Notion Data: Usage

Standalone

If you want a standalone version of this module without the rest of the library. You can get it from the repo below:

https://github.com/CarterGames/NotionToUnity

Supported Properties

Any `string` convertible type should also support JSON for custom classes, but the mileage may vary. Best to just store raw data in these assets and convert the data with an override to the `PostDataDownloaded()` method in the `NotionDataAsset` .

Note that rollups are supported only when they show a property that is otherwise supported below:

Property type	Conversion types supported (Unity)
Title	`string`
Text	`string` `NotionDataWrapper(GameObject(Prefab)/Sprite/AudioClip)` , List/Array of `string` , `int` , `float` , `double` , `bool`
Number	`int` `float` `double` etc
Toggle	`bool`
Single-Select	`string` `enum`
Multi-Select	`string[]` `List<string>` `enum flags`
Rollup	`Any supported in this table.`
Date	`SerializableDateTime`

Notion setup

You need to make an integration in order for the downloading to work. You can make an integration here. The steps to follow are:

- Make a new integration with the `New integration` button.
- Select the workspace the integrations can access. This should be the workspace the database(s) you want to download are in.
- Give the integration a name & continue to the next page.
- Navigate to the `Capabilities` tab from the sidebar and ensure the integration has read access. You can disable the rest as you only need the readability.

- Navigate to the `Secrets` tab and copy the key for use in Unity.

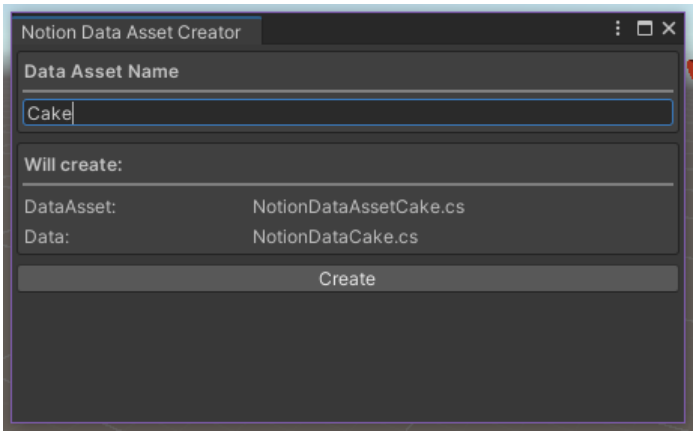
Once done you can then enter Notion and add the integration to one or multiple pages to allow the Notion API to access the data. This is done from: `... > Manage Connections > "Find and add your integration from the options"` If you don't see the integration you just made listed, close & open Notion and try again.

Unity setup

In Unity you use a `NotionDataAsset` to store the data. This is just a scriptable object which has a custom inspector to aid with the data download. Each instance you make will consist of the data asset, a scriptable object class and the data class which holds the data structure for the data asset to store. There is a tool to make these for you which can be found under `Tools > The Cart > Modules > Notion Data > Asset Creator`

Asset creator

The asset creator is an editor window that handles creating the classes for a `NotionDataAsset` . You just enter a name for the class you want to make and press `Create` when ready. You'll be able to see a preview of what the classes will be called before your press the `Create` button. Once pressed you'll be able to select where you want to save each newly generated class. Its best to keep them together in the same directory for ease of use.



Manual Way

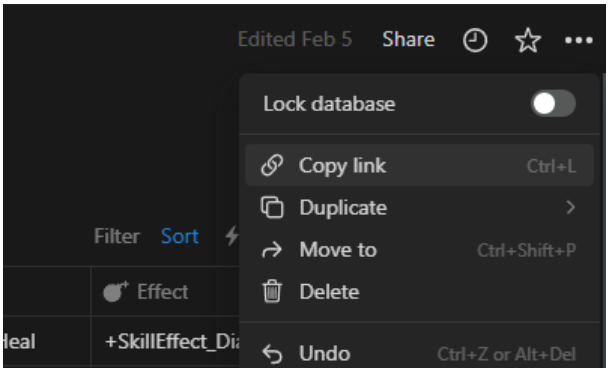
Alternatively, you can just make a new serializable that inherits from `NotionDataAsset` with a `[CreateAssetMenu]` attribute to let you make instances in the project.

Once setup you'll just need to write your data class to have the fields you want. An example of a Persona healing skill data class:

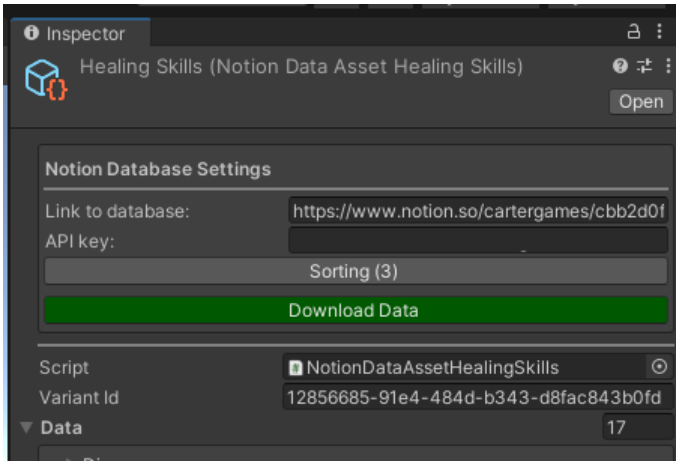
```
[Serializable]
public class DataHealingSkills
{
    /* ————————————————————————————————————————————————————————————————————————————————
    |   Fields
    ————————————————————————————————————————————————————————————————————————————————*/

    [SerializeField] protected string skillName;
    [SerializeField, TextArea] protected string desc;
    [SerializeField] protected NotionSpriteWrapper icon;
    [SerializeField] protected SkillType type;
    [SerializeField] protected ActionTarget target;
    [SerializeField] protected SkillCost cost;
    [SerializeField] protected NotionPrefabWrapper effect;
    [SerializeField] private float power;
    [SerializeField] private StatusAilment cureAilments;
    [SerializeField] private bool canRevive;
}
```

Downloading the data

To download your data you will need the link to the database page and the secret key for the intergration you made earlier. The Database link can be grabbed from the ... menu on the page the database is on:

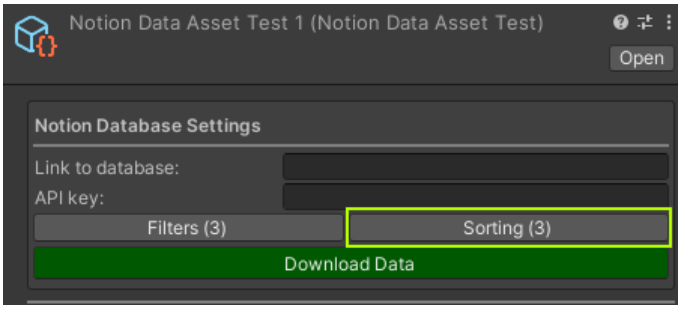


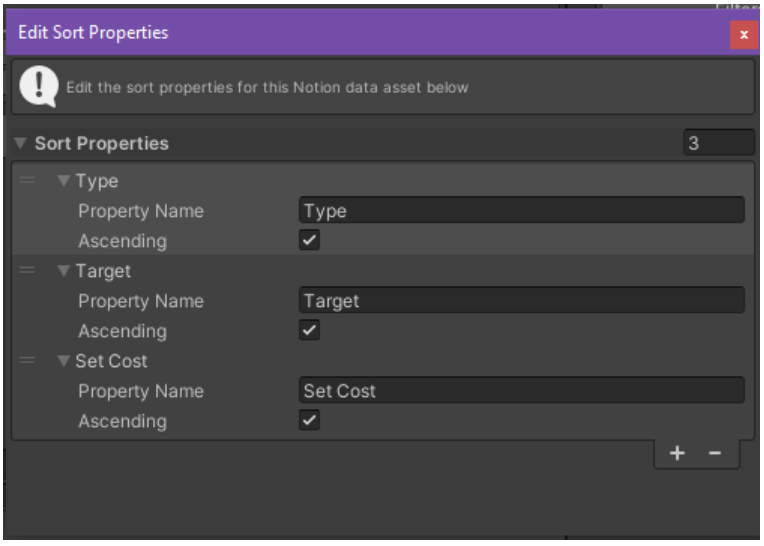
Then just fill the fields on the data asset (make one from the `CreateAssetMenu` if you haven't already) and then press the download button. If all goes well you'll see a dialogue stating so. If it fails you should see the error in the console.



Sorting Properties

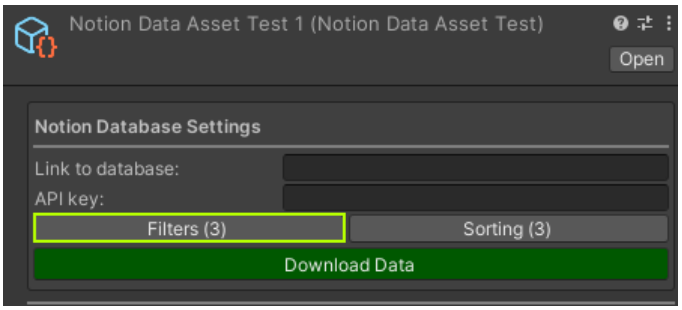
You can apply sorting properties to your download requests by adding them to the sort properties list in the inspector. Press the soring button to open the popup to edit them. The text for each entry is the Notion property name you want to sort by, with the tick box set to if you want to sort ascending for that property. The order of the sort properties in the list defines the order they are used, just like in Notion.



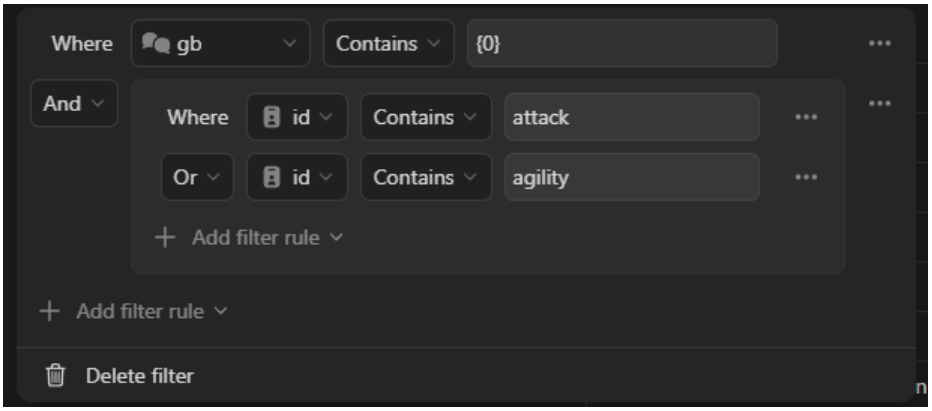


Filters

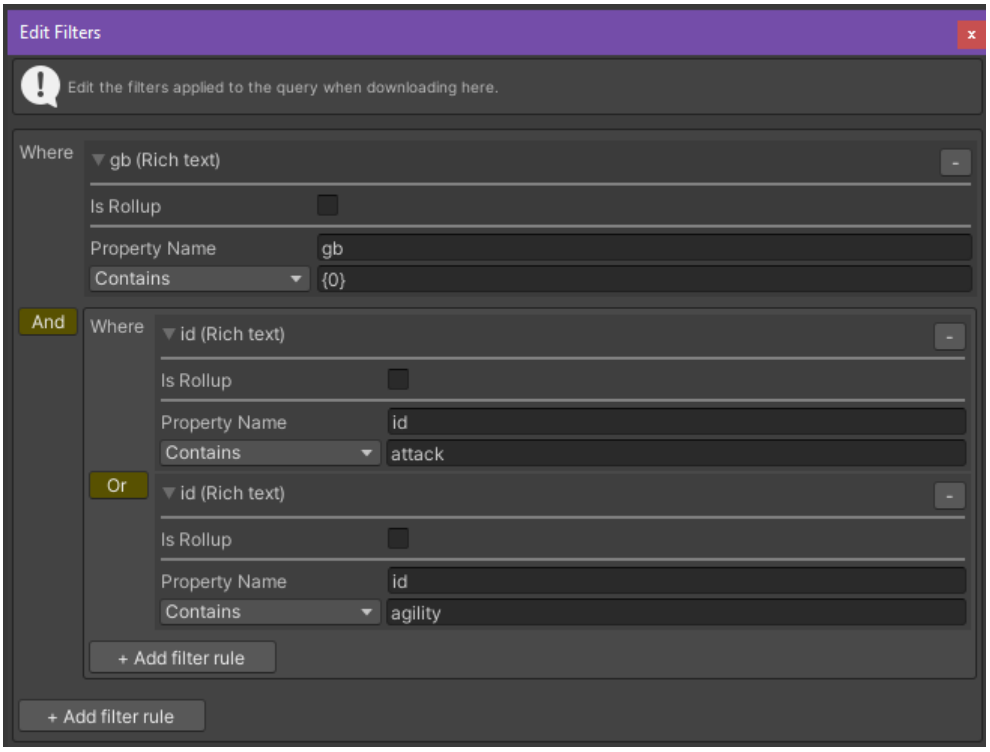
As of 0.4.x you can also apply filters to the download requests by using the filters window. This window more or less mimic's Notion's filters GUI. This setup supports the property types the system currently supports reading. The only notable difference is with rollup support. It is supported, but you'll have to use the type the rollup is displaying and then define it as a rollup of that type for it to work.



Notion Example



Unity Example



Wrapper classes

Some data needs a wrapper class to assign references. This is provided for GameObject prefabs, Sprites & AudioClips should you need it. They are assigned by the name of the asset when downloading the data.

Post Download Logic

You can also manipulate the data you download after receiving it by writing an override to the method called `PostDataDownloaded()` on the `NotionDataAsset` . Note if you need to run editor logic, make sure it is in a `#ifdef`. An example below:

```
#if UNITY_EDITOR
        protected override void PostDataDownloaded()
        {
            skillLookup = new SerializableDictionary<string, DataHealingSkills>();

            foreach (var data in Data)
            {
                // Runs a method called PostDownloadLogic() on the data class inst
                data.GetType().GetMethod("PostDownloadLogic", BindingFlags.NonPubl
                    ?.Invoke(data, null);

                // Adds the data class to a lookup for easier use at runtime.
                skillLookup.Add(data.Name, data);
```

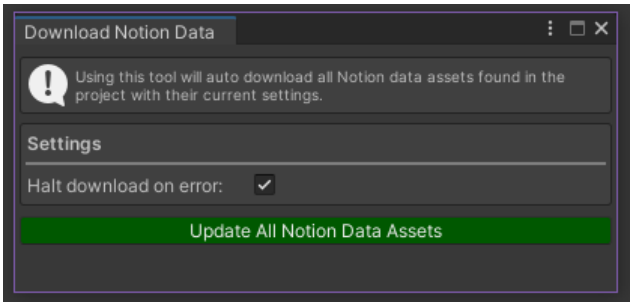
```
            }

            // Saves the changes to the scriptable object.
            UnityEditor.EditorUtility.SetDirty(this);
            UnityEditor.AssetDatabase.SaveAssets();
        }
#endif
```

Updating all assets

You can download all data assets in one process through an additional editor window. The window can be found under:

```
Tools/Carter Games/The Cart/Modules/Notion Data/Update Data
```

The window has the option to halt the downloading of assets if an error occurs, by default this true. To download all assets in the project, just press the download button and wait for the process to complete.



Scripting API Info

If you are using custom assembly definitions you will need to reference the runtime assembly from this asset in-order to access the API such as the `DataAccess` class. If you are not using custom assemblies, you should be able to access all the API by default. The runtime assembly is called `CarterGames.Standalone.NotionData.Runtime`

Accessing Notion data assets

You can reference the assets as you would a normal scriptable object in the inspector. Or you can use the `DataAccess` class in the project to get them via code. Each Notion Data Asset has a variant id in the inspector. By default the variant id is a new GUID on creation. You can change this to help you identify a single instance of assets of the same type as another. Some example usage below:

```
private void OnEnable()
{
    // Gets the first asset of the type found.
    var asset = DataAccess.GetAsset<NotionDataAssetLevels>();

    // Gets the asset of the matching variant id.
    asset = DataAccess.GetAsset<NotionDataAssetLevels>("MyAssetVariantId");

    // Gets all of the assets of the type found.
    var assets = DataAccess.GetAssets<NotionDataAssetLevels>();
}
```

Usage Example

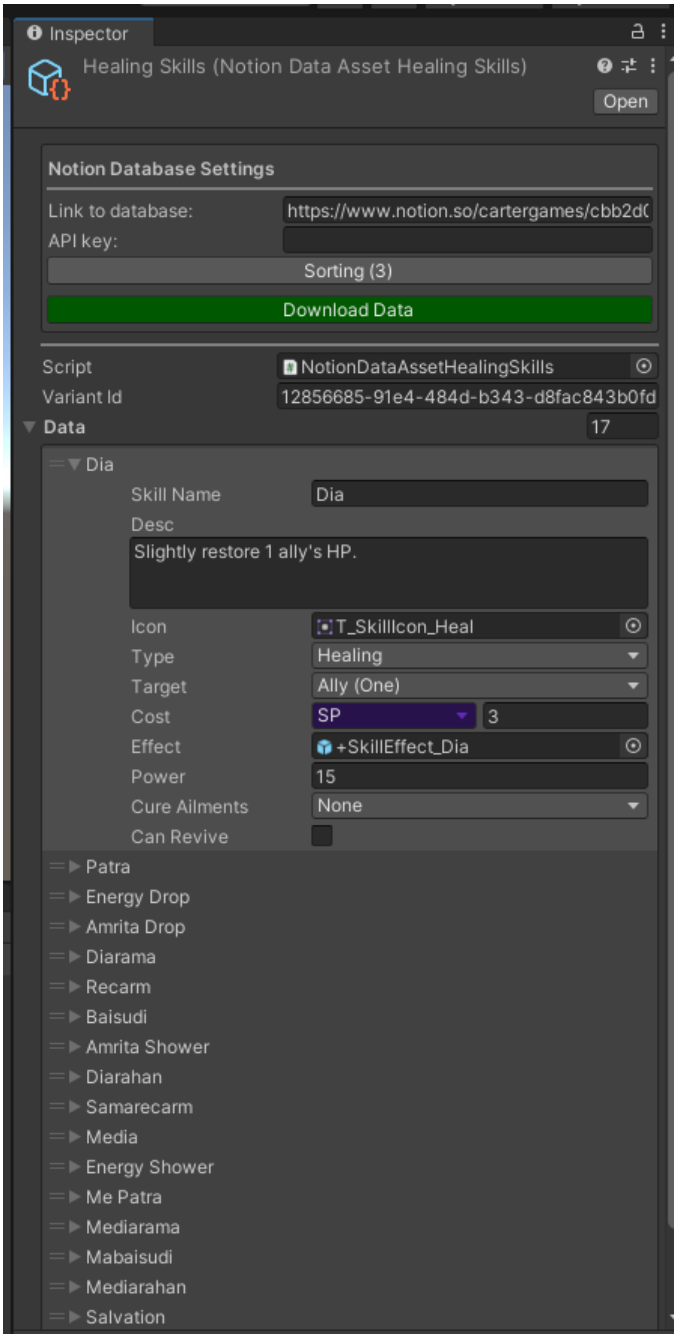
Below is an example using the system to store data for Persona 5 healing skills for persona's.

Notion

A database of all the skills for healing:

Unity

The downloaded data in Unity:


Inspector
Healing Skills (Notion Data Asset Healing Skills)
Open
Notion Database Settings
Link to database:
https://www.notion.so/cartergames/cbb2d(
API key:
Sorting (3)
Download Data
Script
NotionDataAssetHealingSkills
Variant Id
12856685-91e4-484d-b343-d8fac843b0fd
Data
17
Dia
Skill Name
Dia
Desc
Slightly restore 1 ally's HP.
Icon
T_SkillIcon_Heal
Type
Healing
Target
Ally (One)
Cost
SP
3
Effect
+SkillEffect_Dia
Power
15
Cure Ailments
None
Can Revive
Patra
Energy Drop
Amrita Drop
Diarama
Recarm
Baisudi
Amrita Shower
Diarahan
Samarecarm
Media
Energy Shower
Me Patra
Mediarama
Mabaisudi
Mediarahan
Salvation