ASSET INVENTORY
GUIDE - VERSION 2.7.0

INTRODUCTION

Put your asset workflow on steroids and say good-bye to the Package Manager as you know it! Asset Inventory is your ultimate asset management companion: a lightning-fast search for assets for your current project. Find content in assets you purchased or downloaded without importing and bring single files in with just a click.

Eliminate the time-consuming task of finding a sound file, a texture or a model you know you purchased but which is hidden inside one of your many Asset Store purchases. The Asset Inventory provides a **complete list of all assets you own**, including their content.

CONTENTS

FEATURES

The tool aims to provide **what you always wanted the Asset Store & Package Manager to be**: The best possible Asset Management solution for Unity.

Find assets effectively. Search inside your purchases. Include custom packages you downloaded from Humble Bundle or other places. And packages from a registry. Manage everything through tags. Perform bulk operations. Import multiple assets and packages in one go. Identify used assets inside your projects. Search using asset-specific criteria. And much more.

Powerful Search

Browse & find anything inside your purchased Asset Store packages without importing. Quickly preview audio files. Narrow your search using asset type, tags, image dimensions, audio length, colors and more. Exclude items you don't want to see. **Save and recall** searches.

Easy Setup

Works immediately out of the box. Lots of optional view & configuration options. Hassle-free indexing. Start, stop & resume at any time. Works with Unity 2019.4 and higher. Windows, Mac & Linux. Lightning-fast indexing and search.

Intelligent Updates

Handle updates from the Asset Store and from the Unity registry in one holistic UI. Define update strategies per package. Instantly see if there are any updates you should consider. Full package manager functionality for registry packages including add/update/remove and compatibility information.

Import & Export

Import only what you need instead of a whole package. Automatically determines asset dependencies to import complex prefabs and materials. Save space & reduce clutter in your project. **Bulk import** multiple packages at once. Automatically store imported assets in a specific sub-folder and keep the Assets root clean. **Export** assets easily for reuse in other contexts.

Many Sources

Your complete asset library: Automatically indexes Asset Store purchases. Triggers download of missing assets. Handles packages from registries. Integrate the Unity Cloud Asset Manager. Add custom folders to search through Unity packages downloaded from other locations. Indexes folders and zip/rar archives containing **arbitrary media files** like 3D models, audio libraries, textures and more. Automatically generates previews.

Organize

Automatically imports labels from the Asset Store. Use additional **tags** to group assets effectively. Assign tags to either packages or individual files inside packages. Assign colors and group by tags. Exclude unwanted items. Browse all sub-packages. Perform bulk operations. Import multiple assets and packages in one go. Builds on Package2Folder to allow importing packages into a custom sub-folder. **Backup** packages automatically to ensure you always have a working version to go back to.

Reverse Lookup

Quickly identify used assets and packages in your project. Ensure license compliance.

Constant Updates & Support

Receive regular updates, optimizations, and new features. Super-fast support. Discuss ideas in a great Discord community.

GETTING STARTED

INSTALLATION

It is strongly recommended to remove any previous version of the Asset Inventory before installing a new version. The best way to do so is to close Unity and afterwards delete the full Asset Inventory folder since used libraries cannot be replaced if Unity is active.

Install the package through the Unity Package Manager. When asked if to install additional dependencies, select **Install/Upgrade**, otherwise the asset will not work.
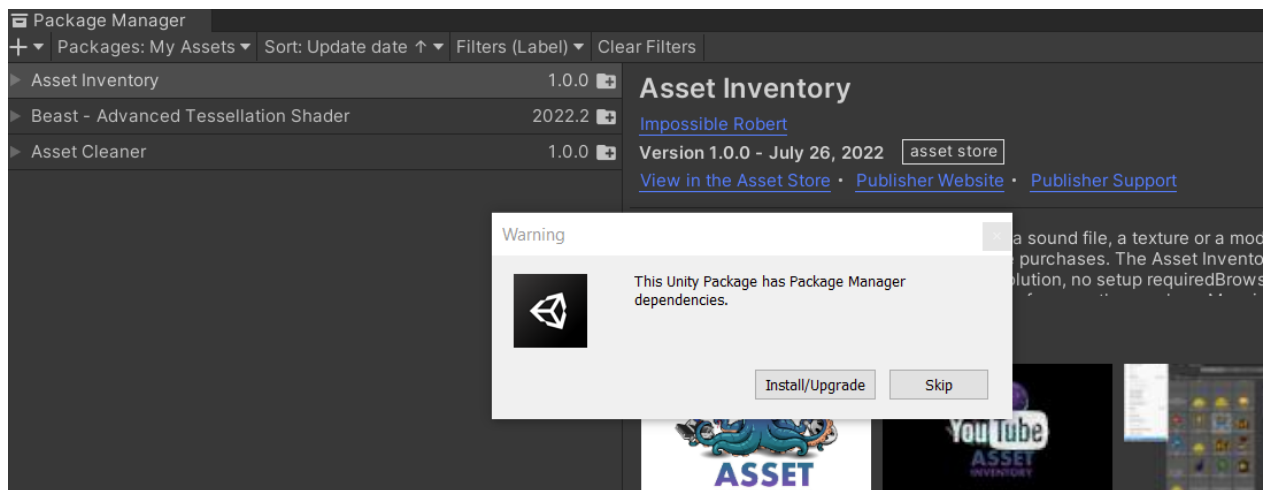
When indexing packages Unity might try to call third party applications like Blender, 3ds Max or Maya. It is recommendable to at least **install Blender** (free) to make sure previews for blend files are calculated correctly.

USAGE

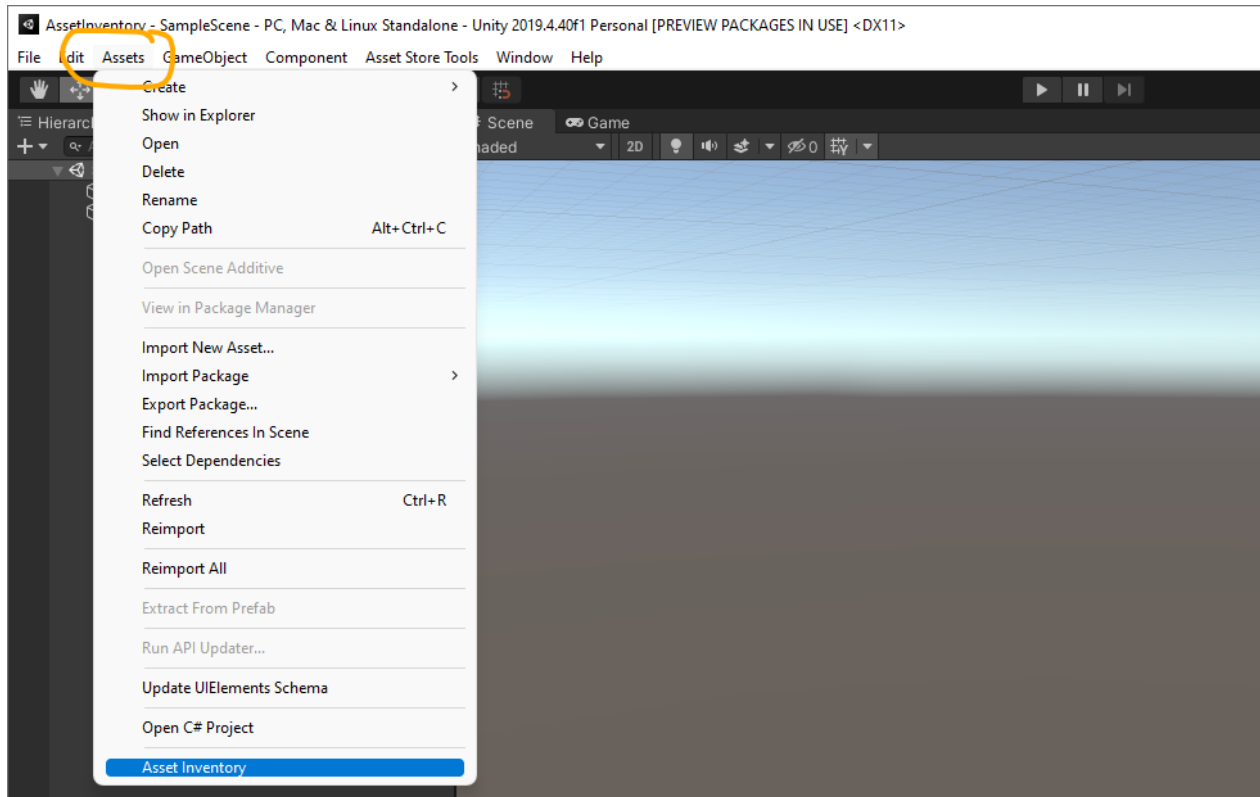Open the Asset Inventory through the **Assets menu**.

There is **no manual setup** needed. The only requirement is a fair amount of disk space for storing preview images and temporary files during extraction and browsing.

Press **Start Indexing** and wait for results to come in. When indexing is done it can be beneficial to **delete your project Library folder** which has accumulated lots of temporary unneeded data.

Whenever you purchase new packages or add new assets to your folders, make sure to press the "Update" button under **Settings** once so that the index gets updated.

Tip 1: Activate the **Download Assets for Indexing** option under **Settings** to index your complete library without the need to download assets manually first.

Tip 2: The UI displays only the core functionality (easy mode). You can access many **additional options** by holding down the **CTRL key** or pressing the eye icon at the top.

PERFORMANCE

Here are things to consider to get the most out of the tool:

- Run the initial indexing or bigger updates in a **new empty Unity project** and use the newest available stable Unity release. The indexing results will be stored in a central location for reuse in any other project. This will give you the highest indexing performance since no other assets/scripts need to be refreshed and new versions of Unity typically bring big performance improvements for object and sound importers (in some cases 10x).
- Index in a **BIRP** project, as otherwise additional URP/HDRP dependencies need to be resolved or converted which takes additional time and can lead to pink previews.
- Delete the **library folder** every now and then as it can become very big and slow down Asset Database operations.
- **Optimize the database** after big operations (Settings/Maintenance menu).
- **Deactivate sorting** of search results for faster paging.
- Deactivate **FBX dependency scanning**. In 95% of cases *fbx* files do not have dependencies listed inside.

ASSET SEARCH

OVERVIEW

Once an asset is indexed, it can be found in the search. Searching will automatically start when typing. By default, every word entered needs to match the search result but not necessarily in the same order. If the exact phrase should be matched, prefix the search with ~.

Examples:

- *"car interior"* will return results like *"CarBlueInterior.fbx"* and *"InteriorCarDes.png"*
- *"~car interior"* will not return the above but only results like *"Car Interior.fbx"*
- *"car +interior -fbx"* will return results that match *"car"* and *"interior"* but not *"fbx"*

There is also an **expert search** available (search starting with "=") which is described later.

Additional filters can be selected to narrow down the results further. Once an asset is selected, details are shown on the right-hand side about the file and the package the file is contained in.
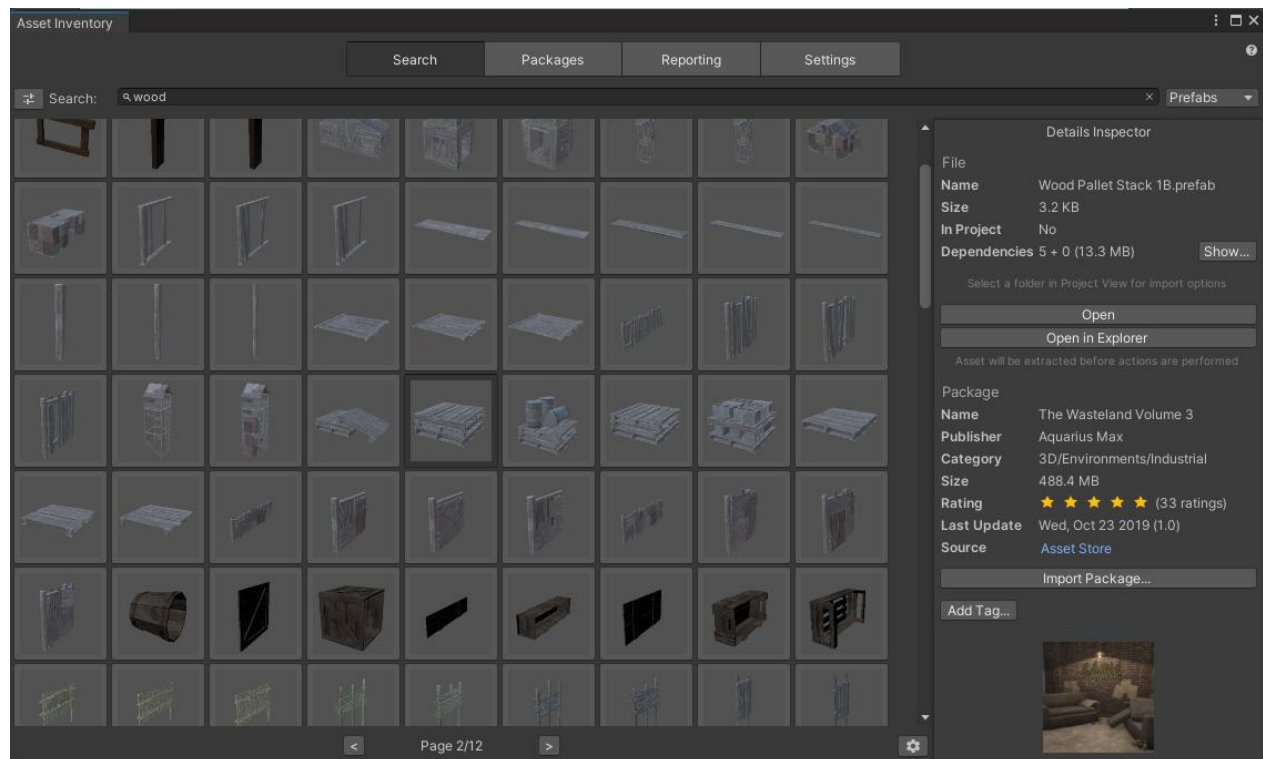
Selecting an audio file will **automatically play** it. This way it is easy to quickly preview audio files. Initial playback might take a while until the corresponding package is temporarily extracted.

Multi-select is possible by holding down the *Shift* or *Ctrl* key. Drag and drop is possible from Unity versions 2021.2+.

ADVANCED FILTERS & SETTINGS

Clicking the **Filter icon** in front of the search will bring up the **filter bar**. It contains additional filters and media-specific properties to filter for specific image dimensions or audio length. Using the buttons in-front of each value toggles if results match that are bigger or smaller than entered. The filter dropdowns will only show values if respective assets are available. If the search type is limited (e.g. to Images), some filters might not show up (e.g. Length) as they are only applicable to other types.
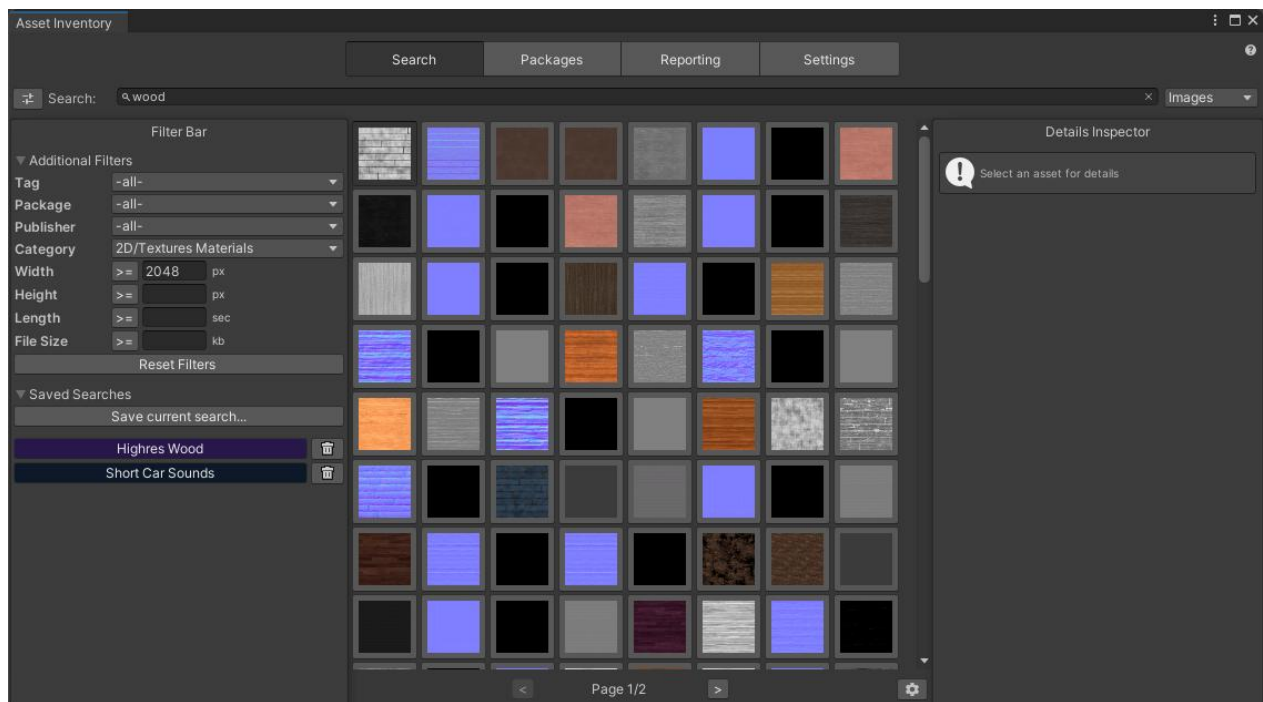
The filter bar also contains the section for **Saved Searches**. This allows to persist the current search filters and later recall these easily. The results are not persisted but instead live from the database.

Using the **settings icon** in the lower right corner will open additional view settings. Specify the tile size, which text to display on the tiles and if assets should be pinged automatically upon selection.

| View Settings	
Search In	Asset Path ▼
Sort by	Asset Path ▼ ^
Results	250 ▼
Hide Extensions	✓ asset;json;txt;cs;md;us
Tile Size	──●────── 98
Tile Text	File Name ▼
Search While Typing	✓
Sub-Packages	☐
Auto-Play Audio	✓
Ping Selected	✓
Ping Imported	✓
Group Lists	✓
Double-Click Action	Import ▼
Dependency Calc	Upon Selection ▼
Previews	Only With Preview ▼

IMPORTING ASSETS

Selecting a folder in the Unity **Project View** will bring up **import** options when selecting an asset in the search. Assets can be imported individually or with all detected dependencies. The latter will automatically create a sub-folder with the asset name and store all files in there.
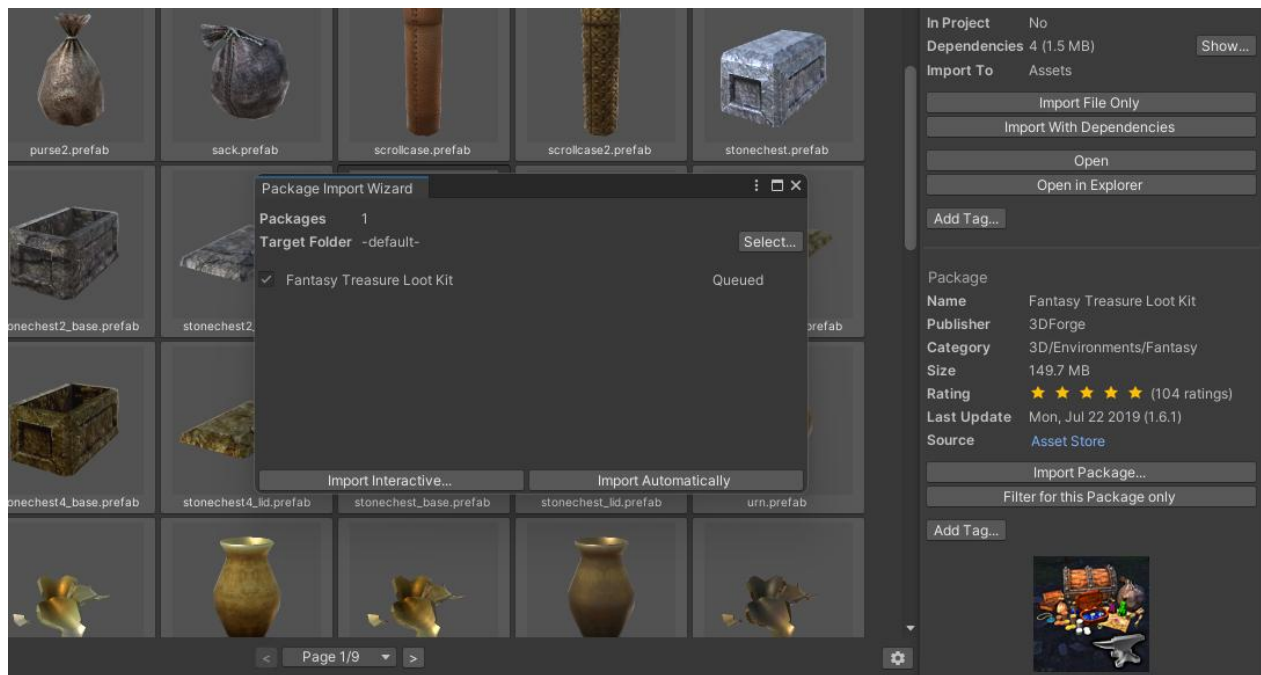
In case there are **script dependencies** these can also be imported. Due to unforeseen dependencies in scripts, that will typically only work for scripts that are self-contained, like custom shader editors or simple logic classes. Otherwise, there might be compilation errors.

Clicking the *Show* button behind the dependency information will bring up the dependency information details, listing all files, their size and if they are already in the project or not.
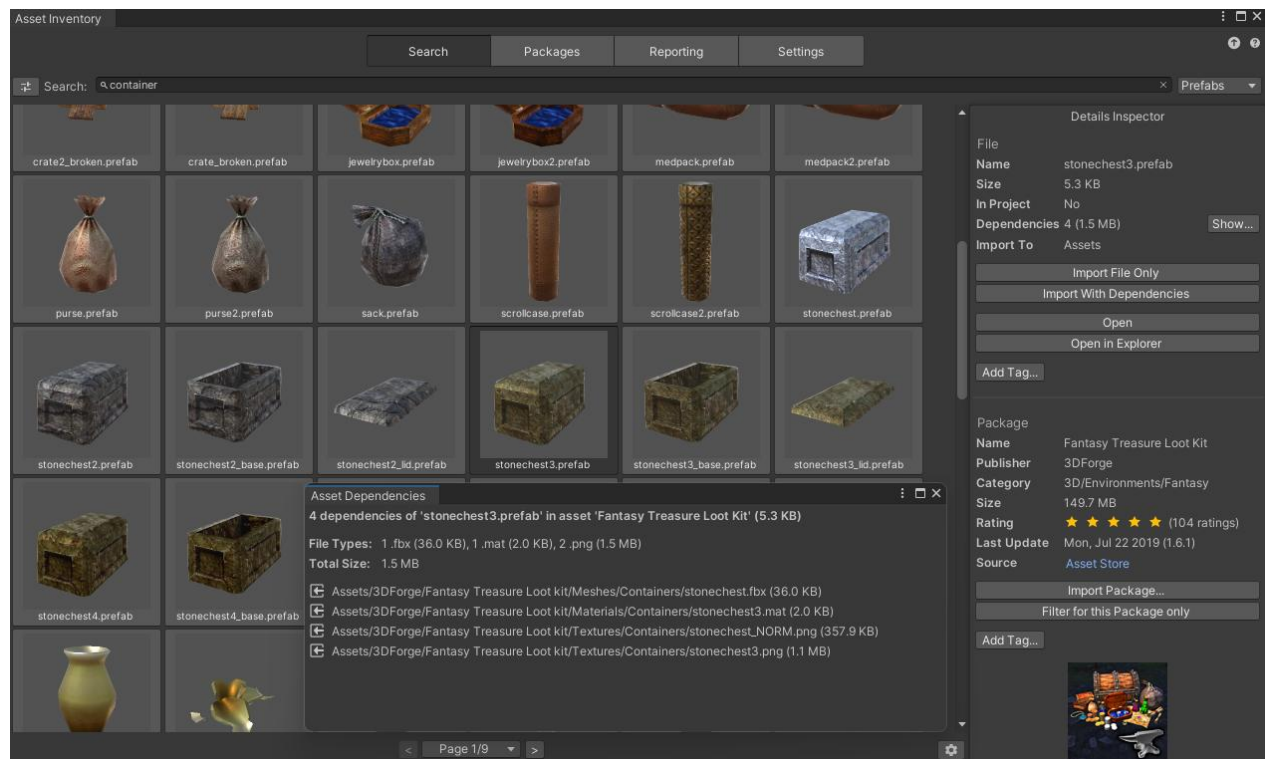
Import can also be triggered by **double-clicking** on an item (if activated in Search settings) or by **dragging** it into the Project Window. Once an item is imported into the project it can also be dragged from the search into any other window, e.g. the scene or fields accepting the dragged object type.
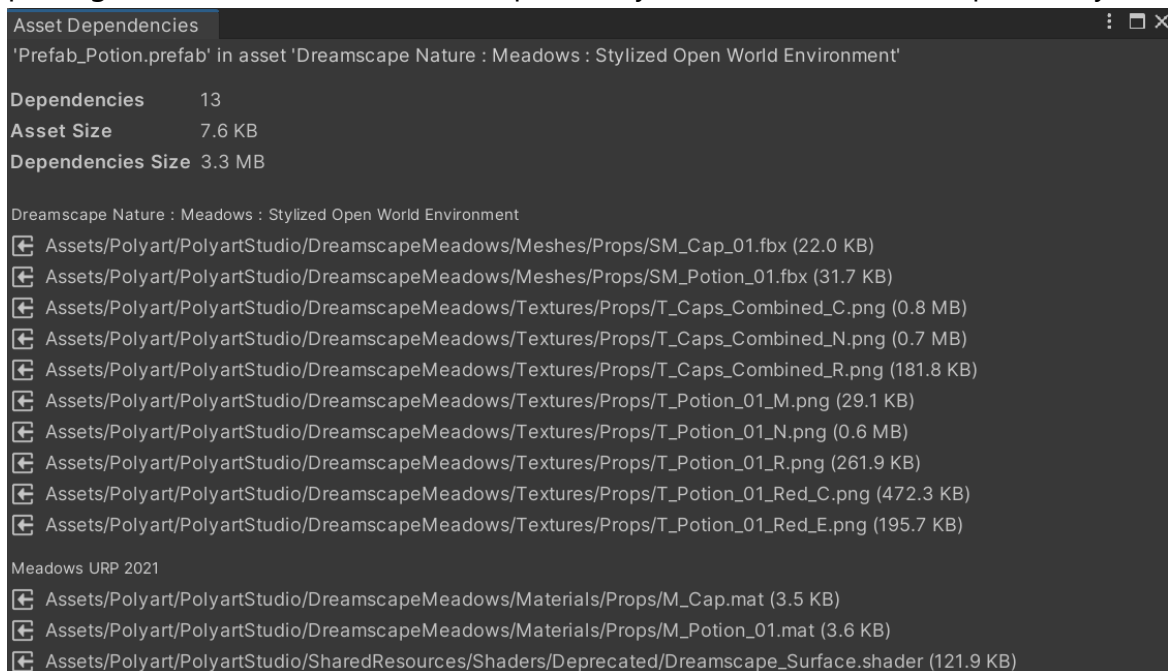
Scriptable Render Pipeline (SRP) Support

When working in an **SRP** project, materials can appear with the pink error shader due to incompatibility. The tool provides extensive support for SRP projects to fix this. Two modes are available which can be combined for maximum coverage:

1. Many packages bring dedicated **SRP support packages** with them which contain additional files that need to be installed when working in a URP or HDRP project. These support packages will **automatically** be detected and when importing an asset, the tool will check which additional files are required from the support

package. You will see the full dependency resolution in the dependency UI.

2. Packages without special SRP support packages can in many cases still be used in URP projects since Unity provides the **Render Pipeline Converter** which will adjust the imported materials to fit the URP materials. If activated under *Settings* (available in URP projects with URP 14+), the tool will **automatically trigger this conversion**, making it very convenient to import assets. If a dedicated SRP support package was found, this step will automatically be skipped. Since there is no programmatic way to influence which materials are converted, this setting will convert all project materials (typically not a problem) and will incur a minor speed penalty. It will also work when recreating previews.
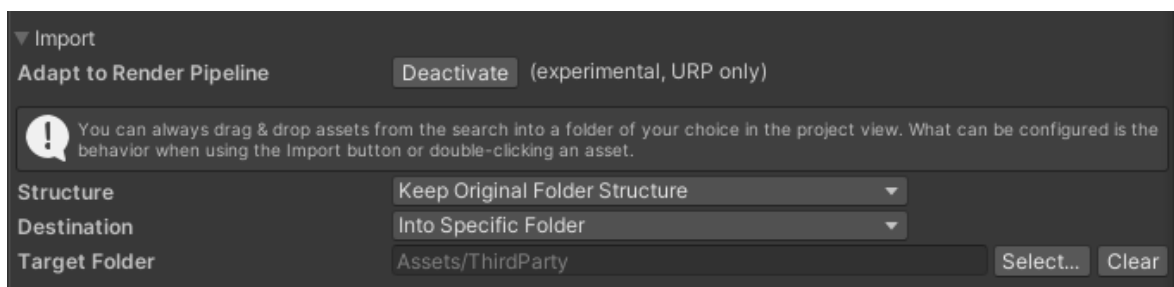
EXPERT SEARCH

It is possible to use nearly the full feature set of SQLite 3 to search. This mode is activated when starting the search with "=". Afterwards the database fields and conditions can be stated.

Examples

=Asset.PackageSize > 0 and AssetFile.Type="wav"

=AssetFile.Width > 3000 and AssetFile.FileName not like "%Normal%" and Asset.DisplayName like "%icon%"
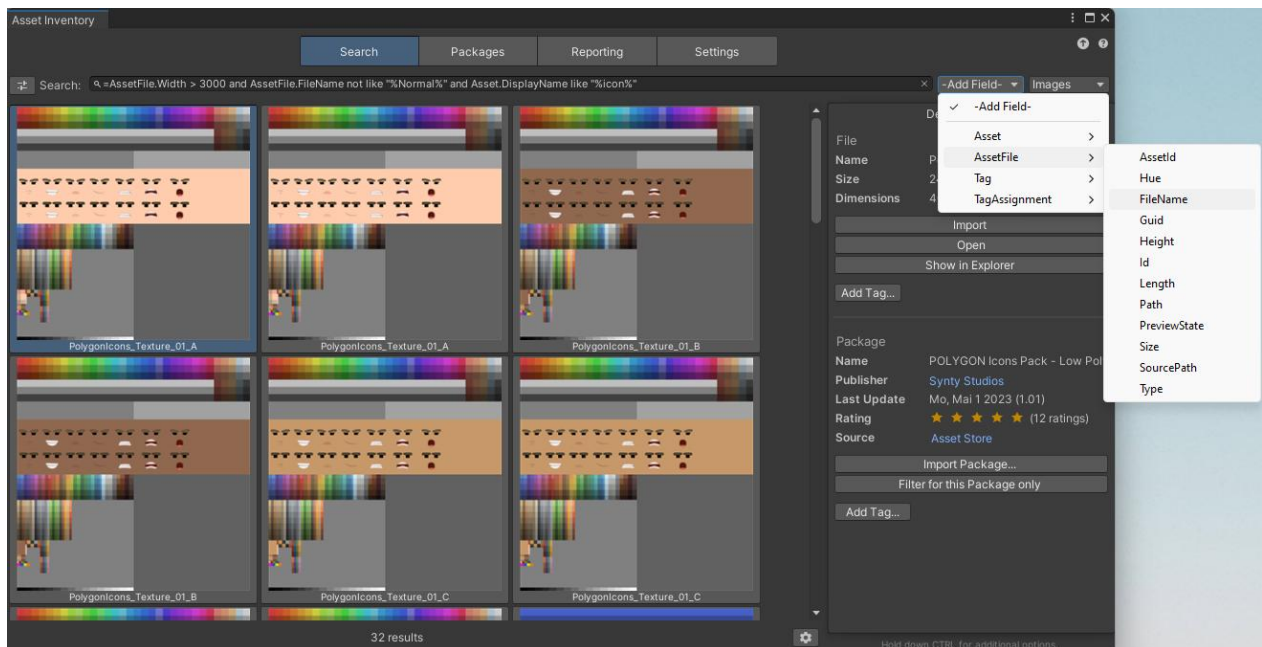
Useable fields can conveniently be picked from the dropdown behind the search field.

PACKAGES

OVERVIEW

The packages overview will show a list of all packages, indexed or detected in the current project. These can come from multiple sources:

Automatic

- Asset Store purchases
- Registry packages
- Unity Asset Manager
- The local Unity asset cache
- The local Unity package cache

Additional folders stated under *Settings*

- Unity packages from other sources (e.g. Synty)
- Local asset packages under development (via package.json)
- Local media libraries (sound files, textures, models…)
- Unity projects
- Archives (zip, rar & 7z contents)

The list will indicate which of these were already indexed. To index more packages, download them into the cache or add more additional folders. Display is possible as a plain list or grouped by categories, publishers, state or tags.

In the case of registry packages, only the latest version of the package will be indexed.

If a package contains **sub-packages** (also recursively), they will be shown directly under the package in a tree structure.

Double-clicking any package will show the contents in the search. After selecting a package, multiple options will appear. It is possible to import it. Alternatively, it can also be removed from the index to trigger a reindexing on the next run. This can be useful for incorrectly indexed files. Packages can also be completely removed from the database which can be useful after cleaning up outdated assets from the local cache.
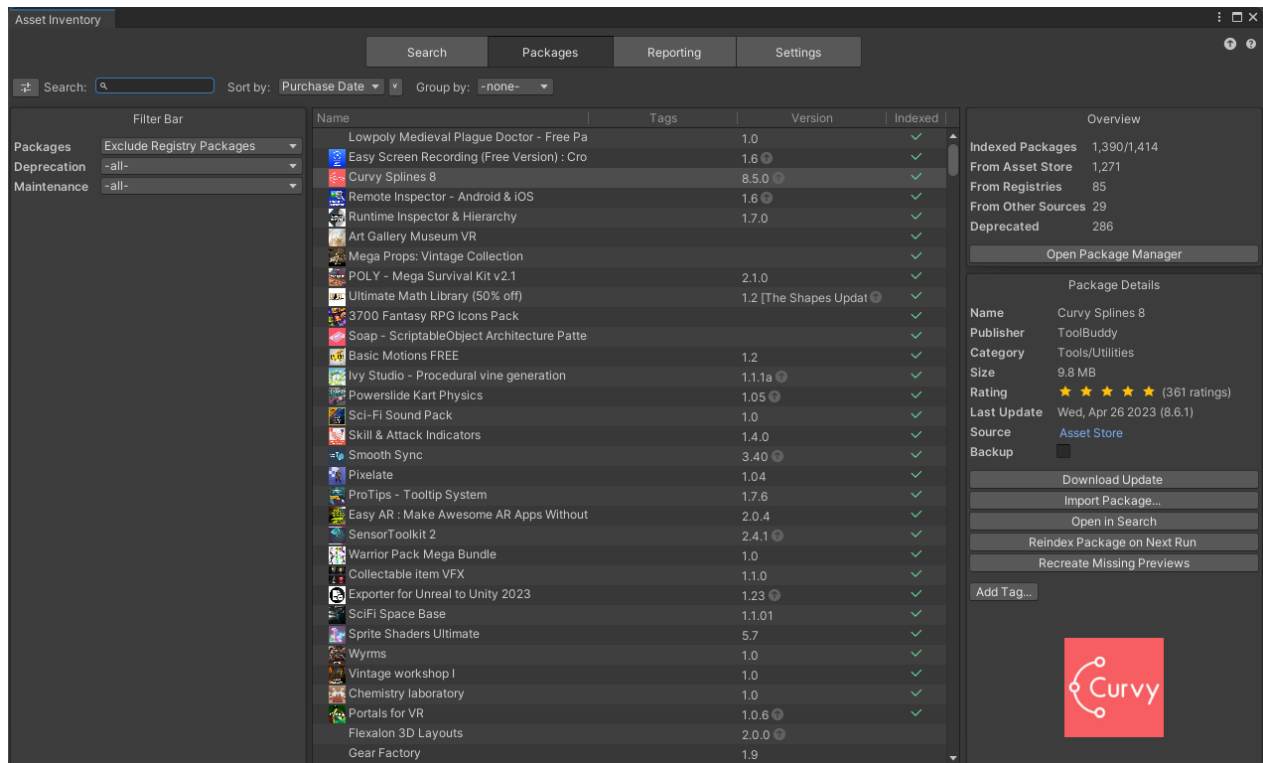
Depending on the package type additional options become visible:

- Registry Packages
  - Support for managing the installed version and setting an **update strategy** (recommended, latest stable compatible, latest compatible, manually).
- Development Packages
  - Use directly via **file link**
- Custom Packages
  - Support for **connecting to metadata** from the Asset Store. This will load the name, description, rating etc. and is especially useful when having downloaded packages from alternative sources like the Humble Bundle to still have all metadata available. See details in Custom Metadata section below.
  - Support for disconnecting from the Asset Store again

Using the buttons under the package list, the display can be switched between list and grid style. Right-click on any column header to add or remove columns.

TAGGING

Tags will be imported from the Asset Store in case any are set there. In addition, local tags can be added and removed here as well (they are not synchronized back to the Asset Store yet). Bulk editing of tags is possible when selecting multiple items in the tree.
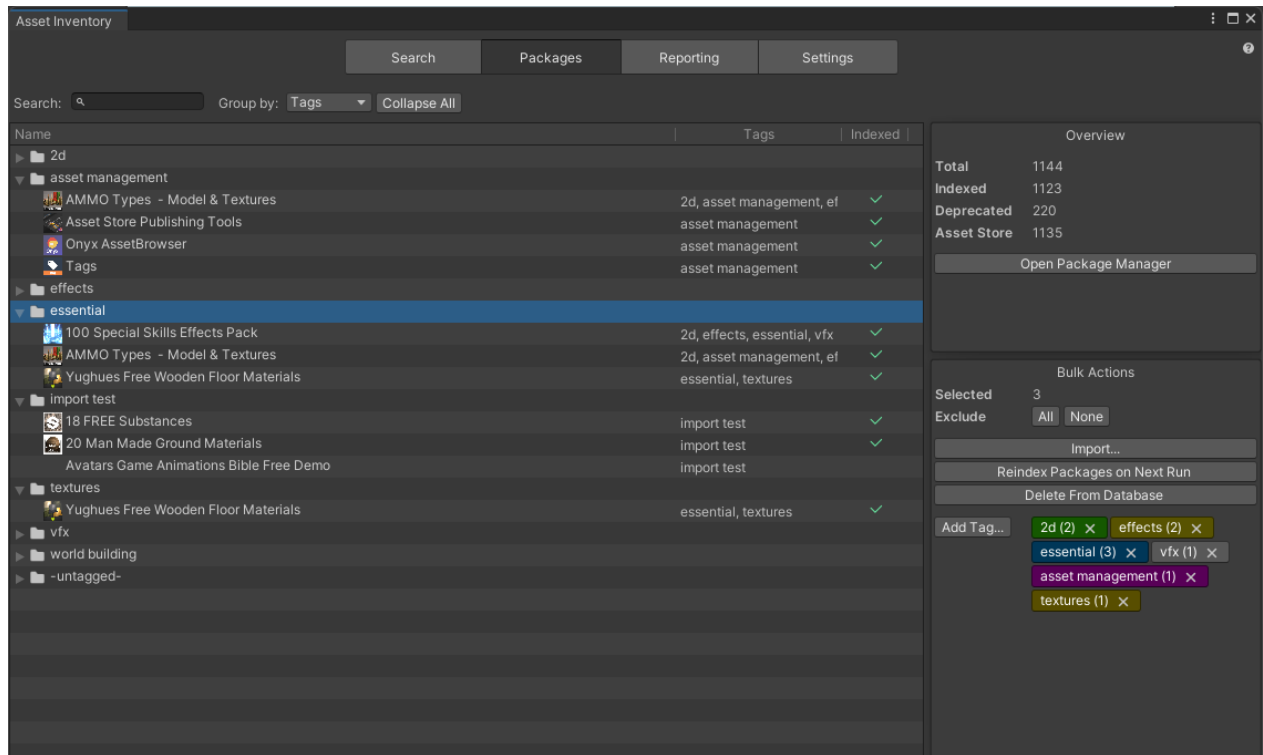
When adding tags, the **Tag Management** window can be opened through the small cog wheel. There, tags can be created, colored, renamed and deleted. The size of the tag selection window can be changed under *Advanced* settings.
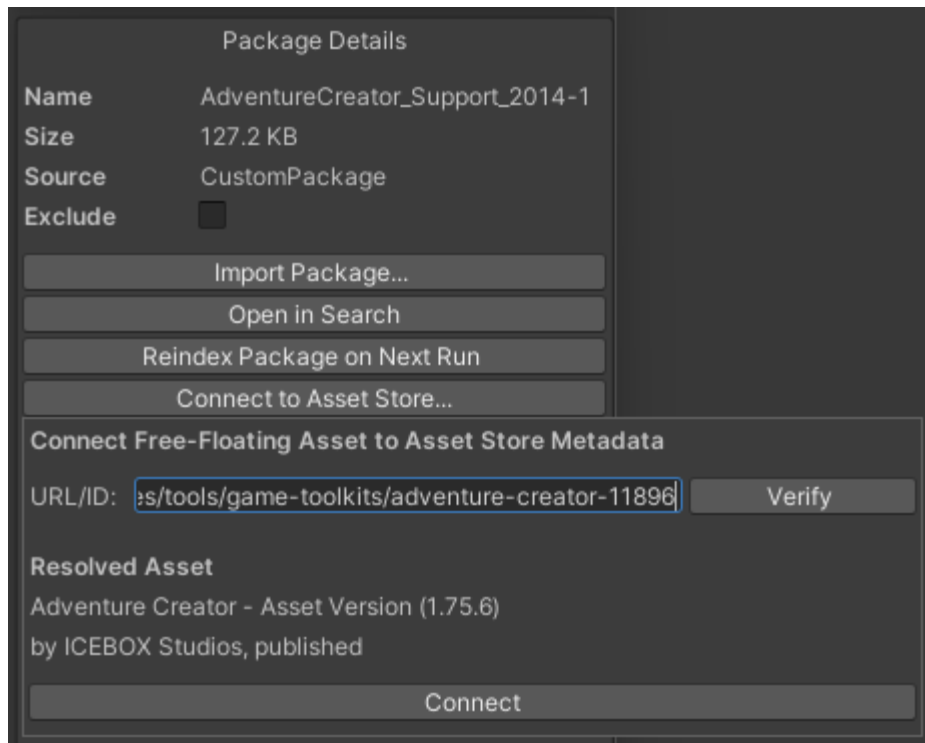
CUSTOM METADATA

Package metadata like name, publisher, category etc. is extracted directly from the package if the information is contained in there. If it is missing there are three options.

Link to Asset Store

If the package exists on the Asset Store, the package can be linked to the entry there and the metadata will be loaded automatically and also regularly to fetch updates to it.

Copy/paste the URL from the Asset Store website of the asset you want to link, *Verify* it is detected correctly and click *Connect*.

Manual Data Entry

Metadata can also be entered manually using the "Edit Data..." command visible for packages that are not linked to the Asset Store. This way also custom preview images can be set for archives and custom packages.
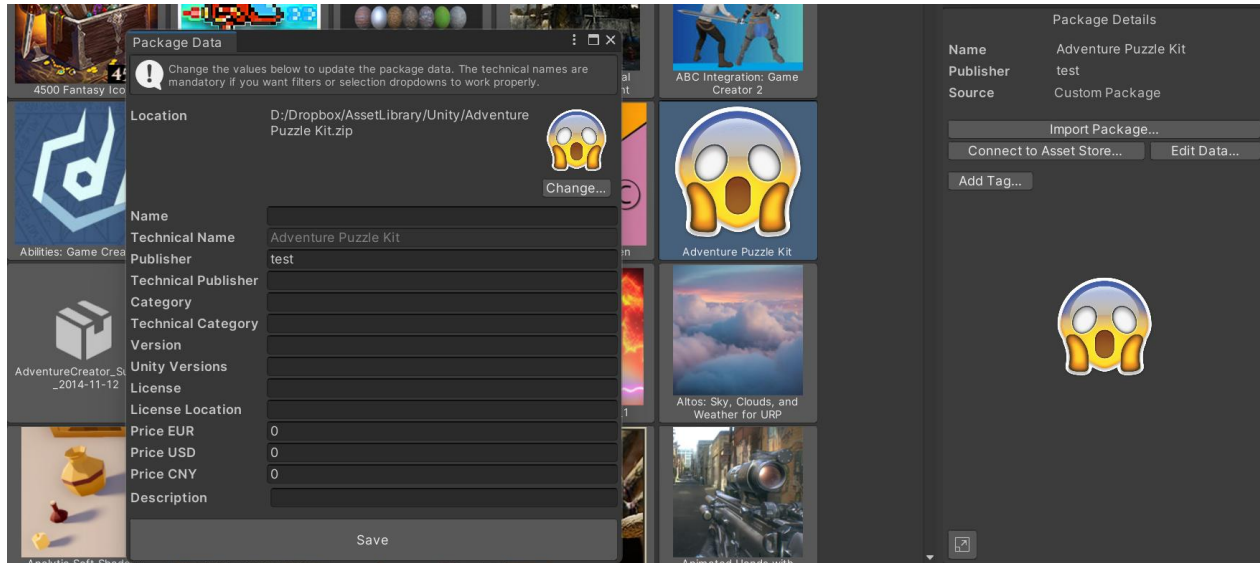
Override File

The last option is to create an Override *Json* file containing metadata. This needs to be named like the package file with the suffix *".overrides.json"*. This overrides file can contain any data a package can hold and also a list of tags in addition. If defined in there, this data will override any existing data read from the Asset Store or the header file.

The advantage of this method is that such files can be put in a central location, so everybody who will index the location has the same metadata. This ensures for example the common use of tags and categories if working in a team.


```
{
    "displayName": "Three Skyboxes",
    "tags": ["sky", "limit"]
}
```
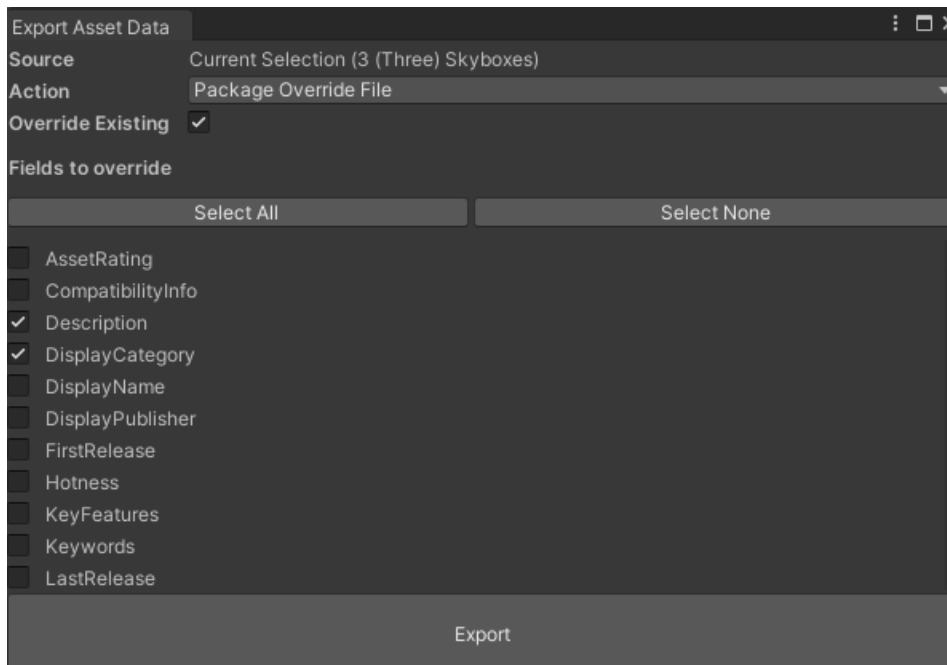
Supported fields: *displayName, displayCategory, safeCategory, displayPublisher, safePublisher, publisherId, slug, revision, description, keyFeatures, compatibilityInfo, supportedUnityVersions, keywords, version, latestVersion, license, licenseLocation, purchaseDate, firstRelease, lastRelease, assetRating, ratingCount, hotness, priceEur, priceUsd, priceCny, requirements, releaseNotes, registry, repository, officialState, tags*

Consider that filtering and grouping works on the *Safe* versions of category and publisher so these should also be set when changing these values. *Safe* versions are named in a way that they could also be used in the file system, without special characters like *&* or */*.

SafeCategory	DisplayCategory
Filter	Filter
Complete ProjectsSystems	Complete Projects/Templates
3D ModelsProps	3D Models/Props
AudioSound FX	Audio/Sound FX
Textures Materials	Textures & Materials
Textures MaterialsGround	Textures & Materials/Ground
3D ModelsVehiclesLand	3D Models/Vehicles/Land
Textures Materials	Textures & Materials
Editor ExtensionsGame Toolkits	Editor Extensions/Game Toolkits
Complete ProjectsPacks	Complete Projects/Packs

To make it easier to create package override files, the Package Export supports to create these conveniently for an individual or also a list of packages.

Export Asset Data				
Source	Current Selection (3 (Three) Skyboxes)			
Action	Package Override File			▼
Override Existing	✔			

Fields to override

Select All	Select None

- ☐ AssetRating
- ☐ CompatibilityInfo
- ☑ Description
- ☑ DisplayCategory
- ☐ DisplayName
- ☐ DisplayPublisher
- ☐ FirstRelease
- ☐ Hotness
- ☐ KeyFeatures
- ☐ Keywords
- ☐ LastRelease

Export

IMPORT & BULK IMPORT

Importing packages can happen one-by-one and in bulk while multiple packages are selected. Both interactive and automatic mode is available. In addition, a **custom root folder** under which assets should be imported can be selected. This is especially helpful if the */Assets* root should be kept clean and organized. Packages from registries will be added to the project manifest. If registry packages need a custom scoped registry this will also be added automatically.
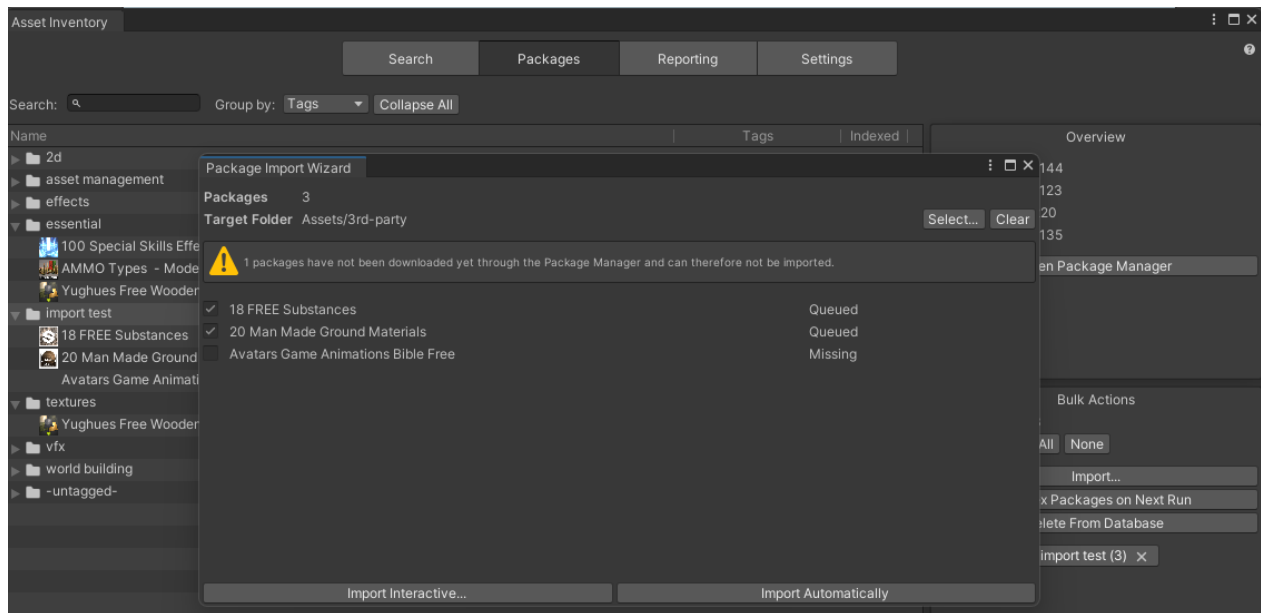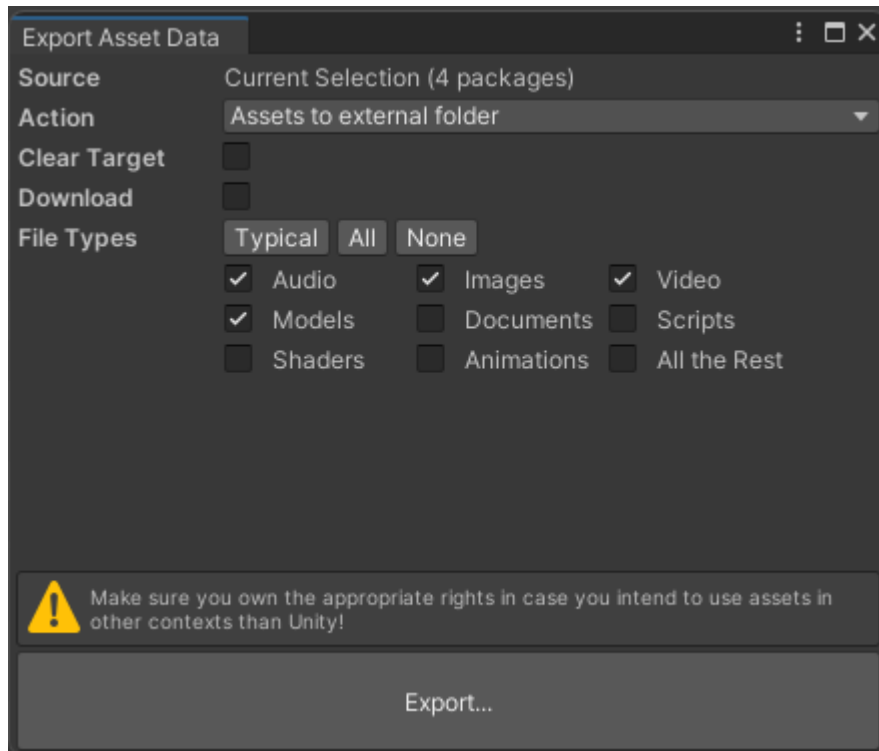

Importing an archive will extract the archive into the target folder.

EXPORT

Exporting packages can also be done one-by-one or in bulk. The export option (hold down CTRL key) will bring up an export wizard from where the file types to export can be selected. This way it is easy to reuse assets in other contexts.

ADVANCED

In case packages should not appear in the search results and not be indexed, the **Exclude** toggle can be used in the package details. Excluded items will disappear from the list and can be reactivated using the "*Excluded*" maintenance view.

Bulk selection will show the total size of the compressed assets before they are downloaded so that you can estimate if temporarily downloading all for indexing would still fit on the hard drive. If available also costs are shown (current, non-discounted).

Clicking the icon in the lower right corner will toggle the details view into an **expanded view** twice as big. Depending on availability it will now also display media, description, release notes and dependencies (for registry packages).
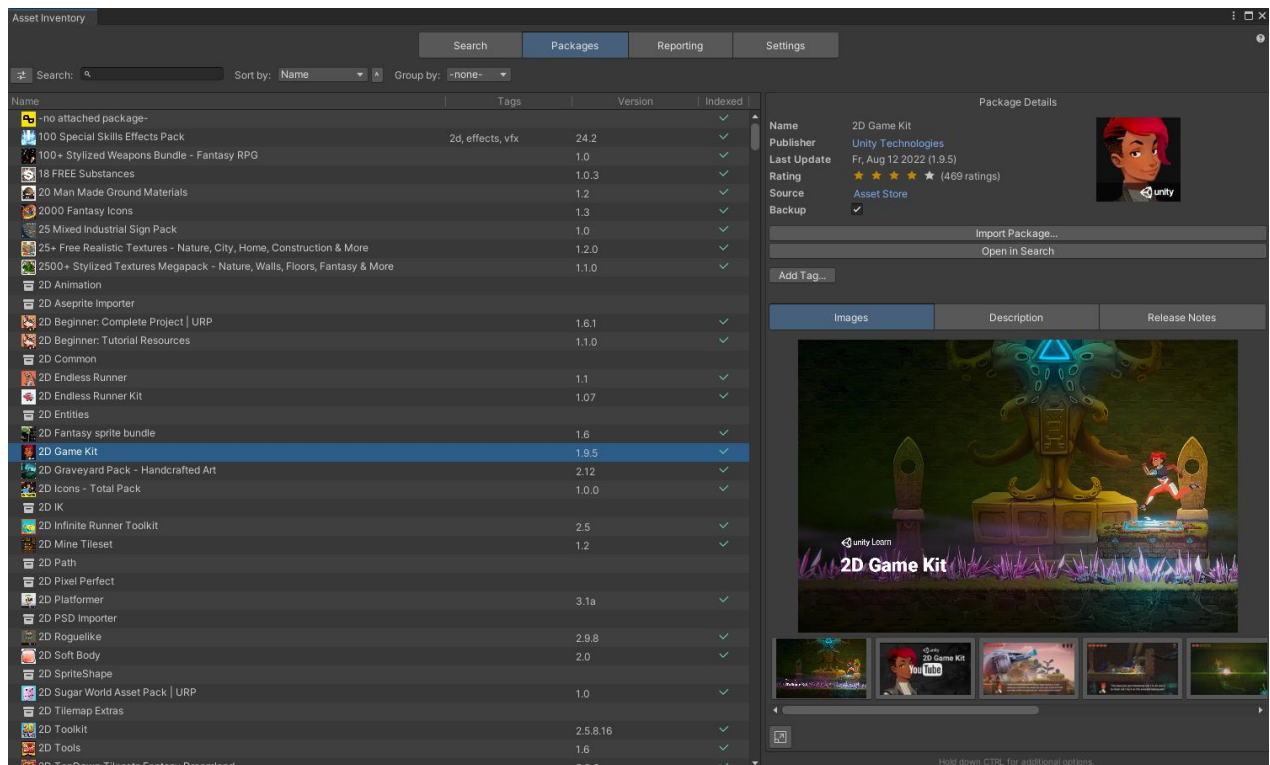
Holding CTRL will reveal much additional package data and also many actions.

REPORTING

OVERVIEW

This module will scan the complete project and try to identify packages that were being used. It also supports exporting your data and listing used licenses.

Packages that could be identified with 100% **confidence**, where also the version is guaranteed to be correct, will be shown with a **bold** version column. All registry packages automatically fall into this category. From Unity 2023+ also local assets are typically identified correctly if imported through Unity 2023+, since it will store the asset origin in the meta data files allowing to pinpoint the exact version correctly.

When selecting an asset in the Unity *Project View*, the reporting tab will try to identify the associated package in the lower right info box.

DATA EXPORT

In case you want to use your own analysis or presentation style it is possible to export the database contents into other formats. Currently supported is:

- CSV: can easily be handled in Excel and other compatible software
- Assets: see section above in the Packages chapter
- Licenses: will list all packages with a non-default license in MD format
- Overrides: metadata which can be used to override package-specific metadata in team scenarios (e.g. to enforce common tags & categories)

SETTINGS

OVERVIEW

Use this section to configure what should be indexed. Indexing can be started and stopped at any time and will pick off again where it last stopped.
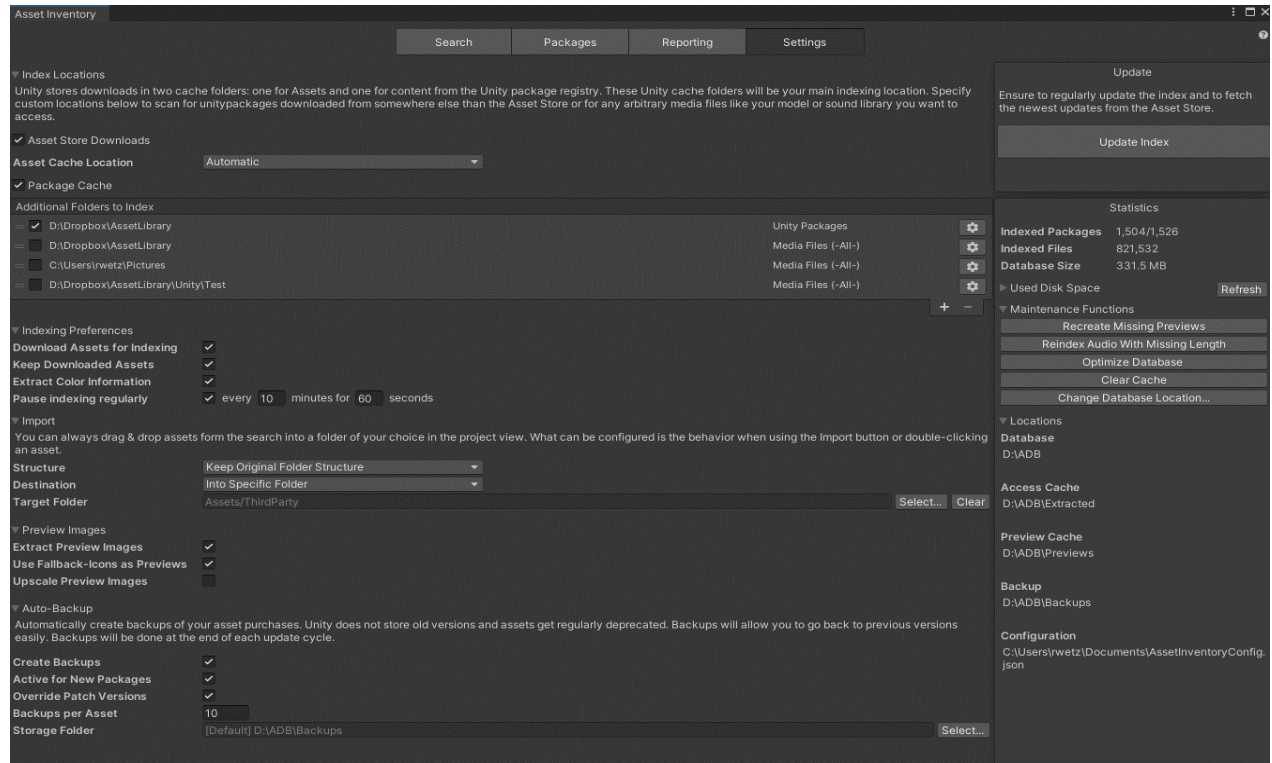
Asset Store downloads and Packages are activated by default. There are two options available:

Index everything that is already downloaded into the local cache.

- That means you only need to click the **Download** button in the **Package Manager** window or right in the Asset Inventory for each asset you own (without importing them) to make them available for the index (bulk selection is also possible).
- If you have specified a **custom asset cache directory** in Unity 2022.1 or higher, it will typically be auto-detected. Press CTRL to see the resolved path and verify. You can manually override the location by switching the Asset Cache Location to *Custom*. If the environment variable *ASSETSTORE_CACHE_PATH* is set it will override the default asset cache location.

Automatically download purchased assets for indexing and delete them again afterwards.

- With this option you will **index everything you have** without the need to download everything, saving a lot of disk space, at the expense of a potentially very long indexing due to all the necessary downloads.

During a normal update cycle (whenever "Update Index" is clicked) the following actions are done in this order (shown also when clicking "What will happen?"):

- Fetch all Asset Store purchases
- In the background, fetch details for all purchases and for assets that received updates
- In parallel scan for new local Asset Store packages and update metadata
- Do the same for registry packages
- Index the contents of new or changed packages
- Index any additional folders
- Index Unity Asset Manager cloud content
- Extract colors of new assets for color search
- Create AI captions
- Perform backup activities

All activities are incremental, meaning you can **interrupt and restart these at any time** and they will mostly continue where they left off.
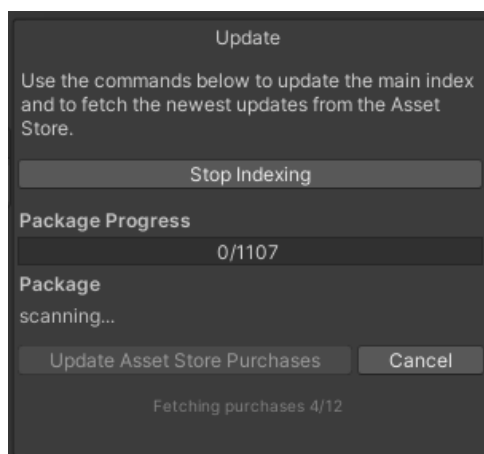
Once an asset is indexed it does not need to remain on your hard drive if you just want to search for it. Only when importing it needs to be available.

LOCAL FOLDERS

If you want to index Unity packages downloaded from **other sources**, simply add the folders to the **additional folders** list. When selecting an entry from the list of additional folders, more options can be specified on the right-hand side. This way it is possible to select what types of files should be searched for:

- **Unity Packages**: Finds any .*unitypackage* files in the folder and indexes the content.
- **Development Packages**: Finds any *package.json* files and treats those as local packages.
- **Media Files**: Allows to index all kinds of other, **free-floating** assets, like images, models, audio libraries etc. Also Unity projects can be indexed this way.
- **Archives**: Will extract archives (zip, rar, 7z) on the fly and index all contents.
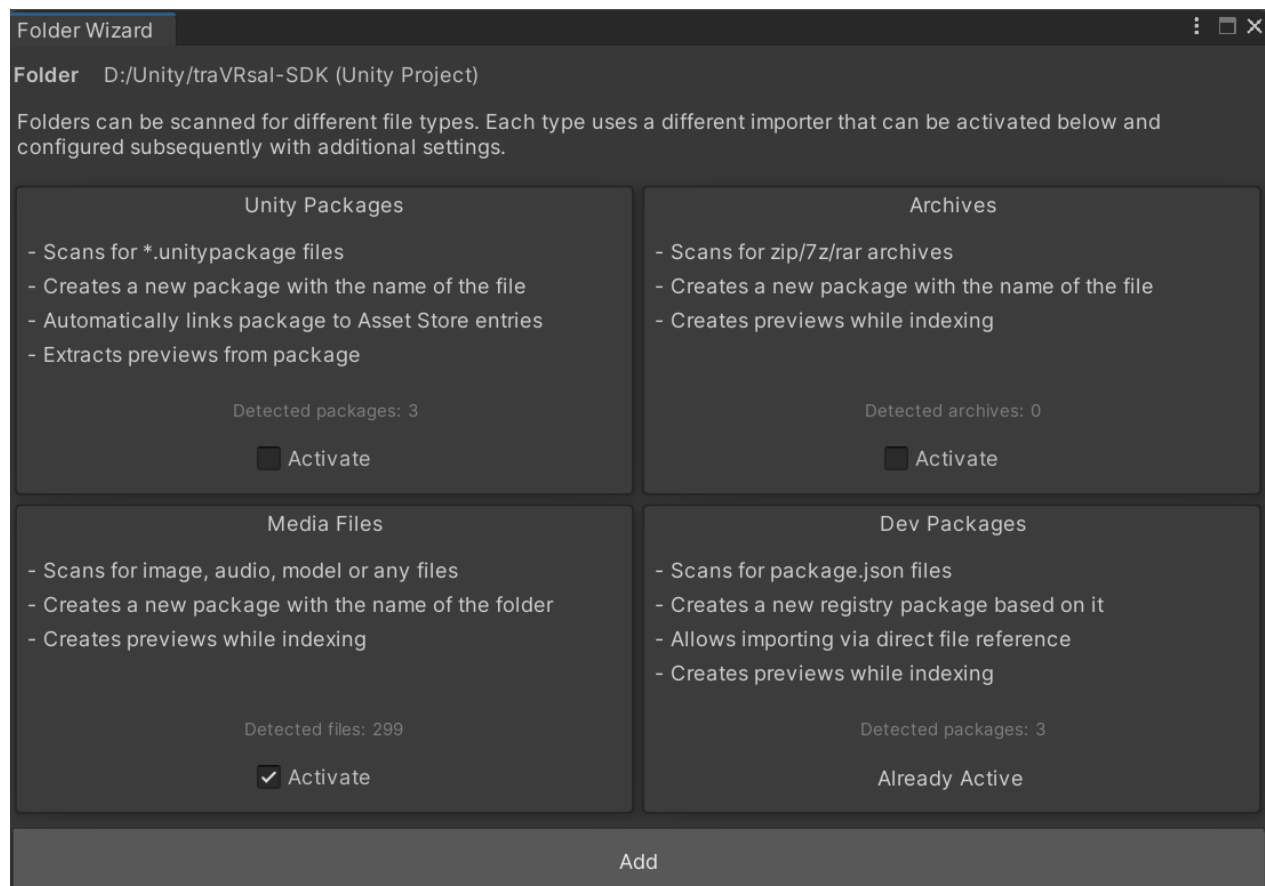
Using additional media folders will let Asset Inventory act as the go-to place for asset management and quickly finding any available asset on your drives from a single, consistent UI. Media files can optionally have no connected asset but otherwise behave the same. If they have a .*meta* file next to them, their guids will also be stored.

The order in which additional folders are processed can be changed by dragging them up or down in the list. Deactivating or removing an additional folder entry will not remove it from the index.
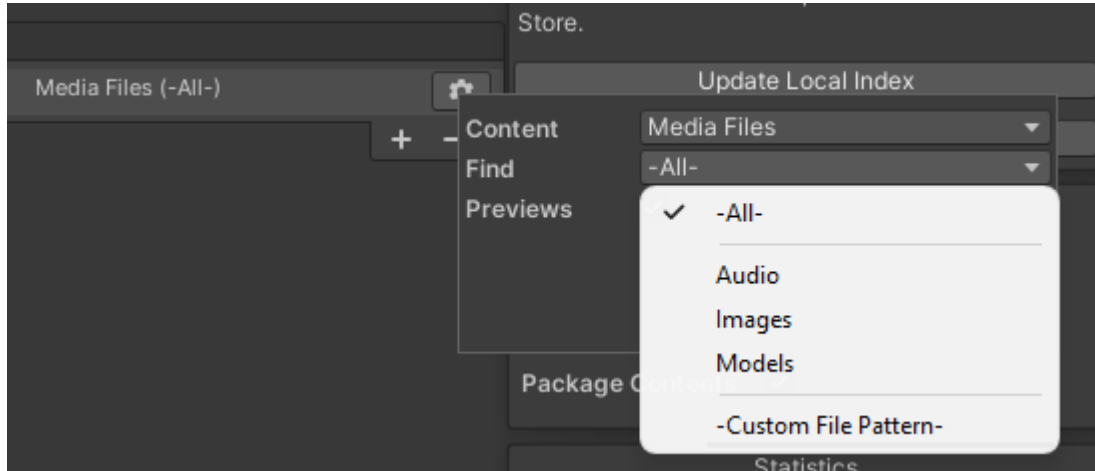
PREVIEWS

The search result will show previews of the assets. These previews can come from three sources:

- Included in the package and created at the time the asset was built (state: pre-provided)
  - This is done by the Asset Store tooling and the reason why e.g. previews for sound files can look different depending on which version of Unity was used.
  - Sometimes these previews can be empty when exported incorrectly.
- Created by Asset Inventory during import (state: recreated)
  - This happens automatically for atomic assets without dependencies, e.g. sound files, textures or models.
- Created by Asset Inventory on-demand (state: recreated)
  - This mode can also create previews for prefabs and materials but will take **much** longer since all dependencies need to be materialized first.

It is possible to recreate a single preview from inside the search view, all previews for a package or perform recreation for missing previews in the complete database. There is a **Preview Wizard** to help with all steps. Previews can also easily be switched between provided and recreated there in case of issues, e.g. incompatible render pipelines.
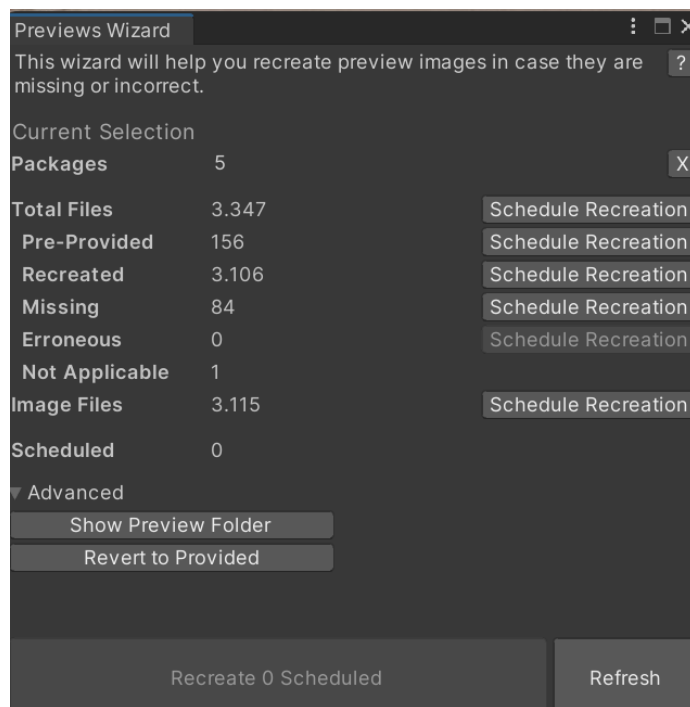
Preview file generation will work by temporarily copying each file into the current Unity project, letting Unity create a preview and removing it again. While creating previews it might happen that a new folder called _TerrainAutoUpgrade will appear under Assets as a side-effect. This process will take a bit of time but will allow Asset Inventory to show preview images and also additional meta data for many file types.

Image files will be handled separately for common file types, bypassing Unity, which is much faster and will, if active, automatically scale preview images to a higher resolution already during indexing. This will work for Unity 2021.2 and above.

Technical Details

Previews are mostly created by the Unity Editor. This works well but is limited to 128 x 128 pixels. An upscale option is available in the *Settings* which produces bigger previews for image files. For all other types this will reduce visual quality (tick the *Lossless* upscale option to only upscale images).
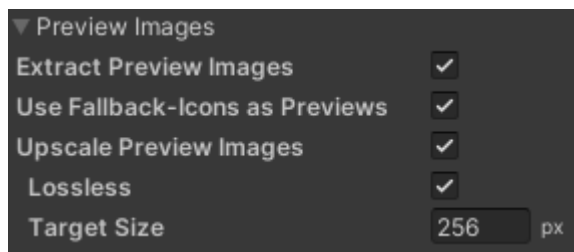
VFX and 2D prefabs cannot be previewed this way. A custom mechanism is under way for future releases.

BACKUP

The tool supports copying packages to a dedicated place for permanent backup. This way you are not affected by package deprecation by Unity where it might be impossible to download older (but for you working or compatible) versions of assets anymore.

The backup tool can keep multiple versions of packages and will automatically handle the life cycle, version comparison and more.
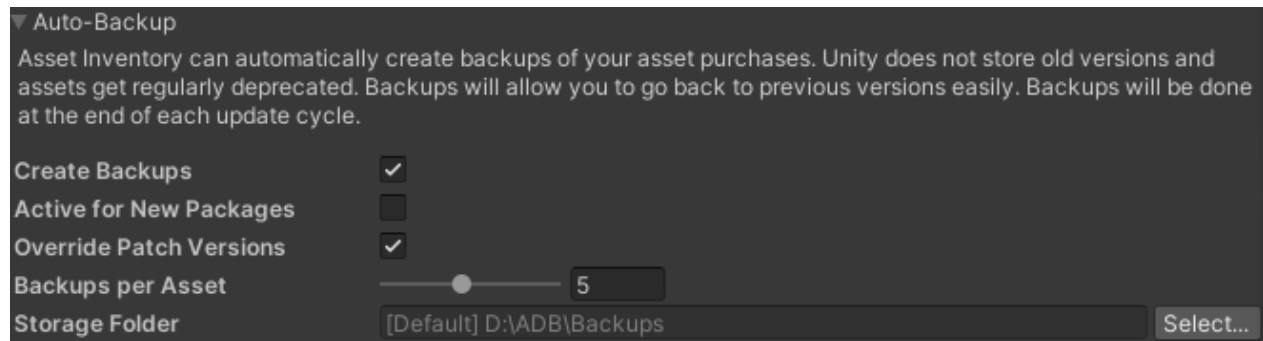
Once backup is activated it can be set individually per package in the package details and in the next "Update Index" run will back up the selected packages.
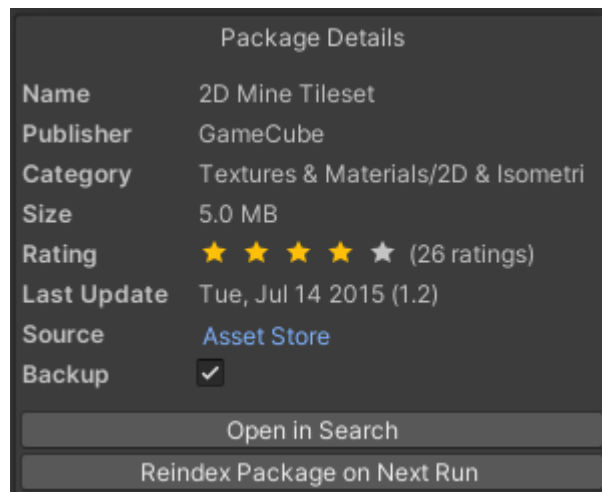
ARTIFICIAL INTELLIGENCE

Using AI it is possible to automatically create captions that describe individual asset files. This way you are also able to find files which are labeled inappropriate or insufficient. See the example below:
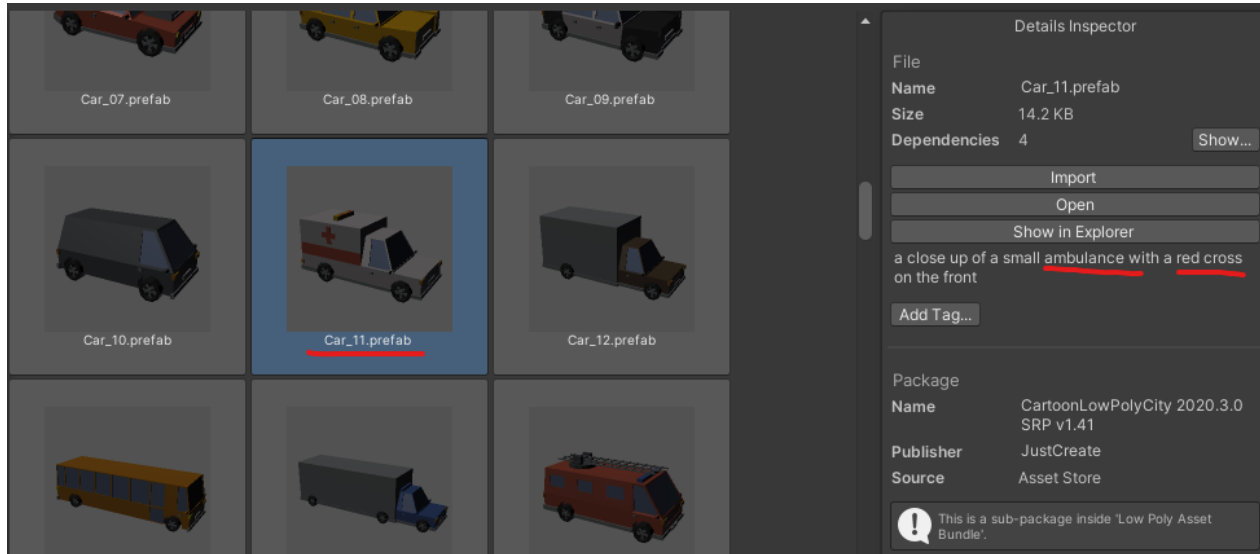
While the asset is simply called *"Car_11"*, the AI has captioned it *"a close up of a small ambulance with a red cross on the front"*.

Searching for *"ambulance"* now and using the AI generated captions during the search will also yield *"Car_11"* as a result, which would have been impossible to find before.

Setup

The basis for AI captioning are the existing preview images. So make sure you have performed preview recreation where applicable.

AI inferencing will happen on your own device. This means it does not use any remote server and also does not incur costs for a third party service. It requires a potent PC though as the process is resource intensive. The best free currently available image captioning model is Salesforce Blip, coming in a *base* and a *large* version.

To use it locally, Python 3, pipx and the <u>blip-caption tool</u> are required to be installed. These are the basic steps assuming basic familiarity with the computer system (assuming Windows below but will work for other OS similarly):

- Install <u>Scoop</u>
    - Open a new PowerShell window (no admin required)
    - Run: *Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser*
    - Run: *Invoke-RestMethod -Uri https://get.scoop.sh | Invoke-Expression*
- Switch to a normal Command line window
    - Run: *scoop install pipx*
    - Run: *pipx ensurepath*
    - Run: *pipx install blip-caption*
    - Run: *pipx ensurepath*
- Restart the system

```
C:\Users\      >scoop install pipx
Installing 'pipx' (1.7.1) [64bit] from 'main' bucket
pipx.pyz (321.6 KB) [===================================================================================] 100%
Checking hash of pipx.pyz ... ok.
Running pre_install script...done.
Linking ~\scoop\apps\pipx\current => ~\scoop\apps\pipx\1.7.1
Creating shim for 'pipx'.
'pipx' (1.7.1) was installed successfully!
'pipx' suggests installing 'python'.

C:\Users\      >pipx ensurepath
Success! Added C:\Users\      \.local\bin to the PATH environment variable.

Consider adding shell completions for pipx. Run 'pipx completions' for instructions.

You will need to open a new terminal or re-login for the PATH changes to take effect. Alternatively, you can source
your shell's config file with e.g. 'source ~/.bashrc'.

Otherwise pipx is ready to go! ✨ 🌟 ✨

C:\Users\      >pipx install blip-caption
⣷ installing blip-caption
```

There is support for GPU acceleration but it requires a <u>custom patch of the blip tool</u> to be installed. You also need to make sure to have PyTorch with GPU support as well as a CUDA runtime installed. Instructions can be found on the <u>official website</u>.

Once that is done you can activate *GPU* (advanced option, hold CTRL), which dramatically speeds up the generation of AI captions (especially with larger bulk sizes).

Usage

On the *Settings* tab the successful installation can be tested.
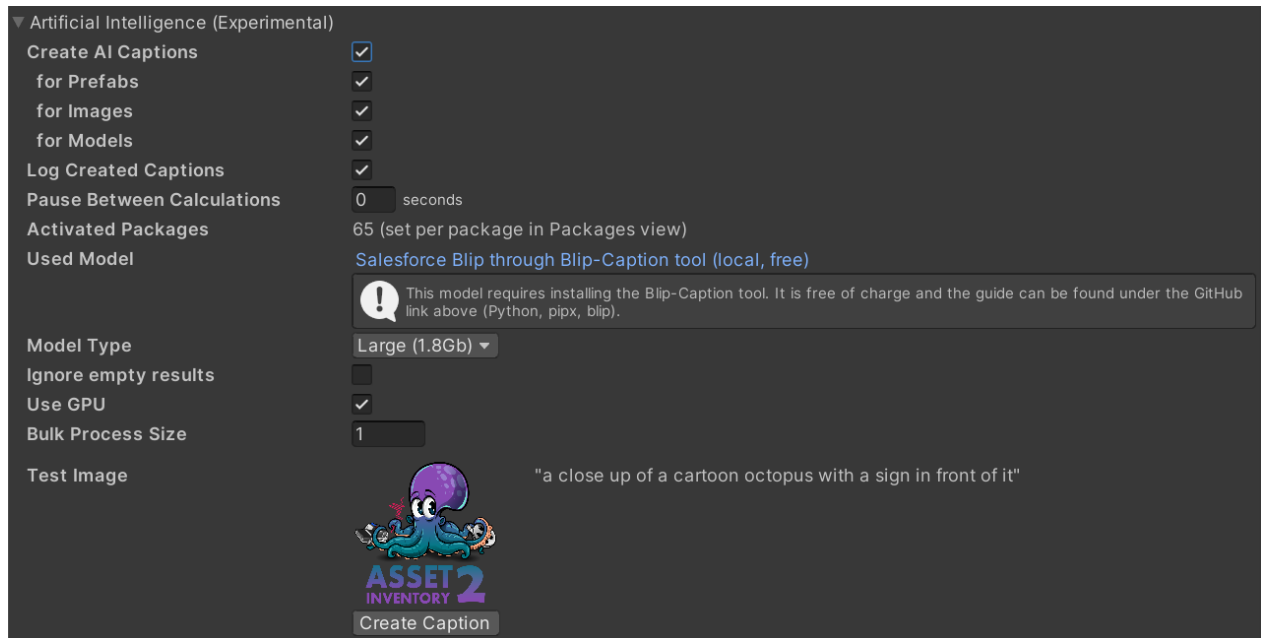
Good bulk sizes are 50 to 100. Higher is more efficient, but will produce less progress information.

Once activated, the tool will create AI captions during the next Update cycle for packages where *AI Caption* was turned on in the *Package* view.
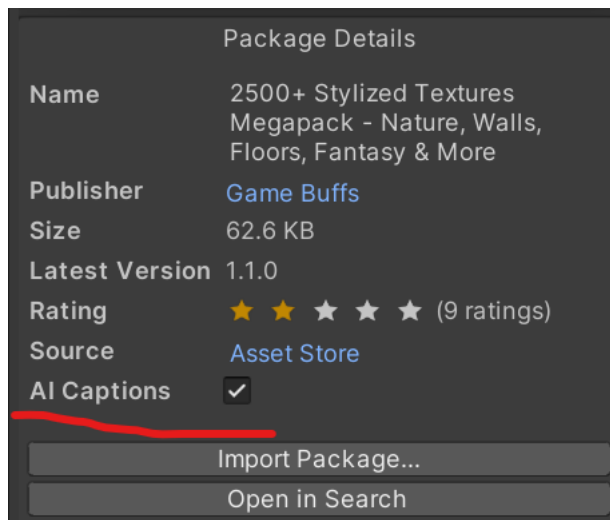
AI captions will be shown in the *Search* results when selecting a file. You can also create individual captions directly from the *Search* results.
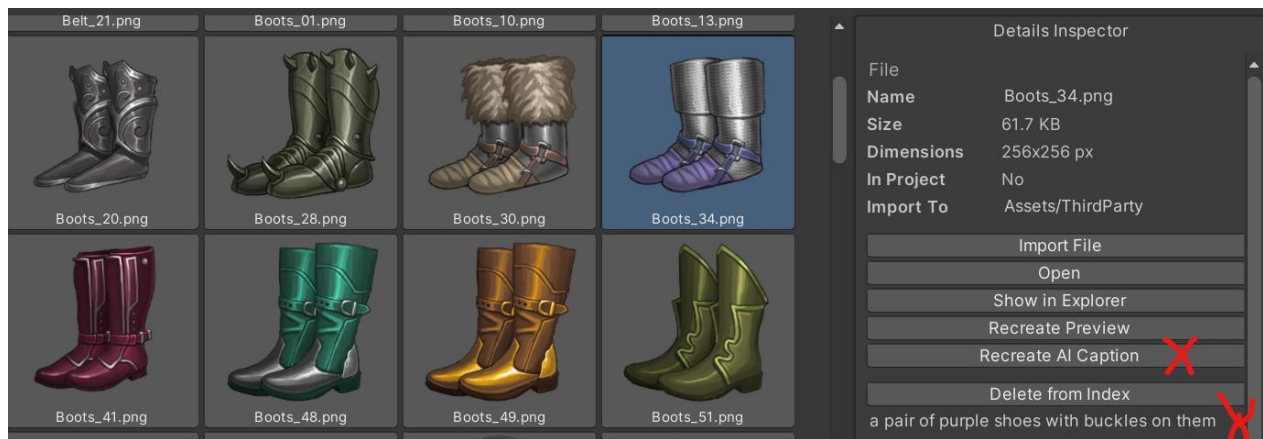
There is also an option to search in the AI captions in the search settings and the AI caption can also be shown as the tile text if wanted.
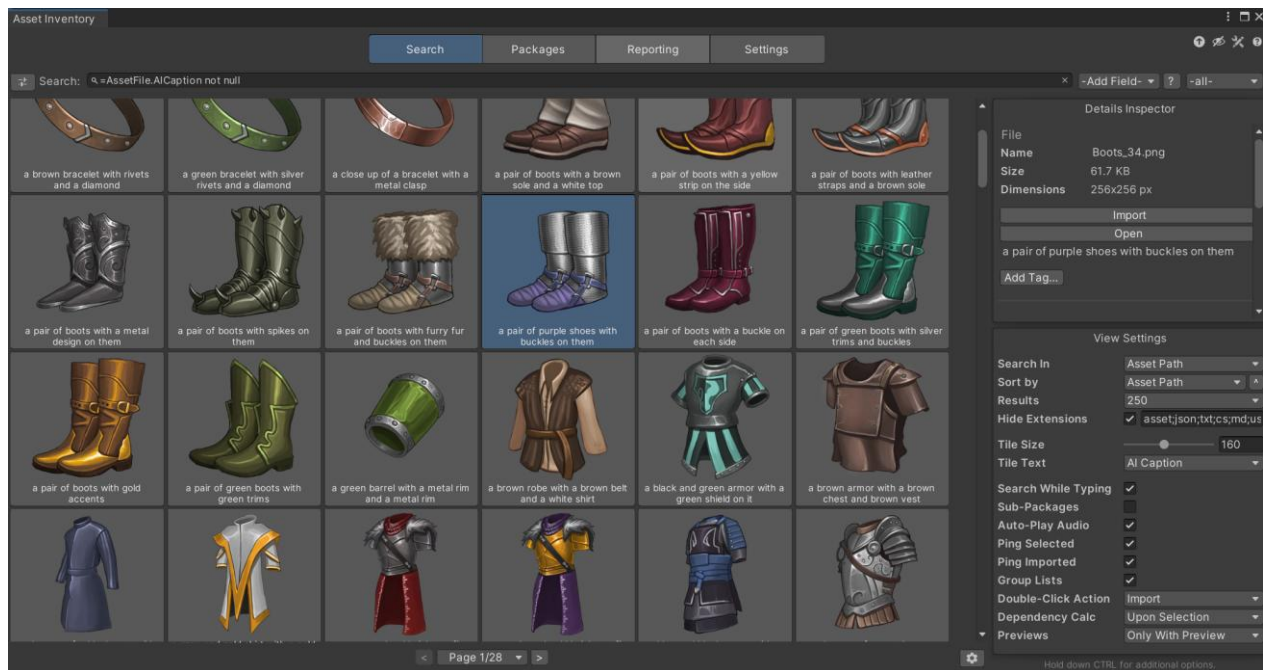
Using an expert search like the one below will return all files for which a caption was already created:

=AssetFile.AICaption not null

CROSS-DEVICE USAGE

It is possible to use the same database from multiple devices. This way the indexed assets can be browsed and seen from every device. The easiest way to achieve this is to make the database and all asset folders available through a mounted network drive. If the drives and folders are identical from each device, it will instantly work this way. Also if Asset and Package cache folders differ the database will be compatible since these paths will be stored with *[ac]* and *[pc]* keys (see below). Furthermore, all paths are stored with forward slashes, making the database compatible between different operating systems.

Sometimes drives or folder mappings cannot be kept identical for various reasons. The tool supports this scenario through a mechanism called **Relative Persistence**. Each additional folder entry is replaced with a **key** and all occurrences in the database to this folder are also replaced with this key. On each device the key can then be **mapped** to the required drive and folder. This is a one-time action and once done the tool can be used as always.
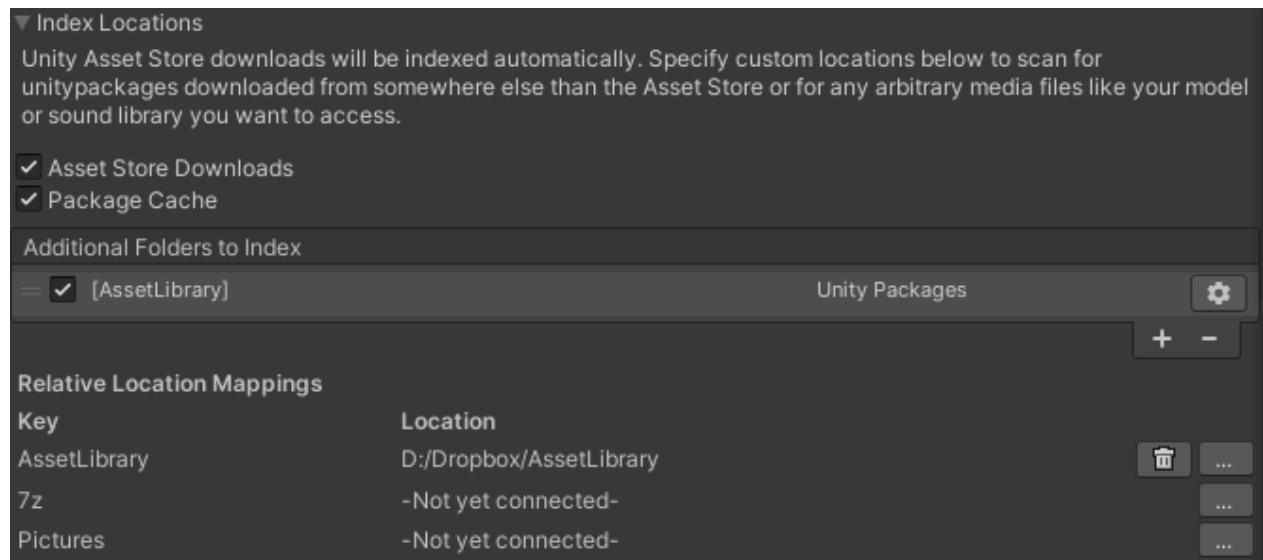
In order to convert a folder to relative format, open the folder settings, hold down CTRL and select **Enable...**
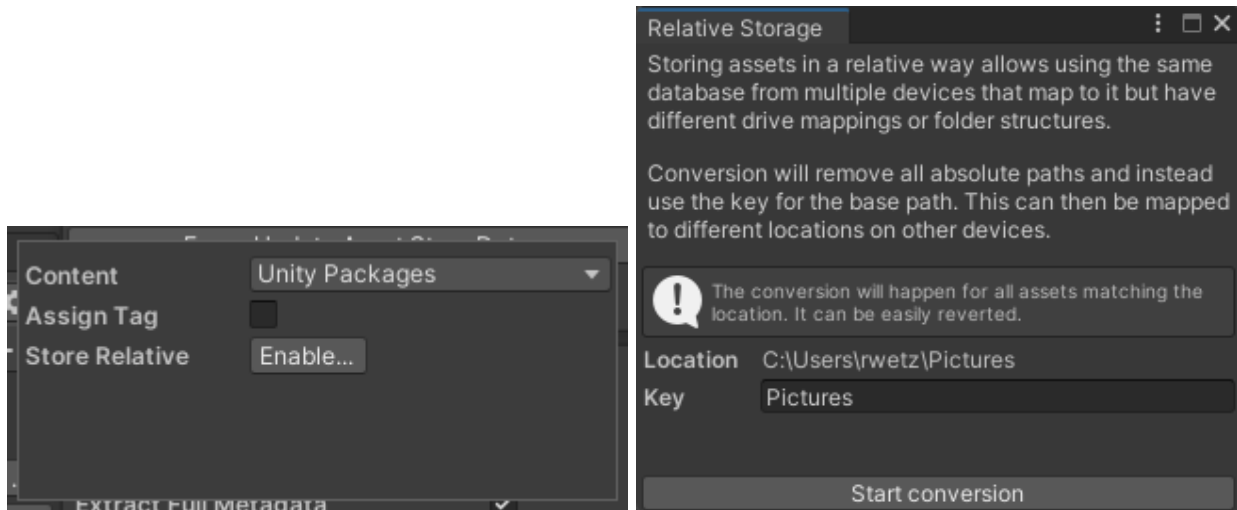
If a folder is already converted it can be **switched back** again through the same mechanism. This will remove all mappings and reorganize the database to contain the full paths again.
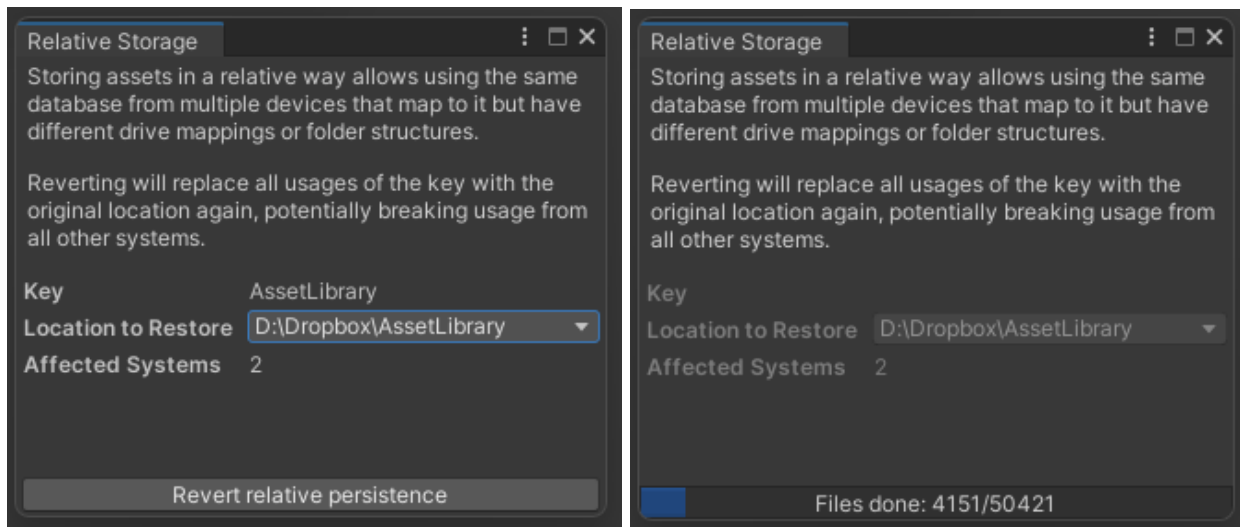
MAINTENANCE

Maintenance functions are also located on the Settings tab:

Harmless & recommended to run every now and then

⇨ Maintenance Wizard: will scan for common database and file system inconsistencies and offer automatic solutions
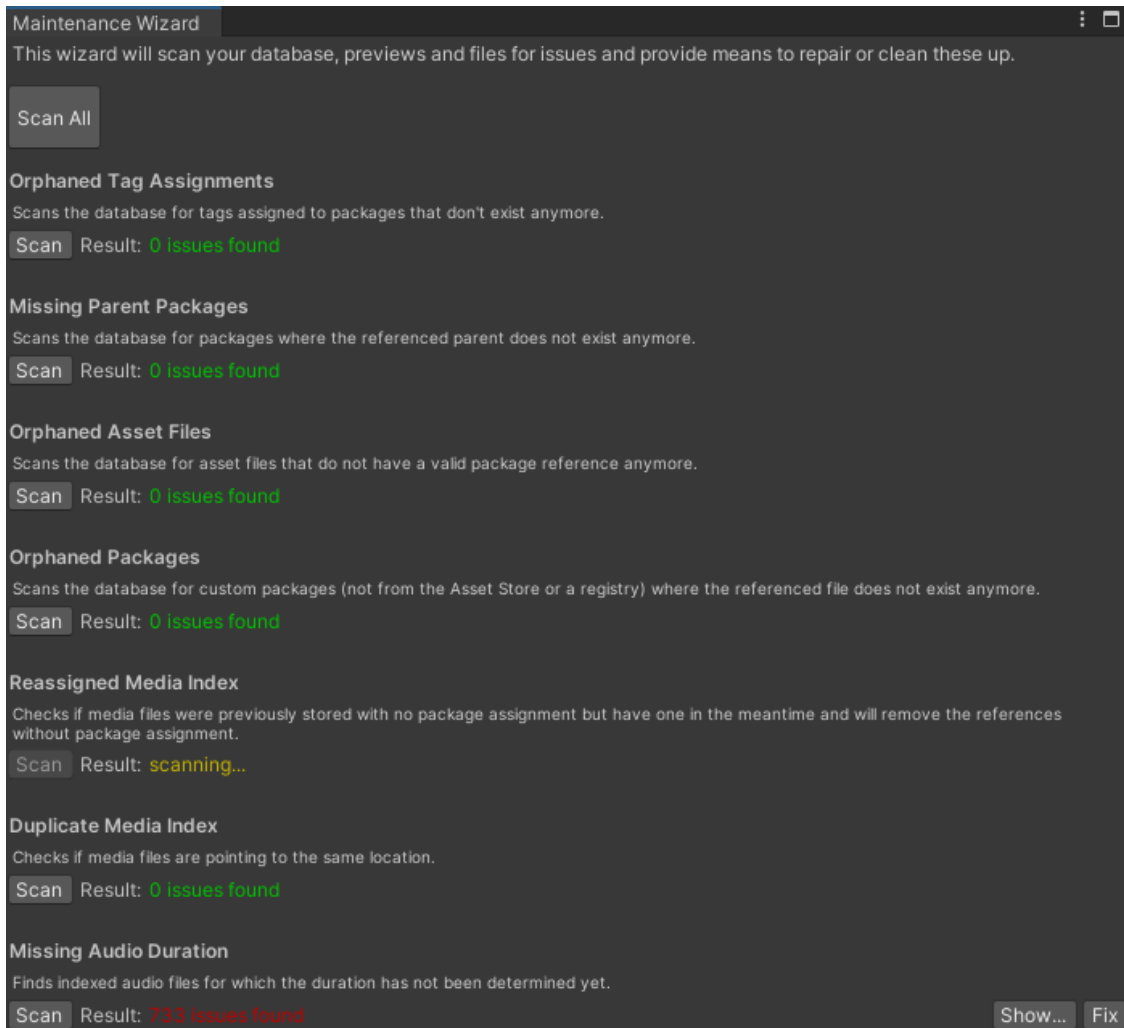
⇨ Previews Wizard: will try to create preview images for asset files which do not have ones so far or were flagged to be recreated. This can take quite some time especially for complex prefabs.

⇨ Optimize Database: compacts and reorganizes the database for more performance and less required space, should be done after bigger indexing activities

Harmless

- ⇨ Close Database: allow backing up or copying the file
- ⇨ Change Database Location: set a different location for the database, move the current, use another existing one or create a new one
- ⇨ Clear Cache: deletes the temporary files to save space, can be done without any side-effects

Potentially destructive

- ⇨ Clear Database: deletes the complete database to start over
- ⇨ Reset Configuration: set everything to like it was initially, removing any additional configuration that was done

ADVANCED FEATURES

EXPERT FUNCTIONS

Many more seldom used and advanced features are initially hidden to not clutter the UI. They can be made visible by holding down the **CTRL key**.

Normal View   Holding CTRL Key
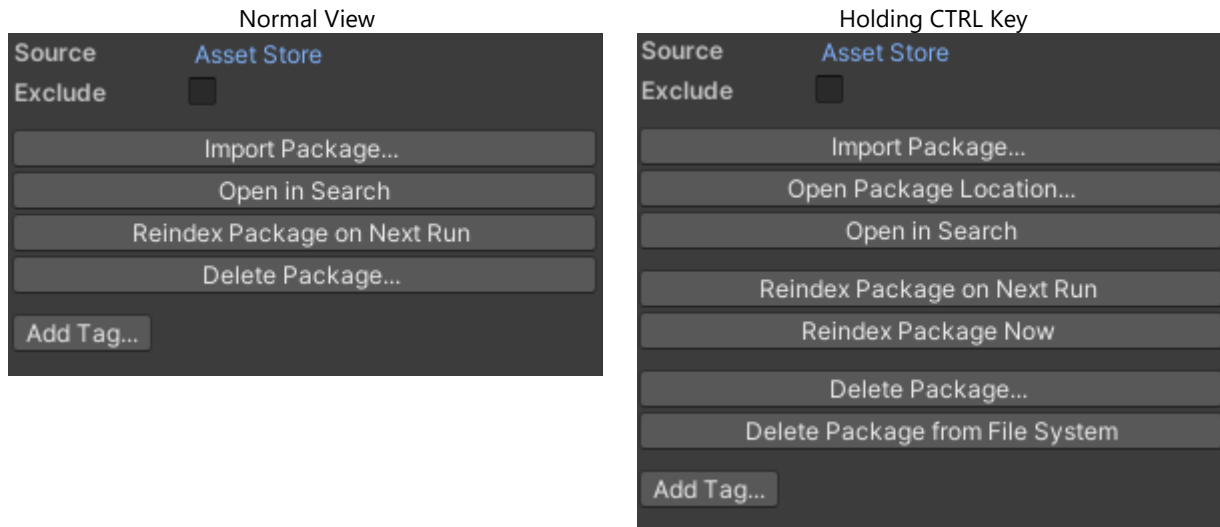
Expert function visibility will not only apply to buttons but also to **additional information** which is not often needed. This ensures the UI is clean for the 90% use-case while also catering to experts on-demand. There are two ways to influence the visibility of the advanced UI:

- Temporarily toggle it on via the eye icon in the upper toolbar or permanently via a setting in the *Advanced* section on the settings tab.
- Customize what should be considered "advanced" (see next section).

UI CUSTOMIZATION

Using the customization icon in the upper toolbar, the UI can be switched to *design-mode* (and back). When active, sections (labels, buttons, info boxes, panels...) that can be customized will be highlighted in green and red and additional actions will be visible. Green sections will always be shown. Red sections will be considered *advanced* and only shown when advanced UI is toggled.

During design-mode, all advanced UI will automatically be shown. Depending on the current selection and context, not all UI elements will be visible (e.g. registry packages will show different actions than Asset Store packages).

AUTOMATION & API

The tool can be controlled via scripting. This happens through the static methods of the *AI.cs* class and through a dedicated search API. The existence of the tool can be detected via the automatically added script define *ASSET_INVENTORY*.

To use the powerful **search** UI features (through the *ResultPickerUI*), the following code can be used (it is also provided in the *Examples* folder). Optionally a search type and a search string can be supplied to narrow down the search.

Two modes are available: close upon selecting a result tile or explicitly using the *Select* button. Once the user selects something, a callback with the path of the asset inside the project will be invoked. The asset will automatically be materialized.

The *ResultPickerUI* offers multiple methods:

Show()

will return the path of the selected asset inside the current project.

ShowTextureSelection()

Will default to "Images" as search type and return a dictionary of files. The selected entry will be inside "original". The tool will do a heuristic analysis of additional adjacent files that belong to the same texture, like normal maps, metal maps etc. If found, they will also be materialized and returned.
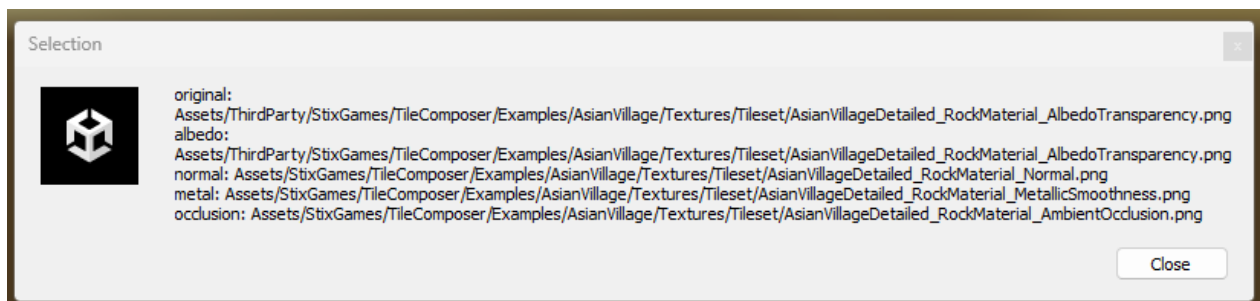
Supported Types: albedo, normal, specular, metal, occlusion, displacement, height, emission, reflection, alpha, mask, roughness

```csharp
using System.Linq;
using UnityEditor;
using UnityEngine;

namespace AssetInventory
{
    [CustomEditor(typeof (OpenSearch))]
    // Editor    Robert Wetzold *
    public class OpenSearchEditor : Editor
    {
        //  Robert Wetzold *
        public override void OnInspectorGUI()
        {
            #if ASSET_INVENTORY
            GUILayout.Label("UI Examples", EditorStyles.boldLabel);
            if (GUILayout.Button("Search for a car..."))
            {
                ResultPickerUI.Show(path =>
                {
                    EditorUtility.DisplayDialog("Selection", path, "Close");
                }, "Prefabs", "car");
            }
            if (GUILayout.Button("Search with details..."))
            {
                ResultPickerUI window = ResultPickerUI.Show(path =>
                {
                    EditorUtility.DisplayDialog("Selection", path, "Close");
                });
                window.instantSelection = false;
                window.hideDetailsPane = false;
            }
            if (GUILayout.Button("Search for texture sets..."))
            {
                ResultPickerUI window = ResultPickerUI.ShowTextureSelection(path =>
                {
                    EditorUtility.DisplayDialog("Selection", string.Join("\n", path.Select(e => e.Key + ": " + e.Value)), "Close");
                });
                window.instantSelection = false;
                window.hideDetailsPane = false;
            }
            EditorGUILayout.Space();

            GUILayout.Label("Programmatic Examples", EditorStyles.boldLabel);
            GUILayout.Label("soon", EditorStyles.miniLabel);
            /*
            if (GUILayout.Button("Search for all cars..."))
            {
                // TODO: Move PerformSearch from SearchUI to AssetInventory.cs
            }
            */

            #else
                EditorGUILayout.HelpBox("This feature is only available if Asset Inventory was imported into this project.", MessageType.Info);
            #endif
        }
    }
}
```

DATABASE ACCESS

It is also possible to **connect to the database** directly and perform your own analysis and queries with the collected data. It is in **SQLite** format. A good viewer is for example DB Browser for SQLite.

CONFIGURATION

OVERVIEW

Settings will be saved automatically in a file called *AssetInventoryConfig.json*. This happens by default in the *Documents* folder of the currently logged in user. This way the tool can work independently of the Unity project and the asset search is available in an identical way everywhere. Two alternatives are available:

- It is possible to force the tool to use a project-specific configuration. To do so, copy the *AssetInventoryConfig.json* file anywhere into the project and restart Unity. The detected location will be shown on the **Settings tab** under **Maintenance Functions**.
- Using the environment variable *"ASSETINVENTORY_CONFIG_PATH"* a custom location (just the folder name) can be specified.

Some advanced settings can only be set in the *.json* file and are not otherwise accessible through the UI. An example for this is the delay after which the search will start searching. Feel free to adjust these values as well.

INSTALLING AS A PACKAGE

Especially when managing many Unity projects keeping Asset Inventory up-to-date can mean spending quite some time on manual package updates. An alternative is to install the tool in one Unity project only and then referencing this single installation via package reference in the *manifest.json*. This way all Unity installations always use the same tool version.

The easiest way to set this up is to open the manifests from the *Packages* folder where the tool should be referenced and adding it directly there. The path can either be absolute or relative. Example:

"com.wetzold.asset-inventory": "file:../../MySourceProject/Assets/AssetInventory",

ADVANCED SETTINGS

In order not to clutter the UI some seldom used settings are to be set directly in the configuration file. It is in formatted JSON format and can be opened in any text editor. The following properties do only appear there:

- **cacheFolder** *(string)*: location where the cache should be located, e.g. "F:/AICache". Default is with the database under *"Extracted"*.
- **centerTiles** *(bool)*: will center grid views instead of left-aligning within window
- **dbJournalMode** *(string)*: defines the journal mode to be used with the SQLite database. Default is "WAL". Use "DELETE" for a more conservative method in case of database issues. Other options [see link](#).
- **filterOnlyIfBarVisible** *(bool)*: will only apply additional search filters if the filter bar is expanded (legacy pre-version 2 behavior)
- **hueRange** *(float)*: degrees to widen color search
- **maxResultsLimit** *(int)*: hard limit of maximum results to be shown even if *-all-* is selected
- **mediaHeight** *(int)*: size in pixels for big media image
- **mediaThumbnailWidth** *(int)*: size in pixels for width of media thumbnails
- **mediaThumbnailHeight** *(int)*: size in pixels for height of media thumbnails
- **rowHeightMultiplier** *(float)*: adjustment factor to the height of table rows
- **searchDelay** *(float)*: seconds after typing a character to wait until triggering search
- **showExtensionList** *(bool)*: flag if to list all indexed extensions in the Types dropdown on the search. Can result in quite some lag while typing if the database is large.
- **showHints** *(bool)*: flag if to show additional usability hints in the UI

THIRD PARTY

USAGE

Asset Inventory uses multiple third-party libraries that really make it shine:

- Package2Folder by Code Stage uses an ingenious idea to intercept the Unity package manager and adjust the target folder where to install assets to.
- SQLite is a great way to store data in a single file while providing easy and fast database operations.
- SQLite-Net is an amazing extension to SQLite offering convenient high-level functions when executing SQL queries.
- Editor Audio Utils does a great job to allow playing audio files also while not in play mode.

INTEGRATION

Assets that already integrate with Asset Inventory:

Gaia from Procedural Worlds

Makes it really easy to pick new prefabs to spawn and textures to use in your procedural content. Access all your assets quickly with just a few clicks. The corresponding selectors will appear automatically once you have both tools imported in your project.

TECHNICAL DETAILS

The index will be stored in an SQLite database. This database will be automatically created upon first usage. It is in your *Documents/AssetInventory* directory. The size should be rather small but increases with the number of indexed assets. As a rule of thumb assume 20Mb database + 200Mb preview images per 30k files (if not upscaled).

Inside the *AssetInventory* directory, three additional folders will be created:

- **Previews**: containing all images for the asset catalog
- **Extracted**: containing temporarily extracted files. You can safely delete the *Extracted* folder at any time if needed.
- **Backup**: acting as the default location for package backups

Some operations require Unity to perform work, e.g. creating previews. For such tasks, the folder *_AssetInventoryPreviewsTemp* will appear under *Assets* for a short time and will be automatically removed again once done.

LIMITATIONS

- Dependencies can only be calculated in packages that use *text/yaml* serialization. A few legacy packages still use *binary* serialization. These packages also work in most cases except if Unity cannot resolve them, e.g. because of missing scripts on prefabs.
- Importing scripts is experimental and can produce console errors if the script requires other scripts or assembly definition files to run.
- Running the game view maximized on play and having the Asset Inventory window docked will cause it to reinitialize, resetting all search settings.

FAQ

SQLITE IS ALREADY IN MY PROJECT (CONFLICT)

The only requirement is that SQLite is available. In this case it is safe to delete the one supplied by Asset Inventory. Close Unity and delete either the full *ThirdParty/SQLite* folder or if your own package doesn't bring *System.Data.SQLite* only delete *ThirdParty/SQLite/Plugins/x64 & x86*.

SOME PREFABS SHOW NO DEPENDENCIES

This is due to the serialization format. Only assets serialized as text can properly be parsed right now.

SOME AUDIO FILES USE DIFFERENT COLORS IN THE PREVIEW THAN OTHERS

Preview colors depend on which Unity version an asset author used to upload his asset. Unity seems to change colors over time between Unity versions. There could be a feature someday to recreate preview images with the current version of Unity to make them all uniform depending on interest.

SOME PREVIEWS ARE GREY AND DON'T SHOW ANYTHING

This can happen if the Asset Store tooling did not create a correct preview image while uploading an asset. An example is the Danger Zone asset from Unity released for 2021+. Functionality to recreate preview images is under development.

CONSOLE SHOWS ERRORS DURING INDEXING

There are a couple of errors Unity creates that cannot be intercepted but can safely be ignored. These are:

Could not determine image dimensions for 'filename': Not enough memory to complete operation [GDI+ status: OutOfMemory]

- Can happen when either the image dimensions are too big (5000px+), it is actually a different format (incorrectly named file) or the image file has a sub-format that Unity cannot read, e.g. special types of *png*.

"Cannot create FMOD::Sound instance for clip "" (FMOD error: Unsupported file or audio format.)"

- It typically means that an audio clip was saved with the wrong extension, e.g., a wave as an ogg by an asset publisher. Use the **Open** button to start the native file player which will typically play the file correctly then.

"TEXTURE HAS OUT OF RANGE WIDTH / HEIGHT"

- In that case Unity cannot read the texture dimensions and you will not be able to see the dimensions of the texture or filter for it. Otherwise, it does not have any effect.

"ArgumentException: Getting control 4's position in a group with only 4 controls when doing repaint"

- This can happen if a lot of UI changes are going on while indexing is happening. Can safely be ignored and is without any effect.

"Curl error 28: Operation timed out after 30000 milliseconds with 0 bytes received"

- This can happen if Unity servers did not respond in time and the details for an asset could not be retrieved. This is solved by starting the Asset Store update process again.

"Invalid or expired API Token when contacting…"

- Happens if Unity was not in use for a longer period. The user login happens in the Unity hub and in that case the token for online access has expired. Solved by restarting Unity.

"Could not create asset from Assets/_AssetInventoryPreviewsTemp/1016771.png: File could not be read"

- Happens if Unity cannot create a preview for specific assets or if this is a malformed png. Can be ignored.

"TLS Allocator ALLOC_TEMP_THREAD, underlying allocator ALLOC_TEMP_THREAD has unfreed allocations, size 1261"

- Can occur after triggering a download. No negative side-effect seen so far. Can most likely be ignored and seems to be a Unity issue.

"You are trying to replace or create a Prefab from the instance '...' that references a missing script. This is not allowed. Please change the script or remove it from the GameObject."

- Can occur during creating preview images when a prefab has script dependencies. Can be ignored since Asset Inventory fixes this situation automatically but the error cannot be suppressed.

"NormalMap settings dialog comes up"

- Can occur during creating preview images when a texture is misconfigured in a prefab. Can be ignored since this is typically invisible in previews.

SUPPORT

Web: https://www.wetzold.com/tools

Discord: https://discord.gg/uzeHzEMM4B

Roadmap: https://trello.com/b/SOSDzX3s/asset-inventory