DSON Editor

Manual Version 1.11

Content

Ralf Sesseler

Dimension 3D

E-Mail: d3d@sesseler.de
Internet: d3d.sesseler.de

Dimension 3D

Important Information

Copyright

Acknowledgment

DSON Editor is based on wxWidgets.

Disclaimer

There is no warranty beyond the legal minimal warranty. In no case, the author shall be liable for any damage on hardware or software caused by using *DSON Editor*.

1. Overview

The *DSON Editor* is an editor for DSON files, which define the content and scene data for DAZ Studio 4 and up. It presents the content in a hierarchical tree structure for objects and arrays. It provides basic editing capabilities like direct inline editing and a sophisticated search as well as special editors and tools to facilitate working with particular kinds of content.

To motivate the structure of the editor, this manual starts with a short overview of the DSON file format and how it is supported in the *DSON Editor*. The remainder of the manual describes the user interface, the various tools and editors, and the configuration of *DSON Editor*.

Note: Some basic understanding of the DSON format is required to work with *DSON Editor*.

2. The DSON Format

The *DAZ Scene Object Notation* (DSON)[1] file format is used by DAZ Studio 4.0 and up to store data for all kind of scene objects and related information. It is based on the *JavaScript Object Notation* (JSON)[2] format. The *DSON Editor* displays the content of DSON files in a hierarchical tree that is based on the structure of the DSON format.

This format consists of the following data types:

- *Object*: a list of key/value pairs; the key is a string, the value may be of any type
- *Array*: an ordered list of values of the same type
- *String*: text
- *Number*: a numeric value
- *Boolean*: *true* or *false*
- *Null*: no value

To improve reading and editing, *DSON Editor* uses an additional type *Vector* for short arrays of numbers. This is useful because 3D applications use many of these vectors, e.g. for colors, deltas, or coordinates.

DSON defines a special object for arrays with a counter. This object has the key "count" with the number of elements and the key "values" with the actual array. *DSON Editor* recognizes these objects and automatically adjusts the counter when the number of elements in the array changes.

Each DSON file consists of one single object. By nesting objects and arrays as values inside of other objects and arrays, the resulting structure may be arbitrarily large. Objects have a predefined structure that define the valid keys and the types of their values.

Note: If applicable, all editors and tools are based on file version 0.6.0.0 for DSON files.

Note: Because DSON files are a special case of JSON files, any JSON files may be opened and edited with *DSON Editor* as well.

1 http://docs.daz3d.com/doku.php/public/dson_spec/start

2 http://www.json.org/

Representation as a Tree

When opened in a text editor, a JSON file may look as follows:

```
{
      "text" : "whatever",
      "number" : 1.0,
      "choice" : true,
      "empty" : null,
      "numbers" : [1,2,3],
      "words" : ["one" , "two"],
      "object" : {"key" : "value"}
}
```

The curled brackets denote objects, the square brackets arrays. The comma separates elements in objects and arrays, the colon separates key and value in an object. The quotation marks enclose strings. Line ends are arbitrary.

In *DSON Editor*, this looks as follows:

Key	**Type**	**Value**
{}	object	
text	string	whatever
number	number	1.0
choice	boolean	true
empty	null	null
numbers	vector	[1, 2, 3]
words	array	
[1]	string	one
[2]	string	two
object	object	
key	string	value

The tree representation in *DSON Editor* helps to structure the content more readable thanks to indention. It allows to collapse objects and arrays, which is in particular useful for large arrays with hundreds or thousands of elements. The elements of arrays may be enumerated. Keys, types, and values are aligned in columns. Finally, the editor guarantees syntactical correctness for all changes.

2.1. Strings

All strings in DSON are UTF-8 encoded, so the files can be stored using a 8-bit character set and can be edited with any regular text editor. To prevent conflicts between the notation of DSON and strings, the following escape characters have to be used inside of strings:

- \\: a single backslash
- \": a quotation mark
- \t: tabulator
- \r: carriage return
- \n: new line
- \f: form feed
- \b: backspace

DSON Editor displays and allows editing strings in Unicode. However, it resolves escape characters only in a special text editor, because they are rarely used in DSON files.

Uniform Resource Identifier

For file names, ids, and references, DSON uses the Uniform Resource Identifier (URI)[3] format with its own interpretation. The URI scheme in DSON is as follows:

```
scheme://target#asset/node#asset:file#asset?property
```

In most cases, only partial schemes are used. The *DSON Editor* has a special editor for URIs that splits the URI in its parts.

As part of the URI format, these strings use a very limited character set with letters, numbers, and a few other signs only. Anything else must be percent-encoded[4]. All editors in *DSON Editor* that are made for files, ids, or other URIs will care about the correct encoding. In addition, there is an option to resolve the percent-encoding to display URIs more readable in the editor. To indicate that a string is percent-encoded, the type columns shows *string %* in that case.

Important: Even when percent-encoding is resolved in display, it is not for inline editing and search.

Note: There is no way to distinguish between percent-encoded strings and strings that only happen to include %XX, which is the format for percent-encoding. But the latter should be rare.

File Names

File names are a special case of URI. They don't require any of the separating characters of the URI scheme, but they are percent-encoded. For platform independence, the path separator is always the slash character /. The path should always start with a slash. If possible, the path is relative to one of the content folders of DAZ Studio.

If applicable, *DSON Editor* will take care about converting file path and names to the correct format, i.e. to DSON format inside of DSON files and to full path in OS-dependent format otherwise.

2.2. The DAZ Object

The one main object, of which each DSON file consists, is called *DAZ object* and has a fixed structure. It starts with a header consisting of the *file version* number and the *asset info*. The file version is a string with four numbers separated by points.

The asset info is an object with generic information about the file. It starts with the *id*, which is the same as the file name in URI format. Next is the *type* denoting the kind of content in the file, e.g. a scene, figure, or material. The *contributor* is an object with the author and its email and website. Finally, there is a *revision* number and the *modified* date[5]. *DSON Editor* has a special editor for the header called *asset editor*.

3 http://en.wikipedia.org/wiki/Uniform_resource_identifier

4 http://en.wikipedia.org/wiki/Percent-encoding

5 The date is simply a string, but should be in ISO 8601 extended format.

Next come optional *libraries* for geometries, nodes, uv-sets, modifiers, images, and materials. Each library is an array with elements of the according type.

Finally, there is an optional *scene* object with the elements of a full or partial scene. Again, this consists of arrays with nodes, uvs, modifiers, materials, animations, and other scene elements or settings.

Object Definitions

There are too many different kinds of predefined objects in the DSON specification to name them here. You can find the definitions at the DAZ documentation center:

http://docs.daz3d.com/doku.php/public/dson_spec/object_definitions/start

Unfortunately, these definitions may differ for file version 0.5.0.0 used by DAZ Studio 4.0 and file version 0.6.0.0 used since DAZ Studio 4.5. If editors in *DSON Editor* are based on object definitions, they require the format for file version 0.6.0.0.

3. User Interface

The user interface of *DSON Editor* consists of a tool-bar at the top, a hideable search-bar below it, a file-list to the left, the tree representation of the current file in the center, and a status-bar at the bottom.

The tool-bar (see section 3.3) contains all user actions, either directly or by opening a menu. Many actions can be accessed also by the context menu in the tree representation or by keyboard shortcuts. In the search-bar (see section 4), the user specifies what to search and where to search. It is shown or hidden by the search button in the tool-bar or with Ctrl-F.

The file-list (see section 3.2) shows all files that are currently open. Several files may be selected in the file-list, because some tools and operations work for several files at once. The currently selected file is shown in the tree representation (see section 3.1), where it can be edited in many different ways.

The status-bar is divided into two parts. On the left is the full path and name of the currently selected file. The help text for the tool-bar is also shown here. On the right is the notification area, where operations like search display information about their results.

3.1. Tree Representation

The main area of *DSON Editor* is the tree representation of the currently selected file, which is marked in the file-list by a red frame. It has three columns: *Key*, *Type*, and *Value*.

Under *Key*, you find the key for elements in objects, {} for the root object, or [] for elements in arrays. For arrays, the elements may be enumerated optionally. Objects and arrays display their elements as a hierarchical tree structure that may be collapsed and expanded by the plus/minus sign in front of each object and array.

The *Type* column shows the type of the element: *object*, *array*, *string*, *number*, *vector*, *boolean*, or *null*. The type can't be changed, except that *string* and *null* can be converted to each other. (It seems to be common to display an empty string as null, but it is not mandatory.) If the type is *string %*, a percent-encoded string is shown resolved to Unicode in the *Value* column.

Finally, the *Value* column shows the value of the element. It is empty for objects and arrays because their values are the elements that are shown as their children in the tree structure. Optionally, the value column may show the identifier for an object or array, which is the value for one of the following keys: *id*, *url*, *type*, *channel*. (Under Windows, these identifiers are italic.)

A line in the tree is selected by clicking on it. It is also possible to select several lines with the shift or control key. Some actions are possible only for a single selected line. A right click on a line opens the context menu with actions that can be performed for that line or for the selected lines. Click twice on a key or value to edit it with the inline editor.

3.2. File-List

The file-list shows the thumbnail (if it exists) and the name of all opened files. The currently selected file shown in the tree representation is marked by a red frame. If a file was modified, it is marked by a yellow frame. (For the current file, the *save* button in the tool-bar indicates if it was modified or not.) You set a file as current file by clicking on it. Several files may be selected for actions that are performed on all selected files.

A right click on a file opens the context menu. (Due to implementation reasons, this will also select the file.) The context menu has the following actions:

- *Save*: saves the selected file (for single selection)
- *Save as...*: saves the selected file with a new path/name (for single selection)
- *Close*: closes the selected file (for single selection)
- *Save selected*: saves all modified selected files (for multiple selection)
- *Close selected*: closes all selected files (for multiple selection)
- *Uncompressed*: marks the selected file(s) to save without compression
- *Compressed*: marks the selected file(s) to save with compression
- *Select all*: selects all files
- *Small icons*: switches to small icon view (Windows only)
- *Large icons*: switches to large icon view (Windows only)

If a single file is selected, a check-mark indicates if it will be saved with or without compression. Initially, this is how the file was loaded or for new files how it is set in the options.

3.3. Toolbar and Menus

The toolbar is divided into five sections with the following buttons:

- Current File
  - *New*: creates a new file
  - *Open*: shows a file selection dialog to open an existing file
  - *Recent*: shows a menu to select one of the most recent used files
    (press shift to keep the menu open when selecting a file)
  - *Save*: saves the current file
  - *Save as*: saves the current file with a new path/name
  - *Close*: closes the current file
- Multiple Files
  - *Save all*: saves all modified files
  - *Close all*: closes all files
- Undo/Redo
  - *Undo*: undoes the last action
  - *Redo*: restores the last undone action
- Tools
  - *Edit*: shows the edit menu (see below)
  - *Insert*: shows the insert menu (see below)
  - *Tool*: shows the tool menu (see below)
  - *Search*: shows or hides the search-bar (see section 4)
- Application
  - *Options*: opens the options dialog (see section 7.1)
  - *Folders*: opens the content folders dialog (see section 7.2)
  - *Info*: opens the info dialog
  - *Help*: opens the PDF manual
  - *Exit*: closes the application

Note: You can change the size of the tool-bar buttons and if text is shown in the options.

Insert Position

Because of the tree structure, new lines can be inserted either *before* or *after* the currently selected line. For objects and arrays, it's also possible to insert a new line as *first* or as *last* child. Thus, all actions to insert lines will have the options *before*, *after*, *first*, and *last* if applicable.

Edit Menu

The edit menu is context-sensitive to the current selection in the tree representation. It is identical to the context-menu shown by right-clicking on a line in the tree. Similar context menus are also used in the animations editor and the value editors.

The edit menu has the following sections and actions:

- type or value dependent
  - *File...*: opens a dialog to select a file (string only)
  - *File/ID editor*: opens the ID editor (string only)
  - *URI editor*: opens the URI editor (string only)
  - *Text editor*: opens the text editor (string only)
  - *Convert to 'null'*: converts string type to null (string only)
  - *Convert to 'string'*: converts null type to string (null only)
  - *Select color*: opens a color selection dialog (only for vectors with three elements and values between 0 and 1)
  - *Set to 'false'*: sets a boolean value from true to false (boolean only)
  - *Set to 'true'*: sets a boolean value from false to true (boolean only)
  - *Set to '**n**'*: sets the value for a *count* key to the number of elements in the *value* key (shown only for counted arrays and if it is not the correct value)
  - *Animations editor*: opens the animations editor (for scene/animations or lines below)
  - *Formulas editor*: opens the formulas editor (for formulas)
  - *Asset editor*: opens the asset editor (for the asset_info object)
  - *Import geometry*: imports geometry from an OBJ file (below geometry_library only)
  - *Export geometry*: exports geometry as an OBJ file (below geometry_library only)
  - *Import UV-mapping*: imports UV from an OBJ file (below uv_set_library only)
  - *Create morph*: creates morph from two OBJ files (below modifier_library only)
  - *Calculate morph*: opens the dialog to recalculate a morph (below modifier_library only)
- generic editing
  - *Cut*: cuts the selected lines and places them in the clipboard
  - *Copy*: copies the selected lines to the clipboard
  - *Paste*: pastes the clipboard *before*/*after*/*first*/*last/replacing* the current line (available if clipboard contains text, but successful only if text is formatted for DSON)
  - *Duplicate*: duplicates each selected line and inserts it after its original line
  - *Delete*: deletes the selected lines
  - *Delete all children*: deletes all elements of objects or arrays
- *Insert*: inserts a new line with the according type and a default value *before*/*after*/*first*/*last* the current line. For *strings*, a value can be entered directly with one of the file/ID/URI editors. Also, the *color* editor can be used to directly give a value for a vector. Finally, a *counted array* can be inserted, which is an object with a *count* number and a *values* array.
- *Tree*: for objects and arrays with elements, the next level or all levels may be expanded or collapsed

For animations editor and value editors, the context menu also allows to *insert* a new value or to *remove* an existing one.

Insert Menu

The *insert* menu allows to define and insert arbitrary fragments of DSON files. You may think of this as a permanent clipboard, where you can paste everything once stored.

With *save*, the selected lines are saved as a fragment. For example, you may save *file_version* and *asset_info* as *header*, so you can easily add the typical header to each file you create newly. *Delete* shows a list with all fragments, selecting one means to delete it after confirmation. *Refresh* will rebuild the list of fragments, but this should be necessary only in case you changed the files with the fragments manually. Finally, you can *insert* each fragment *before*/*after*/*first*/*last* the currently selected line.

Note: For implementation reasons, the number of fragments is limited to 500.

Tools Menu

The tools menu has two sections. The tools at the top are either applied to the current document or to all documents selected in the file-list. The other tools can be used with the current document only.

- *File references*: opens the ID editor with all file references
- *IDs*: opens the ID editor with all identifiers
- *URLs*: opens the ID editor with all URL references
- *Asset editor*: opens the asset editor
- *Value editor*: opens the value editor for the following objects (if existing)
  - *Node*: nodes in node library
  - *Node channel*: channels for nodes in node library
  - *Node channel (extra)*: extra channels for nodes in node library
  - *Material channel*: channels for materials in material library
  - *Material channel (extra)*: extra channels for materials in material library
  - *Scene node*: nodes in scene/nodes
  - *Scene node channel*: channels for nodes in scene/nodes
  - *Scene modifier*: modifiers in scene/modifiers
  - *Scene modifier channel*: channels for modifiers in scene/modifiers
  - *Scene modifier channel (extra)*: extra channels for modifiers in scene/modifiers
  - *Scene material*: materials in scene/materials
  - *Scene material channel*: channels for materials in scene/materials
  - *Scene material channel (extra)*: extra channels for materials in scene/materials
- *Animations editor*: Opens the animations editor (if animations exist)

The animations editor and the value editors can be used for the full document or for selected lines only. Without selection or if an arbitrary line is selected, they use the full document. All editors will be shown in the menu even if they are not applicable to the document. If you select a tool but no editor opens, this implies that the document or the selection contains nothing that would be shown in the editor.

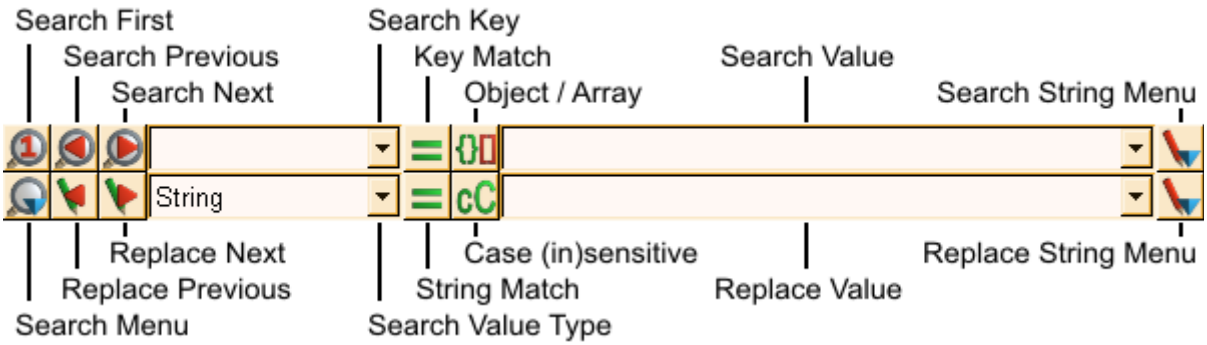
3.4. Keyboard Shortcuts

The following keyboard shortcuts are defined:

File		*Application*		*Search*	
Ctrl-N	new file	F1	help	Ctrl-F	show/hide search
Ctrl-O	open file	Shift-F1	info	F3	search next
Ctrl-S	save file	F2	options	Shift-F3	search previous
Ctrl-Shift-S	save file as	Shift-F2	content folders	Ctrl-F3	search first
Ctrl-A	save all	Ctrl-Q	exit	F4	replace next
				Shift-F4	replace previous
Edit					
Ctrl-X	cut	Ctrl-C	copy	Ctrl-Z	undo
Del	delete	Ctrl-V	paste after	Ctrl-Shift-Z	redo
Shift-Del	del all children	Ctrl-Shift-V	paste first	Ctrl-Y	redo
		Ctrl-D	duplicate		

4. Search

The search bar is opened and closed with the *search* button in the menu bar or with the key Ctrl-F. Search is adjusted to the particular structure of DSON files. You may search for values of specific types, inside of objects and/or arrays, for particular keys, and even in several files at once.



The search bar consists of the following elements:

- *Search first*: searches for the first matching line in the current document
- *Search previous*: searches upwards for the next matching line in the current document
- *Search next*: searches downwards for the next matching line in the current document
- *Replace previous*: replaces in the current line and then searches for the previous match
- *Replace next*: replaces in the current line and then searches for the next match
- *Search menu*: opens the search menu

    - *Select all*: selects all matches in the current document
    - *Select parent*: selects the parents for each match in the current document
    - *Search all*: searches all matches and shows them in the report window
    - *Search in selected files*: searches all matches in all selected files
    - *Search in all files*: searches all matches in all opened files
    - *Delete all*: deletes all matches
    - *Delete in selected files*: deletes all matches in all selected files
    - *Delete in all files*: deletes all matches in all opened files
    - *Replace all*: searches all matches and replaces them
    - *Replace in selected files*: replaces in all selected files
    - *Replace in all files*: replaces in all opened files
    - *Close report window*: closes the report window
    - *Report replacements*: if checked, replacements are shown in the report window
    - *Reset selection*: if checked, any existing selection is reset before selecting matches
    - *Apply to parent*: if checked, delete and select are applied to the parent of matches
- *Search key*: the key in an object to search for (when searching in objects only)
- *Key match*: full or partial match for search key (when searching in objects only)
- *Object / Array*: search values in objects, arrays, or both
- *Search value type*: the data type to search for
- *String match*: full or partial match for search value (when searching a string value)
- *Case (in)sensitive*: ignoring case or not (when searching a string value)
- *Search value*: the value to search for
- *Replace value*: the value to replace by
- *Search string menu*: string encoding and file selection for the search value field
- *Replace string menu*: string encoding and file selection for the replace value field

All input fields remember the ten last used terms. You can select them from the drop-down list. This list also has an empty field to clear the field, either by selecting it or by pressing the down arrow. The value fields have own lists for each value type.

4.1. Search Modes

The simplest search mode is to search in the current document and to select each found match in the tree. This is done by the first three buttons *search first*, *search previous*, and *search next*. With these buttons, you can search step by step downwards or upwards in the current document. *Replace previous* and *replace next* do the same as *search previous* and *search next*, but additionally do the replacement in the currently selected line. That way, you can decide whether to replace or not simply by using the *search* or the *replace* button while searching through the document.

The search menu provides further search modes. With *select all* and *select parents*, all matching lines are selected in the current document. Any previously selected lines stay selected, so you can build your selection from several searches.

Note: Selecting lines by search is useful for editors that depend on the selection. For example, you may search for objects with a particular id by using "id" as key and the actual id as value, then use *select parents*, and all objects with that id are selected and hence shown when using the editor for objects of that type.

Finally, you may use search and replace on the full document or even on several selected or all documents. For search, the result is shown in the report window that opens below the tree window. For replace, this is done only if *report replacements* is checked in the search menu.

While the report window is open, you can select a line to show the match or replacement in the according document. To close the report window, double click on the divider bar or use *close report window* from the search menu.

The result of each search operation is shown in the notification part of the status bar. This is either a simple *found* or *not found*, or the number of found matches or replacements.

Search Scope

In DSON, values may be either part of an object or of an array. The object / array button decides whether to search for values in objects only, in arrays only, or in both (i.e. everywhere).

When searching in objects only, it's possible to name a particular key to search for. This search is always case-sensitive, but you may select whether to search for the complete term or to do a partial match. The search value may be empty in this case to search for the key only.

4.2. Search Data Type

Because the values in DSON have a data type, searching for a value is limited to a particular type, which is selected in the *search value type* drop-down list.

String

Searching for a string may be case-sensitive or insensitive and it may be a full or partial match. When replacing a string with partial match, each occurrence of the search term is replaced.

Note: Case-insensitive search is limited to ACSII-letters A-Z.

Because there are some special aspects about strings (see section 2.1), both input fields have a menu with utility features. On the one hand, the search term may be percent-encoded with *encode %* and may be converted to use escape characters with *encode* \. This is needed because search operates on the string values as they actually are. On the other hand, the editors for files, ids, and URI can be used to enter values without encoding and get the conversion automatically.

Number and Vector

Vectors are short arrays of numbers, so search for numbers and vectors are related. There are four options:

- *Number*: searches for a number as value in objects or arrays, but not in vectors
- *Number or vector*: searches for a number in objects, arrays, and inside of vectors
- *Number in vector*: searches for a number inside of vectors only
- *Vector*: searches for a complete vector

Numbers are matched with the tolerance given in the options. So, if the tolerance is 0.01 and you search for 1, 1.005 and 0.999 will match as well. To search for exact values without tolerance, put == in front of the number. The relational operators != (not equal), < (less than), > (greater than), <= (less or equal), and >= (greater or equal) may be used as well.

The same holds for numbers inside of a vector. In addition, the question mark ? may be used as a wildcard to match any value. For example, [1 , ? , < 0.5] matches with all triples, where the first number is 1 (within the given tolerance) and the third number is less than 0.5.

Boolean

The data type boolean only allows *true* and *false* as values. For convenience, these values may be abbreviated as 0/1, t/f, or T/F.

Other Data Types

The other data types *null*, *object*, and *array* can't be used with values. The same holds for *any*, which matches with all data types. These options are useful mainly when searching for a key in objects.

5. Editors

Editors are sensitive to the type of the value that is edited. They allow direct input or have specific features for particular kinds of values.

5.1. Inline Editor

The inline editor appears in place in the tree when you click twice on a key or a value. You simply type the new value or modify the existing and hit return or click outside the editor with the mouse to accept the change. Hitting escape closes the editor and discards any changes. The inline editor only accepts syntactically correct input and ignores anything else.

Valid input depends on the type:

- *String*: For strings, no encoding except Unicode is used. Percent-encoding is displayed as it is and escape characters must be used with the leading backslash.
- *Number*: Numbers require dot as decimal point. Also, the notation with e for exponents of 10 like 1.2e3 for 1200 or 1.2e-3 for 0.0012 is understood.
- *Vector*: Vectors accept any sequence of numbers separated by commas. The square brackets around the vector may be used or missing.
- *Boolean*: Besides true and false (in any case spelling), t/f, T/F, and 1/0 are accepted.

Important (Mac OS X): Because of implementation reasons, the inline editor on Mac OS X is actually realized by a dialog editor that opens with a double-click.

5.2. String Editors

Because strings have different encodings and special formats for files, ids, and urls, there are several string editors for different cases.

File...

Not really an editor, but a file selection dialog that automatically converts the selected file to the format used for files in DSON. Path separators are OS-independent and a relative path is used if possible.

File/ID Editor

This editor has a single input field with the file or id. The *file* button opens a menu with utilities to work with files:

- *Absolute path*: converts the path to absolute (if possible)
- *Relative path*: converts the path to relative (if possible)
- *File...*: opens a file selection dialog to replace the file
- *Open file*: uses the system default application to open the file
- *Open folder*: uses the system default application to open the folder
- *Copy path*: copies the absolute file path in OS-dependent format to the clipboard

URI Editor

The URI editor divides a string according to the URI scheme used in DSON:

```
scheme://target#asset/node#asset:file#asset?property
```

This scheme is also shown on top of the editor. For each part of the scheme, there is an input field. The *file* button provides the same utilities as in the *File/ID editor* and is applied to the *file* field.

Note: The *target* may start with an @, which is not a part of the scheme, but nonetheless is not percent-encoded.

Text Editor

This editor is intended for pure text like comments. It resolves all escape characters and allows to use new lines and tabs in the text. When clicking OK, all characters are converted to escape characters if required.

5.3. Color Editor

To be independent of the object definitions in DSON, the color editor is simply available for all vectors with three elements, if all numbers are between 0 and 1. The color editor is the common color selection dialog.

6. Tools

The tools are special editors that rely on particular object definitions in DSON. Hence, they are only applicable if the current document contains some of these objects.

Note: All changes in the tool editors are collected into a single undo step.

Note: You can limit the *value editors*, the *animations editor,* and the *formulas editor* to particular objects by selecting them in the tree before opening the editor.

6.1. File References / IDs / URLs

These three editors are essentially the same, they only differ in the content they display. The *file references* editor shows all file names that are used to refer to external files. The *IDs* are internal identifiers of objects and the *URLs* are references to objects. All editors either apply to the current document or to all documents selected in the file-list.

The editor has three columns. The first column shows the file/ID/URL. The second is the number of appearances and the third is the number of documents where they appear. The table may be sorted by each column.

The value in the first column can be edited inline. In addition, the *File...*, *File/ID editor*, and *URI editor* can be accessed from the context menu.

6.2. Asset Editor

The *asset editor* simply shows the values of the header, i.e. file_version and asset_info. All values can be edited. A click on the *file* button sets the *id* field to the actual file name (if it exists). A click on the *now* button sets the *modified* field to the actual date and time.

6.3. Value Editors

The *value editors* allow to edit particular objects more conveniently by showing several objects and their values in a table. The first column denotes the object by its path in the document. The other columns show the key in the column header and the values for each object.

Note: The initial width of the value columns is the width of the first value column when the editor was closed the last time.

The values can be edited in the same way as in the tree structure. There is an inline editor and the usual type specific editors from the context menu. In addition, an existing value may be removed from the object or a missing value may be inserted into the object.

With the *edit* button, you open an editor to change values is all selected lines at once. Empty fields and the grayed check mark will not do any changes. Also, only existing values are modified, no new values are inserted. The > button provides additional options for strings and opens a color selection dialog for vectors. If the same key is used with different types of values, there is an input field for each type.

6.4. Animations Editor

The *animations editor* is similar to a *value editor*, but shows the values for single frames in an animation. The first column is the URL that refers to the animated value. The second column shows the type for that value. The other columns show the values for each frame in the animation.

Note: The initial width of the frame columns is the width of the first frame column when the editor was closed the last time.

As for the value editors, the values for each frame can be edited, inserted, and removed. The *edit* button opens an editor to change values for the frames in all selected lines at once. The type for the value in all selected lines must be the same. Only existing frames are changed, no values for frames are inserted.

The *insert* button inserts one or several frames, the *remove* button deletes one or several frames. With the *move* button, a frame may be moved to another position in the animation.

6.5. Formulas Editor

The *formulas editor* lists all or selected formulas for a morph or controller in the modifier library. It only supports standard formulas consisting of an output parameter, an optional stage, and an input parameter, which optionally may be multiplied by a constant value.

Output, *stage*, *input*, and *value* are the columns in this editor. The stage value can be set to one of the valid values from the context menu. The values of the other columns can be edited inline. The optional value may be removed or created from the context menu.

6.6. OBJ Import and Export (Geometry, UV, Morph)

There is also some basic support to import and export geometry data in OBJ format. This feature is available only for elements of the geometry_library (import or export of geometry), the uv_set_library (import of UV-mapping), and the modifier_library (creation of a morph). In each case, either the existing data is replaced or new data is inserted. Also, this only affects the data itself, not the other elements of the object representing the geometry, UV, or morph.

Note: The scaling factor for OBJ geometries can be set in the options.

Note: Because the UV-data is stored separately, OBJ import and export does not include the UV-mapping.

6.7. Morph Calculation

Existing morphs can be recalculated with this tool. It is possible to add, subtract, multiply, and divide a constant value or another morph. Subtract and divide can be done in both directions. To use another morph for calculation, select the DSF file with the morph with the File button. This morph is multiplied with the constant value in the field besides the operator selection. Leave the file field empty to use the constant value alone. With the X, Y, and Z buttons, you restrict the calculation to the according directions.

7. Configuration

7.1. Options

Note: Options marked by a star * become effective only after restarting the application.

Toolbar Options

You can select between three toolbar icon sizes: small (20 pixel), medium (30 pixel), and large (40 pixel). Except for small, you may also include text on the toolbar buttons.

Numeric Options

Thumb size is the size of the thumbnails displayed in the file-list.

Most recent used limits the number of remembered files that were recently opened or saved.

Undo steps is the maximum number of actions remembered for undo and redo.

Vector length determines if an array of numbers is represented as a vector or an array. This only affects number arrays when loading files or pasting content, but not opened files or manual input.

Search tolerance is the tolerance that is applied when searching for numbers.

OBJ scale is the scaling factor used for OBJ files. 1.0 will use DAZ Studio scaling, 0.0041 is Poser size.

Generic Options

If *keep backup when overwriting files* is checked, one backup is stored before saving to an existing file. Previously existing backups are deleted.

Activate *ask before closing modified files* to get a notification if you are about to close a file that has been modified but not yet saved.

By enabling *adjust asset id when saving file*, the *id* in *asset_info* is adjusted when saving a file with a new path or name.

With *adjust modified date when saving file*, the *modified* date in *asset_info* is adjusted when a file is saved.

If *compress new files* is active, newly created files are marked to be saved with file compression.

Editor Options

Enable *translate %XX to unicode* to display percentage-encoded strings (see section 2.1) decoded in the tree representation. Please note that this only affects the display in the tree, but not search or any editors.

With *show identifier for objects* active, the tree representation shows in the value column for objects the *id*, *url*, *type*, *channel*, or *output* value, if it has such a key. This is useful to know about the content of an object even if it is collapsed in the tree.

Use *show numbers for arrays* to display an enumeration for each array element in the key column.

Use *start numbering with 1* to start with 1 instead of 0 for enumerating array elements. While 1 may be more common to most people, 0 is more common in computer data. Number-based array indices in DSON (e.g. in geometries) usually start with 0.

If *copy/paste in UTF-8 encoding* is checked, all clipboard activities assume strings to be UTF-8 encoded like it is in DSON files instead of using the OS dependent Unicode format. This is useful if you are using the clipboard also with editors that only use 8-bit strings.

7.2. Content Folders

The folders dialog lists all content directories for DAZ Studio. The first column shows the folder, the second which application versions will use this folder. You can select, for which application versions the content directories should be searched when resolving file references.

With the *Mark* button, you mark all directories that are active for the currently selected application versions. These directories are also marked when you open the folders dialog.

The *Refresh* button reloads the content directory configuration for all application versions. Please note that this change is in effect even if you cancel the folders dialog thereafter.

Version History

Version 1.0, 07.11.2013

Initial release

Version 1.1, 10.03.2014

Bugfixes:

- large numbers (+/-4.2e+9) were displayed/stored wrong
- fixed problem when inserting an item and then deleting the next item
- create morph: vertex_count/deltas lines were missing
- copy/paste didn't work for several array elements at once
- save all didn't reset icons to unchanged
- replace string didn't work if replacing beginning of string with with empty string
- now accepts closing bracket in vectors without space before it
- added "current_value" to materials for value editor

Changes:

- formula editor
- value editor: edit dialog to modify several objects/values
- animations editor: edit dialog to modify several values/frames
- calculation for morphs
- added HD morphs to file references
- added "output" to list ids
- added "delete" as search mode
- added option "Reset Selection" to search menu
- added option "Apply to parent" to search menu
- paste from clipboard: replace selected item

Version 1.11, 22.02.2015

Bugfixes:

- files didn't load if there is no space between a number and the following closing bracket
- "reset selection" and "apply to parent" were missing in search menu for some search types