is found which matches *filespec.* |
| *(inherits)tagset* | The script will continue if the current asset inherits its capabilities from the specified tag set. |

**Examples**

*-Info|Update Asset to TRS2006 Standard*
*-Condition|trainz-build<2.5*
*-CallScript|sub\username.axm*
*-CallScript|sub\fontspath.axm*
*-UpdateCategoryEra*

The *-Condition* statement above checks the value of the tag *trainz-build* and continues to execute the script if the build number is less than 2.5; otherwise it terminates script execution.

*-Info|Add Mesh Table*
*-Condition|(inherits)mesh-asset*
*-MakeMeshTable*

In this example *-Condition* checks whether the current asset inherits its tag data from *mesh-asset* (if it does not it cannot support a mesh table)

## -CopyFile

**Format:**

*-CopyFile|Source|Destination*

Copy the specified source file to the specified destination, renaming the file if necessary. If *destination* is empty the file is copied under the same name to the root folder of the current asset.

**Examples**

```
-IfQuery||Copy Image to this asset?
  -CopyFile|c:\pictures\env_metal.tga
-EndIf
```

Asks if you want to copy the image file to this asset, and if the response is "Yes" then performs a file copy of the env_metal.tga image into this asset.

```
-IfQuery||Copy to NewName.bmp?
  -CopyFile|c:\pictures\oldname.bmp|newname.bmp
-EndIf
```

Copies and renames the file.

**See Also**
-CopyFromAsset

**Notes**
The *Source* file must exist. If no folder is specified the current asset root folder is assumed.

The *Destination* folder must exist. If it is not actually specified the current asset root folder is assumed.

If you are specifying a folder but no filename you must ensure that your folder specification ends with a backslash. %bin% for example expands to *C:\Program Files\PevSoft\AssetX\bin*. If you don't add a backslash to this the folder will be expanded to *C:\Program Files\PevSoft \AssetX* and the filename will be expanded to *bin*. This particular case is tested for and will generate an error but there may be similar examples.

## -CopyFromAsset

**Format:**

*-CopyFromAsset|Filename*

Used in an *–IfAsset..-EndIf* block, this command copies *Filename* from the asset referenced in a previous call to *-IfAsset* to the current asset.

**Example**

*-IfAsset|<kuid:329364:1017>*
  *-CopyFromAsset|env_metal.tga*
  *-CopyFromAsset|env_metal.texture.txt*
*-EndIf*

The example copies env_metal.tga and env_metal.texture.txt to the present asset, from asset <kuid:329362:1017> if it is open in AssetX.

If the referenced kuid is not loaded then nothing is copied.

**See Also**
-CopyFile

## -Delete

Previous Top Next

**Format:**

*-Delete|Tag*

Deletes the specified tag, continues if the tag is not found.

**Example**

*-Delete|config.txt\asset-filename*
*-Delete|config.txt\.cache*
*-Delete|config.txt\frontlength*

The *-Delete* instructions above can be part of an upgrade script, deleting the tags specified if they are present, and continuing on if any of the tags are not found.

*-Delete|config.txt\mesh-table*

In this case the specified tag is a container and deleting it will also delete all of its contents.

**See Also**
-TagForDeletion

## -DeleteFiles

Previous Top Next

**Format:**

*-DeleteFiles|filespec*

Deletes file(s) defined by *filespec*, see (exists) for syntax options.

**Example**

*-DeleteFiles|env_metal.tga*
*-DeleteFiles|*.texture*

Deletes the env_metal.tga file and any binary texture files in the asset.

*-DeleteFiles|%log%*.**

Deletes the entire contents of the *...\AssetX\log* folder.

## -EndCase

Previous Top Next

**Format:**

*-EndCase*

Marks the end of a *-Case* statement. Refer to -Case for details.

**See Also**

-Case

## -EndFor

Previous Top Next

**Format:**

*-EndFor*

Terminates a *-Foreach* statement.

**See Also**

-Break
-Foreach

## -EndSelect

Previous Top Next

**Format:**

*-EndSelect*

Marks the end of a *-Select* statement. Refer to -Select for details.

**See Also**

-Select

## -Exchange

Previous Top Next

**Format:**

*-Exchange|OldTag OldValue|NewTag NewValue*

Exchanges one tag with a specified value for another, also with a specified value. The old tag is deleted if the exchange is successful, otherwise nothing is done.  Each tag and its respective value must be separated by a single space character.

**Examples**

*-Exchange|secret 1|\privileges\permit-listing 0*

Looks for the existing tag *secret* and, if its value is 1, replaces it with the new tag privileges \permit-listing.  In this case creating the new tag also creates the parent container if it doesn't already exist.

If *secret* is not found or if it does not have the value of 1, nothing will be done.

Note that both *OldTag* and *NewTag* must be tags and not containers.

## -Exit

**Format:**

*-Exit|condition|message*

Unconditionally exit from the script (with an optional error message). This would normally be used inside an *–If/-Endif* block or during script de-bugging.

**Example**

*-If|kind!traincar*
*-Exit*
*-Endif*

Ceases execution of the script if the asset kind is not traincar.

*-If|kind!track|Script will only operate on track assets:*
*-Exit*
*-Endif*

Ceases execution of the script if the asset kind is not track and advises the user accordingly.

**Note**
If the message is a comment string preceded by a semi colon, it will be ignored.

## -Expand

**Format:**

*-Expand|Container*

Expands a container to reveal its contents. The result is the same as clicking on the [+] icon in the tree view of the configuration pane.

**Example**

*-Expand|config.txt\mesh-table*

Expands the view of the mesh table so that you can see all of its contents. Does nothing and continues if the container does not exist.

## -FixKUIDs

Previous Top Next

**Format:**

*-FixKUIDs*

Updates certain legacy Auran kuid tags to the current standard format.

**Example**

*-FixKUIDs*

A config.txt entry of

*bogey 12345*

will be updated to

*bogey <kuid:-1:12345>*

## -ForEach

Previous Top Next

**Format:**

*-ForEach|List|Specifier*
*...*
*...*
*-EndFor*

Carries out the instructions between the *-Foreach* and *-EndFor* tags for every member of a list. When the operation is complete the script continues at the next line after the *-EndFor* statement.

The first parameter (which must match one of the options below) defines the type of list to be used and the second parameter defines the list source.

| List Format | Description |
|---|---|
| *-Foreach\|File\|Filespec* | *Filespec* is a wildcard filemask (see File Searches for details of how to form the mask). The script will search for files matching the specification and will carry out the instructions within the statement block for every file. |

| | |
|---|---|
| *-Foreach\|Line\|File* | *File* is an existing text file on disc, wildcards are not permitted. The script will carry out the statements in the block for each line in the file. It is the author's responsibility to ensure that the file is a valid text file. |
| *-Foreach\|Reverse\|File* | The same as *-Foreach\|Line\|File* except that the file is loaded in reverse order, last line first. |
| *-Foreach\|Tag\|Tagspec* | *Tagspec* is a wildcard tag specification. The statements will be executed for each matching tag. |
| *-Foreach\|Option\|List* | *List* is a user defined, comma separated list of string options. |
| *-Foreach\|Number\|IntList* | *IntList* is a user defined, comma separated list of numbers in the form 1,2,4-7,10. All numbers must be positive integers and there should be no spaces in the parameter. The list is sorted and duplicate items are removed before processing. |

**Examples**

*-ForEach|File|*.im*

*-ForEach|Line|%tetdata%\category-region.txt*

*-ForEach|Tag|\mesh-table*\auto-create*

*-ForEach|Option|Front,Back,Left,Right*

*-ForEach|Number|1,3-7,9*

**Notes**

-Foreach statements cannot be nested.

If and Case statements may be included but each block must be fully defined and closed within the -Foreach / -EndFor block.

-CallScript may be used within the statement and the nested script can include its own -Foreach statements but the list parameters cannot be passed between scripts.

See script variables for notes on how to access the pointer index and the value of the current list item.

Every -Foreach statement must have a matching -EndFor.

**See Also**

-Break

-EndFor

## -If -IfNot -Else -EndIf

Previous Top Next

**Format:**

*-If|condition*
  *...do stuff*
*-Else (optional)*
  *...do other stuff*
*-EndIf*

*-IfNot|condition*
 *...*
*-Else*
*...*
*-EndIf*

Allows branches of execution depending on the result of the the condition. If the condition is met then the following code is executed. Otherwise the following code is skipped until the next *-Else* or *-Endif* is encountered.

The condition must be ONE of the following:

| Test & Usage | Description |
|---|---|
| *tag=value* | The script will continue if the specified tag exists and has the specified value. |
| *tag!value* | The script will continue if the specified tag exists but does not have the specified value. |
| *tag<value* | The script will continue if the tag's contents are less than the specified value. |
| *tag>value* | The script will continue if the tag's contents are greater than the specified value. |
| *(exists)filespec* | The script will continue if file are found which match *filespec.* |
| *(inherits)tagset* | The script will continue if the current asset inherits its capabilities from the specified tag set. |

As can be seen from this example, If blocks can be nested. Ensure that each *-If* has a closing *-EndIf*.

**Example**

```
-If|kind=traincar
  -If|config.txt\engine=1
    -If|config.txt\trainz-build>1
      -Update|config.txt\script|"loco"
      -Update|config.txt\class|"loco"
    -EndIf
    -If|config.txt\origin="GWR"
      -Update|config.txt\origin|"BR Western"
    -EndIf
  -Else
    -Update|config.txt\script|"carriage"
    -Update|config.txt\class|"carriage"
  -EndIf
-EndIf
```

The example loads the correct script and class reference for a traincar as an engine (loco) or a wagon (carriage). The nested portions also are performed when the traincar is an engine (engine=1).

*-IfAsset* and *-IfQuery* are alternative forms of *-If* statement used for special purposes.  Both are interchangeable with *-If* and can be nested.

*-IfNot* is a variant which reverses the value of the test condition.

*-IfMatches* allows wildcard comparisons for string or numeric values.

**See also:**

-IfAsset
-IfMatches
-IfQuery

## -IfAsset

Previous Top Next

**Format:**

*-IfAsset|<kuid>*

Tests whether the asset represented by *<kuid>* is currently loaded into AssetX. If the asset is loaded, its path is remembered for use by subsequent *–CopyFromAsset* calls. This command must be used in conjunction with *-EndIf* and *-CopyFromAsset*.

**Example**

```
-IfAsset|<kuid:329364:1017>
  -CopyFromAsset|env_metal.tga
  -CopyFromAsset|env_metal.texture.txt
-EndIf
```

The example copies env_metal.tga and env_metal.texture.txt to the present asset, from asset <kuid 329362:1017> if it is open in AssetX. If the specified asset is not found the *-CopyFromAsset* calls will be skipped.

Note that once the -EndIf instruction has executed calls to -CopyFromAsset will no longer work. All such calls must be executed within an -IfAsset/-CopyFromAsset block.

**See also:**

-If -IfNot -Else -Endif

## -IfMatches

Previous Top Next

**Format:**

*-IfMatches|Pattern|String*

Tests whether the wildcard string *Pattern* matches the string represented by *String*

The function first attempts to interpret *String* as a tag and retrieve its value. If there is no matching tag then string is interpreted literally. Any Script Variables or AssetX macros within either parameter are evaluated before the test is performed.

**Examples**

```
-IfMatches|*18?0s*|category-era
  -MessageBox|Asset is appropriate for Victorian layouts
-Else
  -MessageBox|Unsuitable for Victorian layouts
-EndIf
```

This example looks for nineteenth century references anywhere within the category-era tag.

```
-IfMatches|<kuid:*|kuid
  -IncrementKuid
-EndIf
```

Updates the kuid tag but only if it is not already a kuid2 value.

**Notes**

-IfMatches clauses must be terminated with -EndIf statements and can also contain an Else qualifier.

**See also:**

-If -IfNot -Else -Endif

## -IfQuery

Previous Top Next

**Format:**

```
-IfQuery||Message
  ...do stuff
-Else
  ...do other stuff
```

```
-EndIf
```

OR

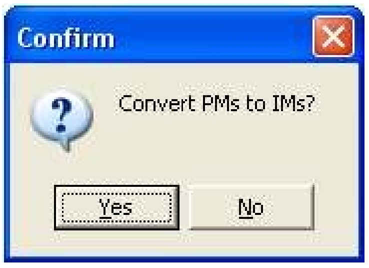
```
-IfQuery|condition|Message
  ...do stuff
-Else
  ...do other stuff
-EndIf
```

Displays a message in a box that requires a "Yes or No" response. If the response is "Yes" the script will execute the code that follows up to the *-EndIf* or *-Else* statement. If a condition is included the condition must be also met to open the message box.

**Example**

```
-IfQuery||Convert PMs to IMs?
  -PM2IM
  -Expand|config.txt\mesh-table
-EndIf
```

The example above causes the following message box to be displayed:



Convert PMs to IMs?

If the user responds with "Yes" the macro will call PEV's PM2IM utility to convert all progressive meshes in the present asset into indexed meshes. Additionally the references to the PMs in the config.txt will be changed to IMs. The example above also calls for the mesh-table to be expanded so the user can see the changes made by the PM2IM instruction.

If the user presses the "No" button, the macro will not execute the two instructions before the *-EndIf*, and will continue with any instructions that follow.

If there is a condition in the instruction as well we get:

```
-IfQuery|trainz-build>2.4|Convert PMs to IMs?
  -PM2IM
  -Expand|config.txt\mesh-table
-EndIf
```

The example above causes the message box to be displayed only if trainz build is greater than 2.4. In other cases there will be no message and the subsequent instructions will be ignored.

**Note:**

In batch mode user interaction is disabled and the response to any query is assumed to be *Yes*. If a condition is included it will be respected however.

**See also:**

-If -IfNot -Else -Endif

## -IfThumbnail

**Format:**

```
-IfThumbnail|Filename
  ...do stuff
-Else
  ...do other stuff
-EndIf
```

Checks if the named file is a standard Trainz thumbnail image i.e. a JPG 240 wide x 180 high.

If the file is a thumbnail, the script will execute the code that follows up to the *-EndIf* or *-Else* statement.

**Example**

The code below will search the opened asset for JPG files and checks that any found are thumbnails.
For the thumbnails that are found the macro will then execute an external tool using the -MenuExecute instruction.
In this case the tool is Images2TGA with parameters to add the "watermark_CRG.tga" to the thumbnail.

```
-ForEach|File|*.jpg
  -IfThumbnail|^F
    -Info|found thumbnail
    -MenuExecute|*.jpg|CRG Watermark|^F
  -EndIf
-EndFor
```

This example uses a tool (Images2TGA) that is associated in AssetX with *.JPG ( the "CRG Watermark" in the right click popup menu).

In the configure tools feature, these are the settings for Images2TGA to add a watermark to a thumbnail.

| Switches | File Parameters (Quoted) |
|---|---|
| | %pev%\Images2TGA\Images2TGA.exe |
| -watermark·-close· | %pfe% |
| watermark_CRG.tga | |
| | |
| Show Window | Minimised |
| Capture Output | Show in Message Box |
| Before Action | Save Selected File |
| After Action | None |

The -MenuExecute instruction replaces the file path and name macro (%pfe%)  with the path and file name found as ^F in the -ForEach routine.

**See also:**

-MenuExecute
-ForEach
-If -IfNot -Else -Endif

## -IgnoreCase

Previous Top Next

**Format:**

*-IgnoreCase*

Instructs the script processor to ignore the case of string parameters. The IgnoreCase flag is reset after every call so it must be issued immediately before the call to which it applies. Currently this method affects calls to -Replace and -ReplaceAll.

**Example**

*-IgnoreCase*

**See Also**

-Replace
-ReplaceAll

## -Images2TGA

Previous Top Next

**Format:**

*-Images2TGA*

A special instruction to execute PEV's Images2TGA utility.

Because assets with Auran binary textures cannot be committed into TS2009 or TS2010, this instruction is included to allow early assets to be transferred into these versions.

The instruction executes Images2TGA to convert any Auran textures in an asset to uncompressed Targa images. Images2TGA also creates the necessary texture.txt files to connect the images to the game.

Please note that you require version 1.08 or later of Images2TGA to work with Asset·X.

**Example**

*-Images2TGA*

## -IncrementKUID

Previous Top Next

**Format:**

*-IncrementKUID*

Increases the build version of the selected asset.

**Example**

*-IncrementKUID*

If the existing value is <kuid:1234:5678> it will be updated to <kuid2:1234:5678:1>

If the existing value is <kuid2:1234:5678:9> it will be updated to <kuid2:1234:5678:10>

**Notes**
Like the corresponding menu command this is only functional for your own assets.

## -Info

Previous Top Next

**Format:**

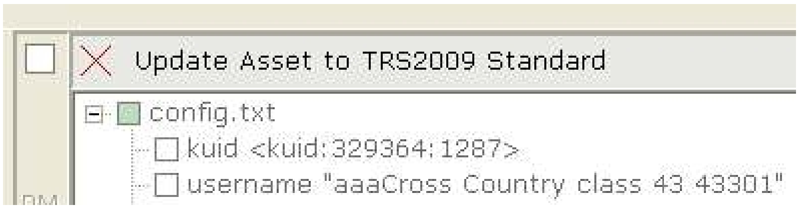
*-Info|Message*

Displays the Message in the "alert" bar at the top of the Configuration Pane of Asset·X.

**Example**

*-Info|Update Asset to TRS2009 Standard*

The example above displays the message in the Configuration Pane as follows:



Update Asset to TRS2009 Standard
config.txt
kuid <kuid:329364:1287>
username "aaaCross Country class 43 43301"

**Note:**

In batch mode the messages will be displayed in the final report screen rather than the Alert bar.

**See Also**
-MessageBox

## -INIRead

Previous Top Next

**Format:**

*-INIRead|n|Section|Key*

Reads the value of *Key* from the specified *Section* and stores the result in script variable *n*.

**Example**

*-INIRead|0|Paths|PevSoft*

*-MessageBox|^0*

The example above reads the stored path to the PevSoft folder from AssetX.ini and displays it in a message window.

## -InstallTool

Previous Top Next

**Format:**

*-InstallTool|Menu|ToolData*

A special instruction to create new tool options in AssetX.

Suppliers of Trainz tools can use this instruction to create a macro that will automatically set up a new tool. To create the data, firstly manually set up a new tool using the AssetX *Options/Configure Tool* menu selection. Once the new tool is working correctly open the *AssetX.ini* file and copy the entry for the new tool into the data field of the *-InstallTool* instruction.

The *Menu* parameter specifies which menus the tool will appear in.  Specifying *Tools-Menu* your new tool will appear in the main application menu under *Tools*.  Any other value causes the tool to appear in the right click context menu of the Filelist pane.

'Tool' can be a document, helpfile, URL or even a mailto: string as well as a command line.

**Example**

As a tool supplier you may wish to connect your new tool to open PM files from the popup menu in the Files View pane of AssetX. Use the Options/Configure Tools menu to connect your tool to the *.pm files selection. Once you have your new tool working correctly as a call from AssetX, you can gather the data to make your macro.

Open the AssetX.ini file in notepad and go to the *[*.pm]* entry. The new tool will be in the numbered list. It's entry will look something like:

*[*.pm]*
*1=My Tool¦¦"Program Files\MyTools\MyTool.exe"¶"%pfe%"¦1¦0¦0¦0¦*

To make your script command, put **.pm* into the *Menu* field and then strip off the entry number and equal sign from the second line and add it as the *ToolData* parameter.

Remember that the pipe symbol ( | shift-backslash) is the field separator for the script system but the tool definition uses a different character (on most keyboards it can be entered by holding down the alt key and entering 2 2 1 on the numeric keypad)  To be certain of adding the correct delimiters it is best to use copy and paste.

So your script line will be:

*-InstallTool|*.pm|My Tool¦¦"Program Files\MyTools\MyTool.exe"¶"%pfe%"¦1¦0¦0¦0¦*

Alternatively you can *Copy* a command from the menu editor grid in Options/Configure Tools and *Paste* it into the second command parameter.

## -MakeBogeyTable

Previous Top Next

**Format:**

*-MakeBogeyTable*

A special instruction for updating early version assets to TRS2006 and later. The instruction calls routines in Asset·X to extract bogey data and present it in later format, automatically creating the bogey table and making the appropriate entries for the reversed tag.

**Example**

*-MakeBogeyTable*

Creates the bogey table for traincars if it does not already exist. Does nothing if there are no bogey entries.

If only one bogey tag is found in the asset this routine assumes that the same bogie will be used for front and rear and creates two entries to reflect this.

## -MakeMeshTable

Previous Top Next

**Formats:**

*-MakeMeshTable*

A special instruction for updating early version assets to TRS2006 and later. The instruction calls routines in Asset·X to scan the asset for the presence of a main mesh and a shadow mesh. If there is no mesh-table in *config.txt* a new one will be made and populated with the mesh data.  The tag *auto-create 1* will be added to the default mesh container and animations will be added if found.

If both *.im and *.pm files with the same path and filename are found the user will be prompted to delete the duplicate *.pm meshes.

This command can also create a mesh-table referencing all meshes in the asset using numerically serialised labels, a form suitable for a mesh library asset.

**Examples:**

*-MakeMeshTable*

Creates and populates the mesh-table for any asset if it does not already exist.

Does nothing if there is already a mesh-table, if there are more than 2 meshes in the asset or in other circumstances where the intended use of the meshes cannot be fully resolved.

For a an asset which is of kind 'mesh', the script creates a list of all *.im, *.pm and *.lm.txt meshes found in the asset (including sub-folders) and provides a mesh-table with numerically serialised submesh names.  The tag auto-create 1 is also added to each container and, if an animation file with the same name as the mesh file is found, the tag *anim meshname.kin* is also added.

**Notes:**

*-MakeMeshTable* will not alter any existing mesh-table.

To change this behaviour you can delete the existing mesh-table before issuing this command:

*-Delete|mesh-table*
*-MakeMeshTable*

## -MakeNormalMap

Previous Top Next

**Format:**

*-MakeNormalMap|SourceFile|TargetFile*

Creates a normal map from SourceFile and saves it to the TGA file specified by TargetFile.

*-MakeNormalMap|SourceFile*

Creates a normal map from SourceFile and saves it to a TGA file name generated by appending the text *-normal.tga* to the end of the source filename.

*-MakeNormalMap*

Creates a normal map from the currently selected file and saves it to a TGA file name generated by appending the text -normal to the end of the filename.

**Notes**

If the source filename includes a disc drive specifier the file will be opened, and the target file saved to the specified locations.  If there is no path the file will be opened from, and saved to, the current asset.

Sourcefile can be any BMP, JPG or TGA file, however the output file will always be in TGA format.

## -MakeTGA

Previous Top Next

**Format:**

*-MakeTGA|Filename*

Creates 8 x 8 pixel single colour TGA image files to replace textures which are commonly missing from assets.  The blue value of one pixel is adjusted to avoid falling foul of CMP uniform colour errors and warnings.

The files available are:

- blank.tga, this is an alpha blank texture used by reskinners to hide submeshes
- black.tga
- blue,tga
- green.tga
- red.tga
- white.tga
- yellow.tga

**Example**

*-MakeTGA|yellow.tga*

## -MakeThumbnailTable

Previous Top Next

**Format:**

*-MakeThumbnailTable*

Creates or updates the asset's thumbnails container.  This routine searches for art icon files as well as thumbnails using commonly defined names.  You should check the result carefully since not all naming conventions can be fully determined by examining the asset.

**Example**

*-MakeThumbnailTable*

## -MessageBox

Previous Top Next

**Format:**

*-MessageBox|Message*

Displays the message in a window with an 'OK' button.

**Example**

*-MessageBox|The script has completed.  Please restart AssetX to use the new tool.*

**See Also**
-Info

## -MenuExecute

Previous Top Next

**Format:**

*-MenuExecute|Menu|Menu Item|TargetFile*

Executes the external tool which is defined by the menu name/menu item to process the target file.

**Or**

*-MenuExecute|Menu|Menu Item*

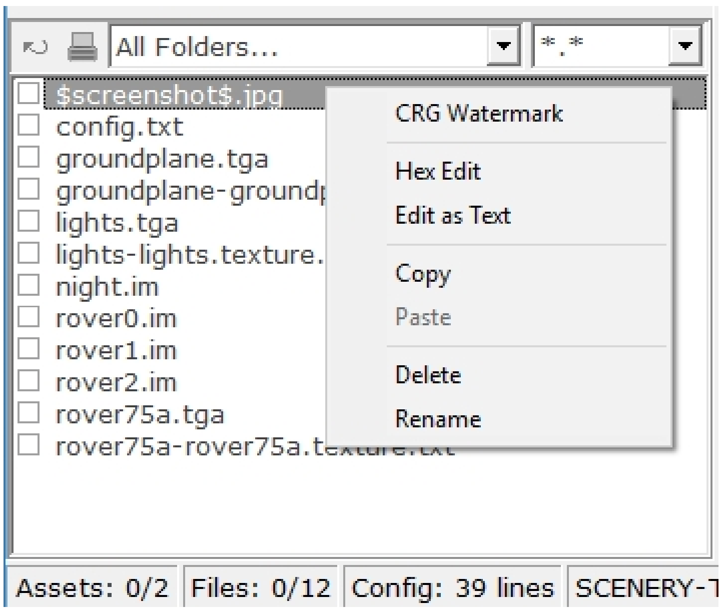
Executes the external tool which is defined by the menu name/menu item.  This option may be used for TrainzUtil commands that process the selected asset, or for other tools that process the asset rather than files within the asset.

MenuExecute only launches tools that are set up in AssetX as menu items. See Tool Definition for instructions for attaching tools to AssetX.

**Example**

The code below is repeated from -IfThumbnail as it shows how the two instructions are used together.
This code will search the opened asset for JPG files and checks that any found are thumbnails.
For the thumbnails that are found the macro will then execute an external tool using the -MenuExecute instruction to launch Images2TGA with appropriate parameters to add the "watermark_CRG.tga" to the thumbnail.

```
-ForEach|File|*.jpg
  -IfThumbnail|^F
    -Info|found thumbnail
    -MenuExecute|*.jpg|CRG Watermark|^F
  -EndIf
-EndFor
```



The image here shows the popup menu for the $screenshot$.jpg file with the user defined menu items above the dividing lines.

In this case the desired tool is called up by CRG Watermark.

The right click popup menu for JPEG images is stored in the INI file under the file extension section named "*.jpg", so our menu name in this case is *.jpg (lower case) and our menu item is CRG Watermark so our -MenuExecute parameters are as shown above. The ^F parameter is the file name resulting from the asset scan done by the -ForEach|File instruction. The path is automatically added by the instruction. So if you add an -Info|^F the file name $screenshot$.jpg will be written to the output (Alert) box.

**Notes**
For items in the CustomA to CustomE menus we use the Menu Name as the first parameter. This is stored in the CustomA (for example) section in the INI file a the first line "Title=Menu Name" in the numbered list for the section.

**See also:**

-IfThumbnail
-ForEach
-If -IfNot -Else -Endif

## -PointRead

Previous Top Next

**Format:**

*-PointRead|pointname*

Reads the position and orientation of the specified attachment point in the current asset and saves the data to memory. The saved data will be retained during the current AssetX session until overwritten by a subsequent call to this procedure.

**Example**

*-PointRead|a.limback*

Searches the current asset for the first mesh found containing an attachment point named a.limback and saves its position and rotation to memory. If no point is found an error message is issued.

**Notes:**
To prevent -PointPosition and -PointRotation from using invalid data *-PointRead* will always reset the point memory location to an empty string following an unsuccessful call.

An error is returned if the attachment point is not present in any of the asset's meshes, or if the asset has no meshes.

**See Also**
-PointPosition
-PointRotation

## -PointPosition

Previous Top Next

**Format:**

*-PointPosition|Tag|X|Y|Z*

Retrieves location data generated by the last call to -PointRead and writes it to the specified tag, this will normally be used with the position tag. Note the use of lower case letters to distinguish the parameters from those used by -PointRotation.

The parameters should be entered as follows:

- ^X, ^Y or ^Z respectively if you want to use the existing locations unchanged.
- ^X, ^Y or ^Z followed by a simple arithmetic operator and a numeric operand.
- A specific numeric value.

| | |
|---|---|
| -PointPosition\|tag\|^X\|^Y\|^Z | writes the location unchanged |
| -PointPosition\|tag\|^Y\|^X\|^Z | swaps the X and Y co-ordinates |

| | |
|---|---|
| -PointPosition\|tag\|^X\|^Y\|0.0 | changes the height of the location to zero without altering X or Y |
| -PointPosition\|tag\|^X\|^Y\|^Z+1.0 | moves the location upwards by one metre |
| -PointPosition\|tag\|^X1.0\|^Y/2.0\|^Z*3.0 | subtracts 1 from X, divides Y by 2 and multiplies Z by 3 |

**Example**

*-PointPosition|\mesh-table\default\position|^X|^Y|0.0*

Retrieves the point location data and writes the X, Y co-ordinates unchanged to the tag position in the default mesh, the Z co-ordinate is changed to zero.

*-PointPosition|\mesh-table\default\position|*^X**2.0|^Y/2.0|^Z-3.0*

Retrieves the data, multiplies X by 2.0, divides Y by 2.0 and subtracts 3.0 from Z. The resulting values are written to the specified tag.

**Note**
Arithmetic errors, such as division by zero, will produce an error.

**See Also**
-PointRead
-PointRotation

## -PointRotation

Previous Top Next

**Format:**

*-PointRotation|Tag|^x|^y|^z*

Retrieves orientation data generated by the last call to -PointRead and writes it to the specified tag, this will normally be used with the position tag.  Note the use of lower case letters to distinguish the parameters from those used by -PointPosition

The parameters should be entered as follows:

- ^x, ^y or ^z respectively if you want to use the existing rotations unchanged.
- ^x, ^y or ^z followed by a simple arithmetic operator and a numeric operand.
- A specific positive value. (in degrees)

| | |
|---|---|
| -PointRotation\|tag\|^x\|^y\|^z | writes the rotations unchanged |
| -PointRotation\|tag\|^y\|^x\|^z | swaps the X and Y rotation values |
| -PointRotation\|tag\|^x\|^y\|0.0 | changes the rotation along the Z axis to zero without altering X or Y |
| -PointRotation\|tag\|^x\|^y\|^z+180 | reverse the direction of the Z axis rotation |

| | |
|---|---|
| -PointRotation\|tag\|^x1.0\|^y/2.0\|^z*3.0 | subtracts 1 from X, divides Y by 2 and multiplies Z by 3. |

**Example**

*-PointRotation|\mesh-table\default\position|^x|^y|^z+180.0*

Retrieves the point rotations and reverses the direction of the Z axis.

**Notes**
Arithmetic errors, such as division by zero, will produce an error.

The point values are retrieved, and the calculations are carried out, in degrees. However the output is converted to radians before being written to config since this is the format expected by trainz.

**See Also**
-PointRead
-PointPosition

## -PM2IM

Previous Top Next

**Format:**

*-PM2IM*

A special instruction to execute PEV's PM2IM utility to search for *.PM meshes and, if they exist, change the references to *.PMs in config.txt to reference *.IM meshes with matching names.

Please note that you require version 1.19 or later of PM2IM to work with AssetX.

**Example**

*-PM2IM*

This instruction can be followed by *-Expand|mesh-table* so the user can see the changes made by the PM2IM instruction.

## -Promote

Previous Top Next

**Format:**

*-Promote|Tag*
*-Promote|Container*

Moves a tag or container out of its current container to the next level up.  Either tags or containers can be moved and if the target is a container all of its child tags and containers will also be moved.

**Example**

```
; promote any subcontainers found inside the obsolete table
-Foreach|Tag|obsolete-table\*\
  -Promote|^V
-EndFor

; promote any tags inside obsolete table which do not have kuid values
-Foreach|Tag|obsolete-table\*
  -IfMatches|<kuid*|^V
  -Else
    -Promote|^V
  -EndIf
-EndFor
```

This example will correct a common error that can arise when a config.txt is loaded into AssetX which does not have a closing brace in the obsolete-table. AssetX will automatically add a closing brace at the end of the file to re-balance the contents but this may sometimes result in tags and containers being incorrectly included inside the obsolete-table.

**Notes**
-Promote cannot process tags or containers at the top level, an error message will be shown if you attempt to do so.

When using this command be sure that you are specifying the tag address correctly.

## -Rename

Previous Top Next

**Format:**

*-Rename|OldTag|NewTag*

Renames a tag without changing its value, can also effectively move a tag into or out of any specified container. If *NewTag* already exists the call will do nothing but will display an appropriate error message.

**Examples**

*-Rename|enjin|engine*

Renames the tag to correct the spelling mistake.

*-Rename|max-permitted-speed|extensions\max-permitted-speed-60850*

Renames the tag and moves it from the top level of config.txt to the extensions container.

*-Rename|queues\load0|load1*

Renames the container, note that containers cannot be moved by the *-Rename* command.

## -RepairTextures

Previous Top Next

**Format:**

*-RepairTextures*

*-RepairTextures|Texture*
*-RepairTextures|Size*
*-RepairTextures|Alpha*
*-RepairTextures|Uniform*
*-RepairTextures|Missing*
*-RepairTextures|Unreferenced*
*-RepairTextures|Duplicate*
*-RepairTextures|All*

This function attempts an automatic repair for all known CMP texture problems. The various classes of error can be dealt with by way of individual modules or the function can be asked to scan the asset and correct all of the errors found.

| Module | Process Description |
|---|---|
| *Texture* | Scans the asset for raw trainz *.texture files and converts them to a single tga (which may have an alpha channel) and a corresponding texture.txt file.<br><br>The original texture file is deleted. |
| *Size* | Checks that all textures are sized as powers of 2 or are 240 x 180 jpegs suitable for use as thumbnails.<br><br>This module will also detect and report the existence of multiple thumbnail images but is not able to correct this automatically. |
| *Alpha* | Scans the asset's texture.txt files for textures where the Primary and Alpha specify different files. Where these are found the two files are merged into a single tga with an alpha channel. The texture.txt file is amended to suit. |
| *Uniform* | Scans the asset for *.tga, *.bmp and *.jpg files which have no colour variation. Each file found is reduced in side and a small variation is introduced. The file will not be edited if it is part of a texture which also specifies another filename (since this would introduce another error). It is therefore useful to call *RepairTextures\|Alpha* before using this command. |
| *Missing* | Scans the asset for missing files. If the filenames are contained in the list below AssetX will create them in the asset folder using built-in defaults.<br><br>blank.tga<br>black.tga<br>blue.tga<br>green.tga<br>red.tga |

| | |
|---|---|
| | white.tga<br>yellow.tga<br><br>If the missing file cannot be automatically replaced its name will be reported when the script returns. |
| *Unreferenced* | This module reports any files which are not referenced, either as part of a valid texture.txt file or by way of a config tag.<br><br>In single script mode you will be asked if you wish to delete the unreferenced files, in batch mode they will simply be listed in the script output.<br><br>It will usually be safe to delete these files but you should first ensure that they are not used in html or by scripts.<br><br>If the selected asset is scripted or contains html files a warning will be issued. |
| *Duplicate* | Checks config.txt for references to image files which duplicate texture.txt references and corrects the config entries. |
| *All* | Calling *-RepairTextures* with no parameter will also select this option.<br><br>Carries out all of the above steps in the order listed in the table. Some modules may be inter-dependent, for example you should always call Alpha before calling Uniform since failing to do so may simply convert a '*uniform colour*' error into '*primary and alpha are not the same size'* |

## -RepairWaves

**Format:**

*-RepairWaves*

Searches for audio files (*.wav or *.mp3) within the Asset, re-encodes the file and writes it back to disc. In most cases this ensures that the file can be successfully loaded into TRS.

Since mp3 files are re-encoded to *.wav format and renamed the routine can be used to convert these files for use in game.

**Notes:**

All files are processed regardless of whether or not they are already valid in-game.

Sample and bit rates will be changed if they do not match common standards and storage formats are converted to a standard uncompressed PCM wave file format.

## -Replace

Previous Top Next

**Formats:**

*-Replace|Tag|Oldstring|Newstring*

Searches for the first instance of a specified substring within the named tag and replaces it with the new string.  If the tag does not contain the search pattern it will not be altered.

Note that both *Oldstring* and *Newstring* may occupy any position with the tag's value and may contain space characters.  Numerals are treated as text by this method.

Both Oldstring and Newstring are case sensitive unless *-IgnoreCase* has been used.

**Examples:**

*-Replace|description|Spotted|Striped*

Assuming that *description* is 'Lesser Spotted Woodpecker' this routine will change the value of the tag to 'Lesser Striped Woodpecker'.

Notes:

If *Newstring* is blank the existing text will be deleted and not replaced.

*-Replace|Username| 1996 Stock|*

Will change the username *'LT 1996 Stock'* to just *'LT'.*

**See Also**
-IgnoreCase
-ReplaceAll

## -ReplaceAll

Previous Top Next

**Format:**

*-ReplaceAll|Oldstring|Newstring*

Searches every tag in the config for the first occurrence of *Oldstring* and, if found replaces it with *Newstring*.  If no match is found nothing will be changed.

Only the first occurrence of *Oldstring* will be replaced in any tag and both strings are case sensitive unless *-IgnoreCase* has been used.

Note that both *Oldstring* and *Newstring* may occupy any position with the tag's value and may contain space characters.  Numerals are treated as text by this method.

**Examples:**

*-ReplaceAll|a.sit|a.stand*

Changes all matching attachment point names, for instance *a.sitpoint05* becomes *a.standpoint05*

*-ReplaceAll|church|synagogue*

Searches the entire config and changes any tag values (including values within nested containers) which contain the text *church* and changes them to *synagogue*,

**Notes:**

This routine should be used with care since it will change ALL instances within the config and may have unexpected results such as changing *Woodchurch Street Station* to *Woodsynagogue Street Station.* This will be a particular issue for assets with string tables.

You may be able to prevent this by including a space at the beginning and/or at the end of each substring specification.

**See Also**
-IgnoreCase
-Replace

## -Restore

Previous Top Next

**Format:**

*-Restore|n|Tag*

Recalls a value stored in script variable n and puts it into the specified tag's data.

**Example**

*-Restore|1|config.txt\trainz-build*

Gets the value stored in script variable 1 and puts it into the *trainz-build* tag data.

Quote marks are added to the data if it is a string. This command does nothing if the specified tag does not exist.

**See Also**

-Save
-Combine

## -Save

Previous Top Next

**Format:**

*-Save|n|Tag*

Saves the value of the specified tag as an unquoted string into script variable number *n*.

**Example**

*-Save|5|config.txt\trainz-build*

Saves the tag value for the tag trainz-build into script variable 5. Any quote marks are removed from the data. Does nothing if the tag does not exist.

**See Also**

-Restore
-Combine

## -SaveConfig

Previous Top Next

**Format:**

*-SaveConfig*

Performs an unconditional save of the present asset's config.txt.

**Example**

*-SaveConfig*

**Notes:**

In batch mode -SaveConfig is disabled, since each config is automatically saved at the end of its execution.

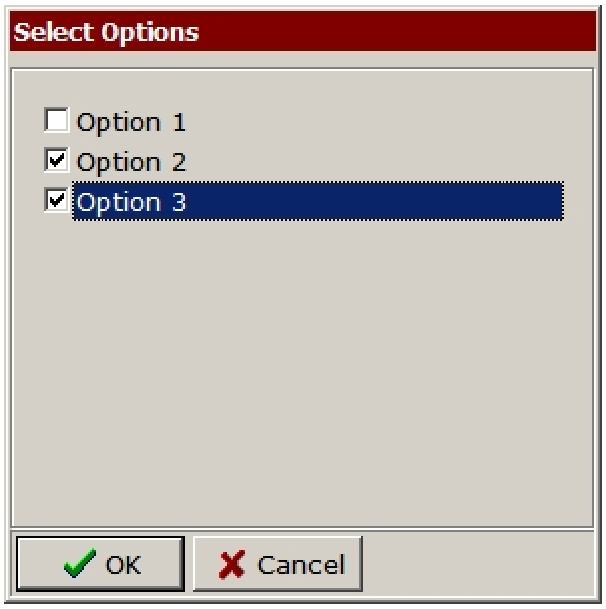
## -Select

Previous Top Next

**Format:**

*-Select|Option 1,Option 2,Option 3,...,Option n|Caption*

An interactive function which provides a dialogue box and requests selections from the user. This is similar to the -Case statement but code will be executed for every selected option.



**Example**

*-Select|Option 1,Option 2,Option 3|Select Options*

The first parameter is a comma separated list of options. The second (which is optional) provides a title for the dialogue. The script code below gives an example of how this function can be used.

```
-Select|Option 1,Option 2,Option 3|Select Options
; code here will be executed if the dialogue is cancelled, if there is no code
; in this section execution will continue after the -EndSelect statement.
  -IfQuery||Are you sure you want to cancel
    -Info|Script cancelled
    -Exit
  -EndIf

-Select|1
; If option 1 is checked then code within this section will be executed
; otherwise code here will be skipped
  -CallScript|option1.axm

-Select|2
  -MessageBox|Executing Option 2

-Select|3
  -If|kind=track
    -MessageBox|Executing Option 3
  -Else
    -MessageBox|Not Executing Option 3
  -EndIf

-EndSelect
-Info|Script completed
```

**Notes:**

- A maximum of nine choices are allowed.
- Each option string must be at least two characters in length.

- You can use *-If -Else and -EndIf* in each section but the entire construct must be balanced and closed within the same section.
- You can also use *-CallScript*, in which case the child script will be run and execution will continue after completion. The nested script will not be able to access the -Select statement parameters.
- You cannot use nested *-Select* statements.
- There must be a matching *-EndSelect* for every instance of a *-Select* block.

**See Also**

-EndSelect

## -SetEnv

Previous Top Next

**Format:**

*-SetEnv|VarName|Value*

Sets the Windows environment variable *VarName* to the *Value* specified.

*-SetEnv|VarName*

Clears the variable *VarName*.

**Example**

*-SetVar|EnvVar|3.14159*

Creates a new environment variable and assigns the value 3.14159 to it as a string. The value will persist until it is cleared or until you close the current AssetX session.

*-SetVar|EnvVar*

Clears the variable EnvVar.

**Notes**
This provides a method for saving a variable within a script file which can be accessed by other scripts or commands. Once EnvVar has been set it can be accessed using the macro syntax *%envvar%*.

Take care to ensure that you do not duplicate any of the predefined AssetX macros as these take precedence over system variables.

## -Silent

Previous Top Next

**Format:**

*-Silent|On*
*-Silent|Off*

Enables or suppresses messages from the script interpreter. User defined messages are not affected.

**Example**

*-Silent|On*
*-CallScript|sub\Subroutine.axm*
*-Silent|Off*

Calls Subroutine.axm and prevents the nested script from issuing error messages. Once the nested script has returned, error messages are re-instated.

**Notes:**

Use this switch to prevent error messages from appearing in the Outline alert box, commands may be issued as often as necessary and placed anywhere in the file. They can therefore be used to turn messages off for specific sections of the script.

This command is ignored for scripts running in batch mode.

The effect is global in the sense that this sequence:

*-Silent|On*
*-CallScript|ChildScript*
*-Silent|Off*

would prevent error messages from being issued by the child script.

Script messages are on by default and this state is reset after any script completes.

## -Sort

Previous Top Next

**Format:**

*-Sort*

Sorts the *config.txt* data into the order specified in *...\AssetX\bin\SortOrder.txt*.

*SortOrder.txt* defines a custom sort procedure for *config.txt* files. The file contains a list of tag names with an optional control character inserted anywhere in the list. Tags placed above the control character will be moved to the top of the tree, tags below it will be moved to the bottom.

Two control characters are supported:

| | |
|---|---|
| @ | simply defines the position at which the list is split, items not specified in SortOrder.txt will be left in their existing positions within the file. |
| > | causes config.txt to be alphabetically sorted before the custom ordering is executed. |

If no control character is used the tags in config.txt are moved to the top following the order defined in the file. The remainder are left unchanged.

**Example**

*-Sort*

**Note**

The sort order is applied whenever a new config is loaded into Asset·X, whenever this script function is run and whenever the user explicitly applies a sort.  The ordering of tags within containers is not affected by this function.

## -Store

Previous Top Next

**Format:**

*-Store|n|Value*

Stores the specified value as an unquoted string into script variable number *n*.

**Example**

*-Store|5|0.123*

Stores the value 0.123 into script variable 5 overwriting any previously saved value.

**Note**

To store the value of a tag use -Save

**See Also**

-Restore
-Combine
-Compute

## -TagForDeletion

Previous Top Next

**Format:**

*-TagForDeletion|Tag*

Marks the specified tag for deletion.  If the tag is a container all of its children will also be marked, continues if the tag is not found.

**Example**

*-TagForDeletion|config.txt\mesh-table*

**See Also**
-Delete

## -Tokens

Previous Top Next

**Format:**

*-Tokens|Data|Delimeters*

Tokenises the string Data using the characters in the Delimiters string as token boundaries. The result is stored in script variable locations 0 to 9.

Location 0 stores the number of tokens returned and locations 1 to 9 inclusive store the actual token values. Any existing values in the variable locations are overwritten.

A maximum of 9 tokens are extracted.

**Examples**

```
-Tokens|<kuid2:1234:5678:9>|<:>
-Info|^0 Tokens Found
-Info|1 = ^1
-Info|2 = ^2
-Info|3 = ^3
-Info|4 = ^4
-Info|5 = ^5
-Info|6 = ^6
```

The call above will produce the following output. Note that if the first character in Data is a delimeter then the first token returned will be an empty string. Similarly if the last character is a delimiter then the last token will be empty.

```
6 Tokens Found
1 =
2 = kuid2
3 = 1234
4 = 5678
5 = 9
6=
```

If the tag category-keyword contains the text loco;gwr;br then the following code will return 3 tokens (loco, gwr and br respectively).

```
-Save|0|category-keyword
-Tokens|^0|';
```

**Notes**
In the second example note the use of the quote ( ' ) character to escape the semi colon which would otherwise be interpreted as an inline comment. You can use any character for this purpose except white space, so select a character which will not occur in the data string.

*-Tokens* takes a literal string, a script variable or a % macro as its data parameter. To use the contents of a tag, save the tag value first and then pass the saved location as the first parameter, as in the second example.

## -Update

Previous Top Next

**Format:**

*-Update|Tag or Container|data*

Puts new data into the specified tag if the tag exists. If the tag does not exist, it is created and the data added. If necessary -Update will create the entire path required to create a tag address. If there is no supplied data and the tag does not exist a container is created.

**Examples**

*-Update|config.txt\trainz-build|2.5*

Creates a *trainz-build* tag if it doesn't exist and sets the value to 2.5. A line such as this can be used in asset upgrading macros.

Don't forget the vertical pipe symbol ( | shift-backslash) as the field separator.

**Creating Containers**

*-Update|config.txt\mesh-table\default\effects\0\kind|corona*

Creates the tag kind "corona" and in doing so creates all of the necessary containers and sub-containers. The above command would produce a config.txt structure looking like this:

```
mesh-table {
    default {
        effects {
           0 {
               kind "corona"
           }
        }
    }
}
```

**Note**

If the tag data includes any pipe characters '|' beyond the number required to separate the script method parameters these will be appended to the tag. This is primarily intended to copy ACS tags correctly:

*-Update|extensions\active-coupling-standard-60850\front\coupler|screwlink|hook*

will result in this

```
extensions {
    active-coupling-standard-60850 {
        front {
           coupler "screwlink|hook"
        }
    }
}
```

-UpdateAssetFilename

Previous Top Next

**Format:**

*-UpdateAssetFilename*

A routine which scans *config.txt* for *asset-filename* tags.

Asset-Filename has been obsolete since the issue of TRS2006 but is still frequently encountered.  The tag affects the location of meshes, art-icons and alpha-numbers.

This procedure will create *mesh-table* and *thumbnails* containers as necessary as well as actually deleting the *asset-filename* tag itself.

**Example**

*-UpdateAssetFilename*

## -UpdateCamera

Previous Top Next

**Format:**

*-UpdateCamera*

Updates the cameradefault tag and the cameralist container for interior assets by ensuring that:

- Cameralist contains a list of camera positions starting at camera0 and correctly serialised.
- Each camera# tag consists of five float values. where fewer than five values currently exist, zero is appended.
- Cameradefault references a valid camera position.

Where there is insufficient data to complete the task an error is entered into the log file.

## -UpdateCategoryEra

Previous Top Next

**Format:**

*-UpdateCategoryEra*

A high level instruction that calls a routine which scans *config.txt* for old style category-era-0 ,1 ,2 etc entries, and consolidates the data into one entry of the type *category-era*. The function verifies values from the standard list of entries.

**Example**

*-UpdateCategoryEra*

## -UpdateCategoryRegion

Previous Top Next

**Format:**

*-UpdateCategoryRegion*

A high level instruction that calls a routine which scans *config.txt* for old style *category-region-0 ,1 ,2* etc entries, and consolidates the data into one entry of the type *category-region*. The function verifies values from the standard region list.

**Example**

*-UpdateCategoryRegion*

## -UpdateKUIDTable

Previous Top Next

**Format:**

*-UpdateKUIDTable*

A high level instruction that calls a routine which firstly scans *config.txt* for single number KUID's from version 1.0, and converts them to <kuid:xxxx:yyyy> format. It then re-scans for kuid references to build the kuid-table. If the kuid-table container does not exist, one is created. Any kuid-table entries with string names are retained whilst the remainder are numbered 0, 1, 2 etc.

Whilst this function will normally give accurate results it does not delete existing entries and does not differentiate between kuid and kuid2 formats.  To be certain that any redundant entries are cleared you will need to manually delete all entries which are not required by scripts before running this function.

**Example**

*-UpdateKUIDTable*

## -UpdateUserData

Previous Top Next

**Format:**

*-UpdateUserData*

Another high level instruction that calls a routine to load the user's own data into his content. If the content does not belong to the user nothing is done. The instruction updates the entries for *author, email, website, license, organisation and info-url* in *config.txt*, using the data contained on the *Options/User Data* dialogue. If any of these user data fields are blank, the corresponding tags will not be included. Any missing tags are created as necessary.

**Example**

*-UpdateUserData*

## -UpdateUsername

Previous Top Next

**Format:**

*-UpdateUsername*

Deletes obsolete *name* tags (including localised versions such as *name-fr*) and replaces them with *username.*

**Example**

*-UpdateUsername*

## -Write

Previous Top Next

**Format:**

Writes text to a file on disc.

| Command Parameters | |
|---|---|
| *-Write\|Open\|file* | Creates a new text file and opens it for writing. If a file of the same name exists it will be overwritten. |
| *-Write\|Append\|file* | Opens an existing text file and prepares it to allow new text to be appended. |
| *-Write\|Line* | Writes an empty line to an open text file. |
| *-Write\|Line\|text* | Writes text to an open text file and adds a carriage return and line feed. |
| *-Write\|Char\|text* | Writes text to an open text file. This command does not add carriage returns or line feeds. |
| *-Write\|Escape\|Text* | This is the same as -Write\|Char except that AssetX delimiters, macros and Script Variables are not expanded. You will need to use this parameter if you are writing quotes or ( ; \| ^ % ) characters to the file. Escape does not add line feeds so you will normally call -Write\|Line immediately after using this call. |

| | |
|---|---|
| | You cannot use inline comments with this parameter since a semi colon is interpreted literally. |
| *-Write\|Close* | Closes an open file and saves it to disc. |

*File* can be either a fully specified pathname including a drive letter or a simple name and filetype. In the latter case the file will be saved to the current asset.  For the Append parameter the file must already exist.

You can use AssetX macros and script variables within the filename and text parameters and within the text strings you are writing.

You must call *-Write|Close* to complete the operation.  If you fail to do so the file will not be saved or updated.

## Asset·X Macros

Previous Top Next

A listing of Asset·X macros. These are shortcuts used extensively by the software to assist with portability of code and configuration data. They can be used in Scripts and Tool Definitions to reduce typing and simplify sharing.

Operating system environment variables such as *%programfiles%*, *%appdata%* and *%systemroot%* can also be used. These are evaluated after the AssetX macros so if there is a naming conflict the AssetX macro will be preferred.

- Path specifications always exclude any trailing backslash.
- Extension specifications include the initial period.
- Macros are returned as strings and are not case sensitive.

**Note**
In the table below 'selected file' means the file which is currently highlighted in the File List pane.

| Asset·X Macro | Contents |
|---|---|
| *%pfe%* | Path filename & extension of selected file. |
| *%pf%* | Path & filename only of selected file. |
| *%p%* | Path to selected file (asset folder). |
| *%fe%* | Filename & extension of selected file. |
| *%f%* | Filename only (extension omitted) of selected file. |
| *%e%* | Extension only of selected file. |
| *%kuid%* | KUID of the currently selected asset. |
| *%kuidf%* | KUID of the currently selected asset in the format *kuid 1234 5678* or *kuid 2 1234* |

| | |
|---|---|
| | *5678 9*, suitable for use in file or folder names. |
| *%app%* | The installation path for Asset·X. |
| *%assoc%* | If defined, the associated windows application for selected file. |
| *%backup%* | Path to Asset·X\backup folder. |
| *%bin%* | Path to Asset·X\bin folder. |
| *%log%* | Path to Asset·X\log folder. |
| *%scripts%* | Path to Asset·X\scripts folder. |
| *%tetdata%* | Path to the lists folder of the current trainz version. |
| *%trainz%* | Path to default TRS installation root. (the version of TRS which Asset·X is currently using) |
| *%trs%* | Current TRS version. eg 'TRS2004', 'TC', 'TS2010'. |
| *%build%* | Returns the trainz build number that Asset·X is currently using. |
| *%TS12%* or *%2012%* | Path to TS12 root, if defined in your AssetX.ini file. |
| *%2010%* | Path to TS2010 root, if defined in your AssetX.ini file. |
| *%2009%* | Path to TS2009 root, if defined in your AssetX.ini file. |
| *%TC%* | Path to TC root, if defined in your AssetX.ini file. |
| *%2006%* | Path to TRS2006 root, if defined in your AssetX.ini file. |
| *%2004%* | Path to TRS2004 root, if defined in your AssetX.ini file. |
| *%pev%* | Path to PevSoft tools root folder, if defined in your AssetX.ini file. |
| *%userid%* | Returns your own TRS user ID as recorded in Options/User Data |
| *%assetid%* | The user ID of the author of the currently selected asset. |

## Keyboard Shortcuts

| Key | Application | Asset List | File List | Outline | Mesh Viewer | Text Edit |
|---|---|---|---|---|---|---|
| Esc | Cancel Scan | | | Cancel Find | | |
| Space | | Toggle Tag | Toggle Tag | Toggle Tag | | |
| Insert | | | | Insert Tag | | |
| Delete | | Delete | Delete | Delete | | |
| ^Delete | | Delete Tagged | Delete Tagged | Delete Tagged | | |
| Return | | | | Tag Data | | |
| ^Return | | | | Edit Tag | | |
| ^F | | | | Find First | | Find |

| Key | Application | Asset List | File List | Outline | Mesh Viewer | Text Edit |
|---|---|---|---|---|---|---|
| ^I | | Invert Tags | Invert Tags | | | |
| ^K | | | | Undo Mark | | |
| ^O | Open Assets | | | | | |
| ^R | Replicate | | | | | |
| ^S | Save File | | | | | Save File |
| ^T | | Tag All | Tag All | Tag All | | |
| ^U | | Untag All | Untag All | Tag All | | |
| ^Z | | | | Undo | | |
| Keypad + | | | | Expand | | |
| Keypad - | | | | Collapse | | |
| Keypad * | | | | Expand All | | |
| Home | | First Item | First Item | First Item | | |
| Up | | Previous Item | Previous Item | Previous Item | Rotate Up | |
| Left | | | | Previous Level | Rotate Left | |
| Right | | | | Next Level | Rotate Right | |
| Down | | Next Item | Next Item | Next Item | Rotate Down | |
| End | | Last Item | Last Item | Last Item | | |
| ? | | | | Help on Tag | | |
| ^Up | | | | | Pan Up | |
| ^Left | | | | | Pan Left | |
| ^Right | | | | | Pan Right | |
| ^Down | | | | | Pan Down | |
| Shift-Up | | | | | Zoom In | |
| Shift-Down | | | | | Zoom Out | |
| F1 | Help | | | | | |
| F2 | | | | Edit Item | | |
| F3 | | | | Find Next | | Find Next |
| F5 | Focus Assets | | | | | |
| F6 | Focus Files | | | | | |
| F7 | Focus Outline | | | | | |
| F8 | Focus Viewer | | | | | |
| ^F2 | Load Home Folders | | | | | |
| ^F4 | Load 2004 \Custom | | | | | |
| ^F6 | Load 2006 \Editing | | | | | |
| ^F8 | Load TC\Editing | | | | | |
| ^F9 | Load 2009 \Editing | | | | | |
| ^F10 | Load 2010\Editing | | | | | |
| ^F12 | Load 2012 \Editing | | | | | |