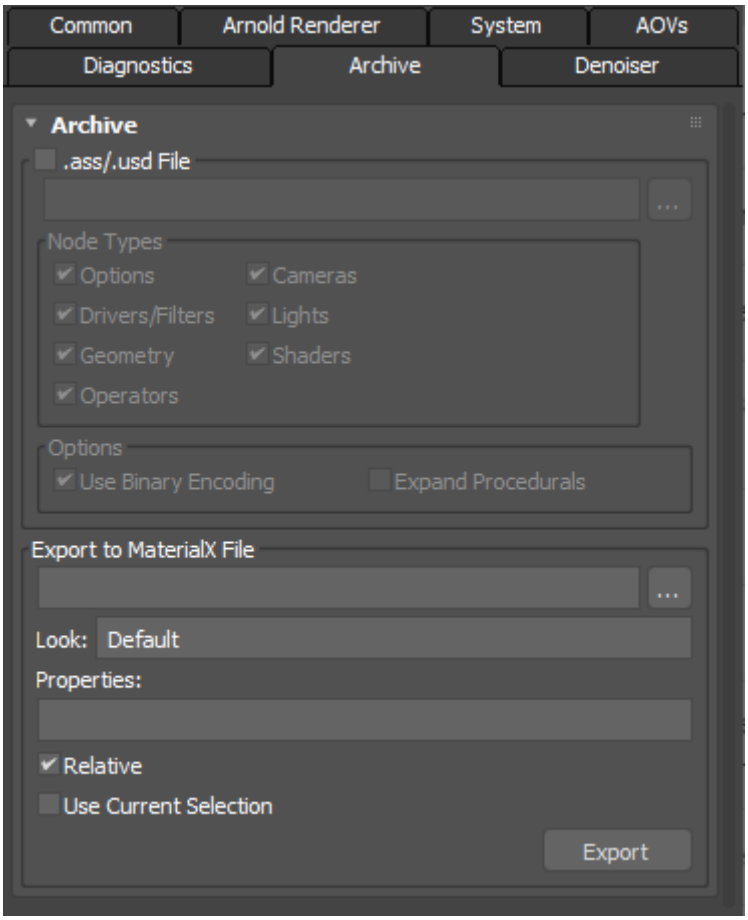
Archive



Archive tab with options for creating an Arnold Scene Source file (instead of a rendering)

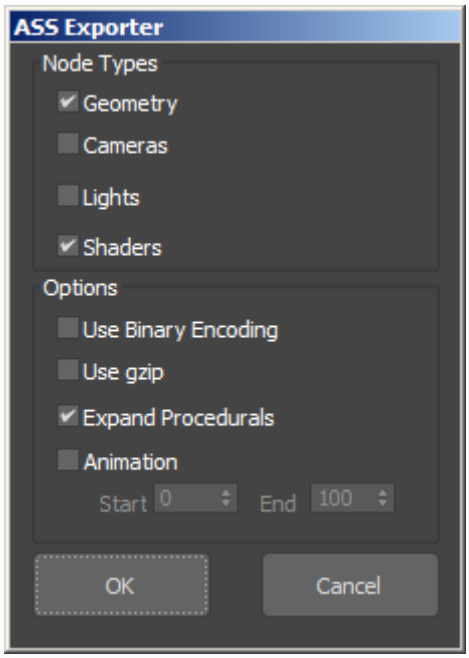
Arnold Scene Source (.ass) is the native scene definition format, stored in human-readable, editable ASCII files.

Exporting To .ASS/.USD

MAXtoA lets you export scenes to *.ass* and *.usd* (*.usda* and *.usdc*) files that can later be rendered outside of 3ds Max with *kick* (a multi-platform, command-line utility) or used for rendering via *procedurals*. It can be found under the *Archive* tab in the Render Setup**.** The window will show some Arnold related options.

> ⚠ If the options are grayed out, it is because you are in ActiveShade. If you change to *Render Production*, they should be visible. Alternatively, you can export a .ass file from the *File> Export* menu.

You can also Export an Arnold Scene Source file through the normal export options - simply choose the option and define your file name. You'll then get a dialog box with export options:



.ass/.usd File

You can export your model as a procedural via this option. The archive is saved as a (*.ass*/*.usd*) file. MAXtoA lets you save scenes to *.ass*/*.usd* files that can later be rendered outside of 3ds Max with *kick* (a multi-platform, command-line utility) or used for rendering via *procedurals*.

Nodes Types

You can also specify which of the following types of nodes are included in the .ass export:

- Option
- Cameras
- Drivers/ Filters
- Lights
- Geometry
- Shaders
- Operators

Options

Use Binary Encoding

Use Binary Encoding does not affect the *USD* format. It depends on the extension (*usd/usda/usdc*). *.usda* is ASCII, therefore the binary flag does not apply.

Specify whether binary encoding is used to compress large arrays (bigger than 16) containing float in their components. They are encoded into a more compact ASCII representation (b85), leading to smaller files and faster load times, while still being mostly human-readable. Also, the binary encoding has exact 32-bit precision, whereas without this binary output floating-point values are truncated into at most 8 ASCII digits (e.g. 1234.5678). The encoded arrays are indicated by prefixing the array type with "b85" as in the example below. POINT2, POINT, and VECTOR arrays are encoded. 16 float

```
polymesh

{

name mymesh

nsides 54 1 UINT  3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4

 4 4 4 4 4 3 3 3 3 3 3 3 3

vlist 47 1 b85POINT

 LJ4iv4THCEE/NV4/1nPCvhpuBkEMk10/LJ4iv2uf+kDVzHI/36OGvhkc/kD9+Fs
/LGenvtmU8EDqJVo/l1cGv+Ij50D1 ...
```

Expand Procedurals

A *procedural* generates Arnold nodes. This option lets you save the generated nodes in the .ass instead of just saving the procedural and its parameters.

Animation

Exports a sequence of .ass files based on the number of frames set in Start/End.

Start

Specifies the start frame of the sequence to be exported.

End

Specifies the end frame of the sequence to be exported.

Export to MaterialX File

Export MaterialX looks for use with Arnold.

You can export shader trees and shape properties from Arnold to MaterialX looks. The export includes:

- Surface shading, displacement and volumes.
- Properties (shape parameters) such as 'visibility', sideness, matte, and displacement settings are automatically exported, and you can specify other shape parameters.
- There is a special visibility assignment which is like the bit mask equivalent, where users can specify they want to disable camera rays.

Exported looks can be reused or shared by Arnold in different DCCs. For each mtlx document you can define what parts of the scene are exported and what the *look* should be called. The Arnold plug-in translates the scene and then calls the public API function which goes through the universe, generates a mtlx document and then writes it to a file.

Supported:

- Export material assignments with node graphs with Arnold's proprietary closure type, channel information and context definitions (shader and displacement).
- Material/property/visibility assignments.
- Replace/append mtlx document mode.

Unsupported:

- Look variants because this is a baked version of the scene.
- Support for the MaterialX standard library.
- Shader assignments for lights.
- Other renderers and MaterialX viewers.

> ⚠ You cannot render whilst exporting to MaterialX.

Export to MaterialX Path

Choose the name and path that you want the .mtlx file to be saved as.

For existing mtlx files, the export will either update an existing look, or append a new look to the document.

Look

The look name to use in the exported MaterialX document.

Relative

If set, the shader/property/visibility assignments will be relative to the namespace the shape is in, otherwise, the assignments will contain the full path to the shape. For instance, if a sphere named '`sphere1`' resides in a procedural '`my_proc`', the full name would be '`my_proc/sphere1`' and the relative name '`sphere1`'. This makes it possible to assign the look to another procedural with a different name.

Use Current Selection

Uses the objects currently selected.

Properties

Additional shape parameters to export. For example, step_size and volume_padding for a polymesh that is rendered as a volume.

The following shape parameters are automatically exported (if they are set to non-default values):

- visibility
- sidedness
- disp_padding
- disp_height
- disp_zero_value
- disp_autobump
- autobump_visibility