FBX Asset Metadata Pipeline

Describes how to get custom user-defined attributes into Unreal through FBX, and how to work with them in the Unreal Editor using Blueprints and Python.

As the size and complexity of real-time 3D productions increases, and as the number of tools that make up a modern production pipeline continues to grow, adding intelligent automation to increase artist efficiency becomes ever more critical. This intelligent automation typically relies heavily on metadata: custom-defined properties about your Assets that give them meaning within your project.

An artist or creator adds metadata properties in one application, typically where the Asset is first created. The metadata then stays associated with that Asset, and applications farther down the pipeline can read that metadata and use it to make decisions about how to process or handle that Asset.

In the Unreal Editor, you can set and retrieve metadata keys and values for your Assets using Blueprints or Python scripting, as explained on the Asset Metadata page. This can help you with any Asset management operations you carry out solely within the context of your Unreal Project.

However, the real value of having metadata on Assets typically lies in interoperability with external applications, and in the ability to create larger Asset management pipelines that span multiple applications. To this end, the Unreal FBX Importer brings in certain kinds of Asset metadata from FBX files and makes that metadata available inside the Unreal Editor.

> Because this metadata is assigned to the Assets in your Project, you can't access it directly in your runtime gameplay code. It is intended primarily for use in the Unreal Editor, for scripting Asset management operations.

Metadata Sources

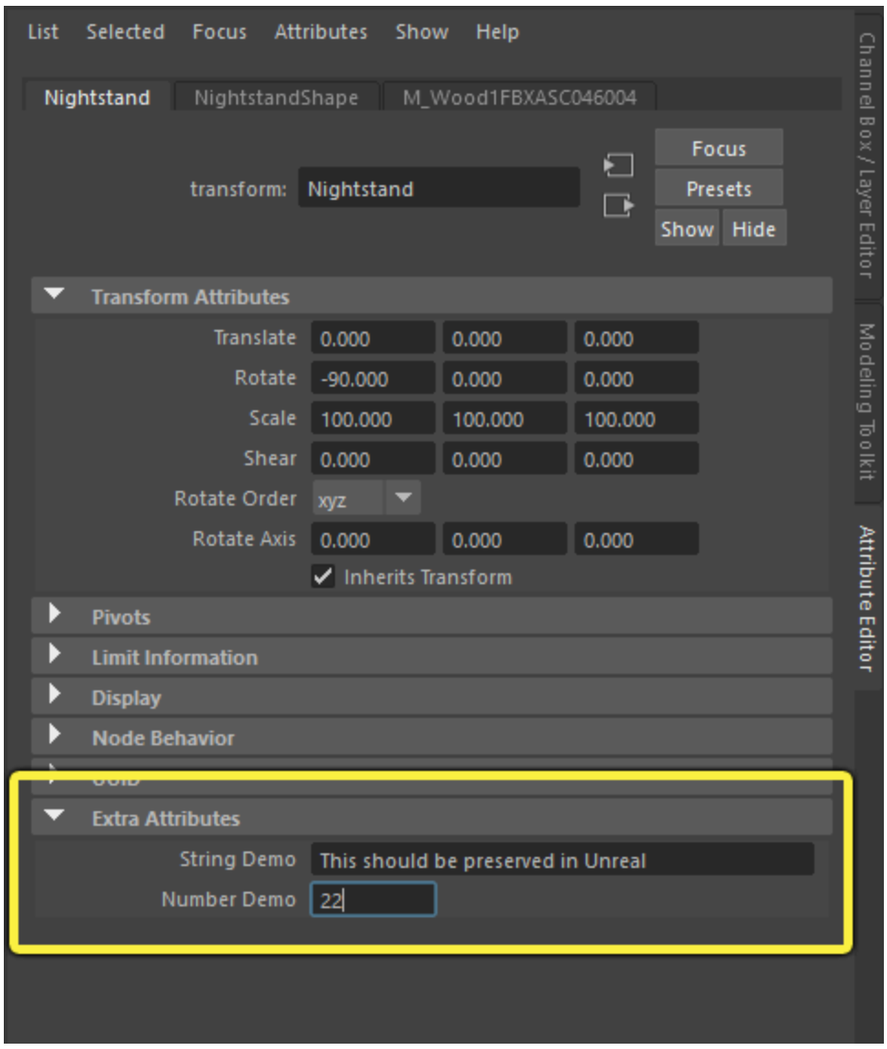
When you import an FBX file into Unreal, the importer reads any `FbxProperty` values attached to the objects in that file and attaches them to the Unreal Assets it creates as a dictionary or map of tag-value pairs.

The following applications are known to support exporting custom user-defined metadata to FBX files through this `FbxProperty` system.

Autodesk Maya

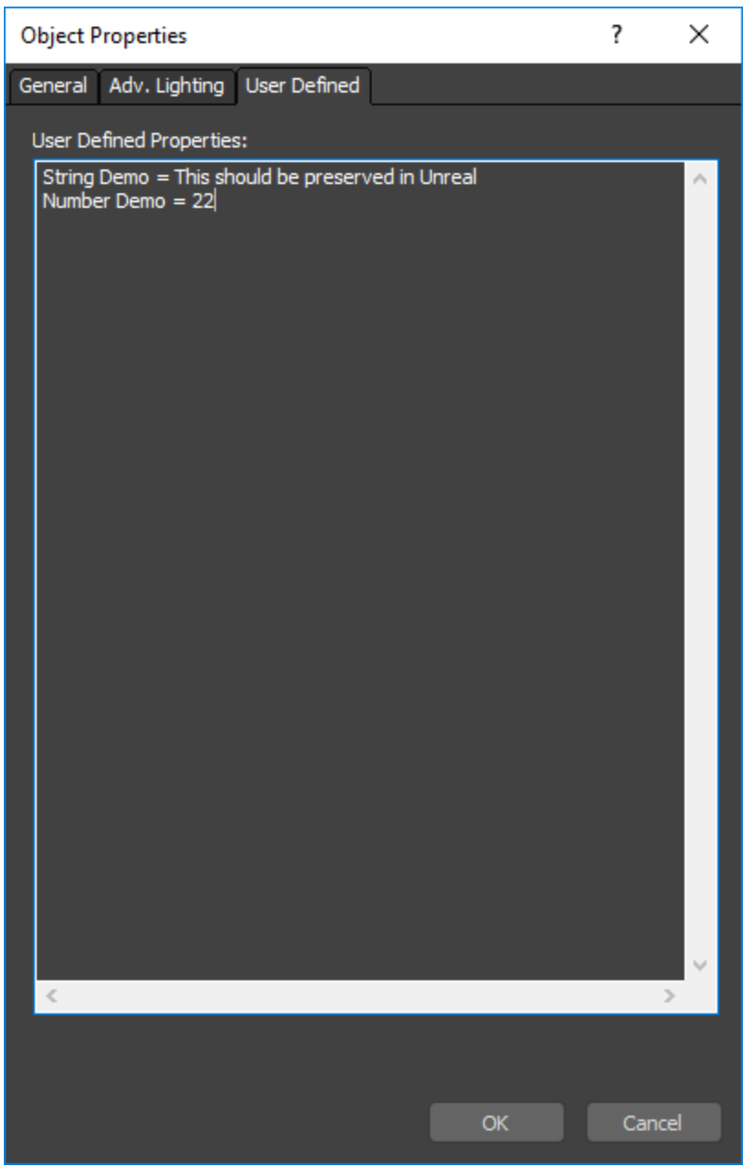
To create metadata for an Asset in Maya, add extra custom attributes to the object. For details on how to do this, see the Maya Help.

Your custom attributes should show up in the Maya **Attribute Editor**, in the object's **Extra Attributes** list. For example, this object has two custom attributes created for it: one string property and one numeric property:

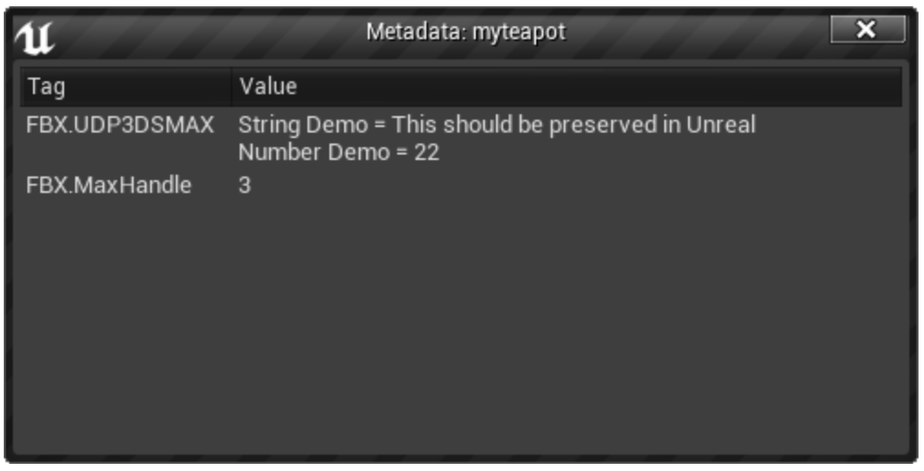


Autodesk 3ds Max

To create metadata for an Asset in 3ds Max, add it to the **User Defined** tab of the **Object Properties** dialog:



Currently metadata from 3ds Max is imported as a single string that contains the complete contents of the **User Defined** text box, which you can access in the `FBX.UDP3DSMAX` metadata tag. You will likely need to split this string value yourself when you read it in your Unreal Editor Python or Blueprint script.



Metadata After Import

There are a couple of things you need to know about how your Asset metadata appears in Unreal.

- All metadata keys and values are stored as strings in Unreal Engine, regardless of their original type when set in an external application. For example, if you set a metadata value in Mayaas a number like `22`, this will be a string that contains the value "22" when you read it back in a script within Unreal. If you need the values to be numbers, use a Blueprint conversion node like **Utilities > String > String to Int** or **String to Float**, or a built-in Python string parsing functions like `int()` or `float()`.
- If you create tag names with spaces in your source application, the spaces will be removed by FBX.
- Typically, tag names have a prefix that indicates the source of the value. For Assets imported through FBX, this prefix is **FBX.**. Other applications or plugins may use other prefixes. For example, the Unreal Shotgun integration uses the prefix **SG.**.
- When you work with Skeletal Meshes, you may be able to assign different metadata values to each separate bone in the skeleton. In this case, the FBX import process still assigns all the metadata values to the Skeletal Mesh Asset, but it includes the bone names in the prefixes of the metadata tag names so that you can distinguish the values that apply to each individual bone. For example, on a Skeletal Mesh you may see tags similar to **FBX.Right_Arm.tag_name** and **FBX.Right_Hand.tag_name**.

So, for example, a tag that you create in Maya with the name **Created By** is available in Unreal as **FBX.CreatedBy**.

For more information on how to use this imported metadata in the Unreal Editor, see Asset Metadata.