ChatGPT

Creating Compendium Packs in Foundry VTT v13 (and Making Folders Appear)

Overview of Compendium Packs in Foundry VTT

Compendium Packs are packaged collections of game data (Actors, Items, Journals, etc.) that are stored outside a specific world. They allow sharing content between worlds and keep the game running efficiently by not loading everything at once [1] [2]. Each compendium pack holds only one type of document (except special "Adventure" packs) [2]. In Foundry VTT, compendium packs can belong to a *world*, a *module*, or a *game system*, and they appear in the **Compendium Packs** sidebar grouped by their source (world, module, or system) [3].

When you create a custom game system, you will typically define a set of compendium packs (for items, actors, etc.) in the system's manifest file. These packs will then show up in the compendium sidebar whenever that system is active. The compendium sidebar in newer Foundry versions (v10+ and above) also supports **folder organization** for compendium packs to help categorize them in the UI [4] [5].

Defining a Compendium Pack in the Manifest

For a **custom system** (or a module), the first step is to declare the compendium pack in your manifest (`system.json` for a system, or `module.json` for a module). In the manifest, there is a `"packs"` field which is an array of compendium pack definitions. Each compendium pack is defined by an object with specific properties, for example:

```
"packs": [
  {
    "name": "my-pack-name",
    "label": "My Pack Display Name",
    "system": "my-system-id",
    "path": "packs/my-pack-name",
    "type": "Item",
    "ownership": { ... },
    "flags": { ... }
  }
]
```

Let's break down the important fields for each pack definition:

- **name** – A machine-readable identifier for the pack. This serves as the pack's ID (and is used in the code, e.g. `game.packs.get("packageId.packName")`).

- **label** – The human-readable name that will be displayed in the Compendium sidebar. (Note: this label is **not** run through localization, so it should be exactly how you want it shown) [6].
- **type** – The document type that this compendium contains (e.g. `"Actor"`, `"Item"`, `"JournalEntry"`, `"Scene"`, etc.). Each compendium may only contain one type of document [2].
- **system** – *(Required for Actor or Item packs as of Foundry V10+)* The game system identifier that this pack is tied to [7]. In a **system** manifest, this should be your system's ID (e.g. `"my-system-id"`). This field ensures the compendium can only be used with the specified system.
- **path** – The relative file path where the compendium's data will be stored. In Foundry V13 (which uses LevelDB for storage), this typically points to a folder (or file prefix) under your system's `packs/` directory. For example `"packs/my-pack-name"` (Foundry will handle creating the actual database files in that location) [8]. You generally **do not need to manually create or edit** the contents of this path if you use Foundry's tools to populate the compendium [8].
- **ownership** – *(Optional)* An object defining default permission levels for each user role (PLAYER, TRUSTED, etc.) [9]. If omitted, the default is that only GMs (and Assistant GMs) can see and edit the compendium. You can set this to allow players to view a compendium if desired (e.g. give players Observer permission) [10].
- **flags** – *(Optional)* Usually an empty object here, unless you need specific flags.

Important: Make sure the `name` of each pack is unique (and does not conflict with packs from other systems or modules). The combination of package ID and pack name (`systemid.packname`) must be unique for the lookup. Also, double-check that if your pack `type` is `"Item"` or `"Actor"`, you have included the `system` field correctly; omitting it can cause the pack not to load properly in Foundry v10+ [7].

Once you have listed your pack(s) in the manifest, start up Foundry and launch a world with your system. If everything is configured correctly, you should see the compendium pack appear in the **Compendium Packs** sidebar (with the label you provided). At this stage, the pack will be **empty** unless you have already added data to it, which we'll cover next.

Populating the Compendium Pack with Data

Declaring the pack in the manifest only creates an empty compendium container. The next step is to populate it with your actual game data (items, actors, etc.). There are a few ways to create or import compendium content:

- **Manual Creation via Foundry UI:** You can create a new compendium in the sidebar (or use the one from your system) and then manually **import** documents into it. For example, you could create items in a temporary world and drag them into the compendium pack. However, doing this for a large number of items can be tedious. Foundry does allow exporting entire folders of documents to a compendium (via *right-click* > *Export to Compendium* on a folder in the Actors/Items directory), which can speed up the process if your data is already set up in a world. *(Note: Make sure the compendium is unlocked if you plan to export into it; compendiums from systems are typically locked by default to prevent changes)* [11] [12].

- **Using JSON Source Files and the Foundry CLI:** A common practice for system developers is to maintain the compendium data in human-readable files (like JSON or YAML) and then use a tool to pack that data into Foundry's database format. **Foundry VTT v11+ uses a LevelDB-based database** for compendiums (and no longer uses the older NeDB `.db` flat files). Foundry **will not read raw `.json` files** placed in the `packs/` folder – the data needs to be in the proper database files for the compendium [13]. Fortunately, Foundry provides a command-line tool (Foundry CLI) to help with this. The workflow typically looks like: keep your source data as JSON (for version control), then run the CLI command to "pack" those JSON files into the actual compendium database before distributing or using the system [13]. For example, the CLI has a command like `fvtt package pack -n "my-pack-name" --in src/packs/my-pack-name-json --out packs` which will create the LevelDB data in your `packs/` directory for that pack. (You would run this as part of a build or release process.) This approach was recommended by Foundry developers for custom systems [14] [15].

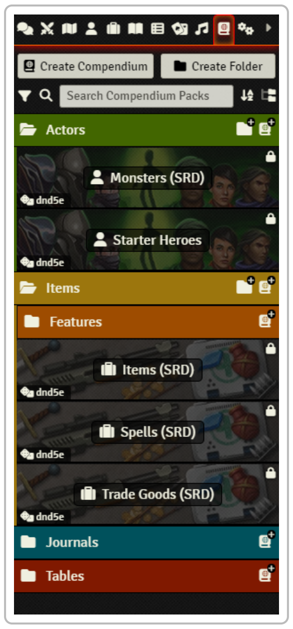
- **Using Foundry's built-in conversion (legacy .db files):** In earlier versions (v10 and below), compendium packs were stored as `.db` files (NeDB format, which is basically a JSON lines database). If you see references to `.db` files (for example, `dnd5e` system has `packs/items.db`), note that in v11+ Foundry will **auto-migrate** those to LevelDB on launch. If you provide a `.db` file and no LevelDB folder, Foundry v11+ will convert it on first launch and then ignore the `.db` afterward [16]. This is a fallback, but using the CLI or a proper LevelDB export is cleaner for new development.

In your case, you mentioned using a script to build JSON directly into the database. If this means you have written directly to the LevelDB files or using some library to insert entries, be cautious that the data is correctly formatted. Usually, it's easier to use the Foundry CLI or let Foundry handle creation by importing from JSON via the UI. Nonetheless, the key point is that after populating the compendium, you should verify that the data appears when you open the compendium in Foundry. Each entry in the compendium should show up with its name, type, etc., as expected.

Troubleshooting Tip: If the compendium pack shows up in the sidebar but appears empty when opened (even though you think you added data), it could mean the data wasn't imported correctly. Ensure that the data file is in the right location and format. For LevelDB, the compendium data isn't just a single file – it's a set of files in a folder. Using the CLI to pack ensures everything is where it needs to be. If you manually edited or created the DB, double-check with a small test (e.g., create one item in Foundry, export it to a compendium, then inspect the files to see how it's represented, and compare to your method). Also, ensure the compendium pack's `type` in the manifest matches the type of documents you added – if you defined the pack as type "Item" but tried to put Actor data in it, those entries wouldn't show up.

Organizing Compendium Packs into Folders (Using `packFolders`)

By default, all compendium packs for a system or module will just list individually under the **Compendium Packs** sidebar. Foundry v10+ introduced the ability to define **Compendium Folders** to group packs into collapsible categories in the sidebar [4]. This is purely an organizational UI feature – it does **not** affect the data inside the packs, only how packs are grouped for the user. If you want your compendiums to be neatly organized under certain headings (folders), you need to use the `packFolders` section in your manifest.



Example of compendium packs organized into folders in the Compendium sidebar. Here, packs are grouped under folders like "Actors", "Items", and a colored "Features" folder (with sub-packs inside). This improves navigation when a system has many compendium packs.

To set up compendium folders, add a `"packFolders"` array in your manifest (at the same level as `"packs"`). Each entry in `packFolders` defines a folder with the following properties [17] [18] :

- **name** – The name of the folder as it will appear in the sidebar.
- **sorting** – Either `"m"` (manual) or `"a"` (alphabetical). This determines how packs inside the folder are sorted. `"m"` means you can manually reorder packs (or subfolders) in that folder by dragging, whereas `"a"` will automatically sort them by name [19] .
- **color** – *(Optional)* A hex color code (e.g., `"#ff0000"`) for the folder's header background in the UI. If not provided, a default color is used.
- **packs** – An array of pack `name` values that should be placed inside this folder [20] . These should correspond exactly to the `"name"` fields of the packs defined in your packs array. (Using our example above, if `"name": "my-pack-name"`, you would put `"my-pack-name"` in this list to include that pack in the folder.)
- **folders** – *(Optional)* An array of sub-folder definitions, following the same structure, if you want to nest folders inside this folder. Foundry allows nesting compendium folders up to 4 levels deep [21] . Subfolders also have their own name, color, sorting, etc., and would contain packs or even more subfolders. (Nesting is only needed if you have a lot of packs and want multi-level categorization. Many systems just use one level of folders.)

Example: Suppose your system has two compendium packs, one for Monsters and one for Heroes (both are Actors). You want them grouped under a folder called "Actors" in the sidebar. You could set up:

```
"packs": [
  { "name": "monsters", "label": "Monsters", "system": "my-system-id", "path":
"packs/monsters", "type": "Actor" },
  { "name": "heroes",   "label": "Starter Heroes", "system": "my-system-id",
"path": "packs/heroes",   "type": "Actor" }
],
"packFolders": [
  {
    "name": "Actors",
    "sorting": "a",
    "packs": ["monsters", "heroes"],
    "color": "#3a803d"
  }
]
```

In this example, when a world using your system is loaded, Foundry will create a folder **Actors** (with a green color) in the compendium directory, and inside that folder will be the *Monsters* and *Starter Heroes* compendium packs. The packs will be sorted alphabetically (`"sorting": "a"`), which in this case means "Monsters" then "Starter Heroes" (or whatever the names are). If you wanted manual ordering or further sub-grouping, you could adjust those settings or add nested folders.

The `packFolders` feature is how you get those folder headers to show up in the compendium sidebar **by default**. You mentioned that you “cannot get the folders to show up” – this likely means the packs are all appearing at the root of the compendium list rather than under the intended folder headings. Common reasons for this include: not configuring `packFolders` at all, a typo or mismatch in the `packFolders` configuration, or caching issues. We’ll address each:

- **Missing or incorrect configuration:** Double-check that your `system.json` (or `module.json`) actually contains the `packFolders` array with the correct structure. Ensure that the `packs` listed under each folder exactly match the pack names. Remember that the pack name is case-sensitive and should exclude any `.db` extension (just use the raw name you put in the `name` field of the pack definition). Also verify that the JSON syntax is correct (commas, brackets, etc.), since a JSON error in the manifest could prevent Foundry from reading any of the packs/folders definitions.

- **Changes not applied to an existing world:** Foundry only sets up the compendium folder structure **when a world is loaded for the first time with that package** (system or module). Once a world has stored the compendium configuration, it won’t automatically update it on its own if you later change `packFolders`. In other words, if you added `packFolders` after already using your system in a world, that world might still be using the old compendium layout (with no folders). According to the documentation, *“packFolders are only loaded once per world (per package); if you want to see how updates to the module/system property are working, you must run* `game.settings.set('core', 'compendiumConfiguration', {});` *and refresh your page twice”* [22] . For a system, a simpler method is to **create a new temporary world** after adding/changing `packFolders` – the new world will load the updated folder layout fresh. If you need to update an existing world’s compendium folders, you can use the console command provided (which resets the compendium folder configuration to default, allowing your new settings to take effect) [23] .

- **Foundry version or known issues:** Ensure you're testing on Foundry V13 (as specified) and that there isn't a bug affecting compendium folders. There was a past issue where packs were not placed in folders as specified, but generally the feature is stable now [24]. If you suspect a bug, check Foundry's release notes or community forums. But most often, it's a configuration mistake or a caching issue as described above.

Best Practices and Summary

Correct Process to Create a Compendium Pack: To summarize the proper workflow for creating compendium packs (especially in a custom system for Foundry VTT 13):

1. **Define packs in the manifest:** In your `system.json`, add entries under `"packs"` for each compendium you need (actors, items, etc.), including the pack name, label, type, system, and path [6]. Double-check requirements like the `system` field for Item/Actor packs (must match your system ID) [7]. This step makes the compendium visible in the sidebar [25].
2. **(Optional) Define folder organization:** If you want those packs grouped under folder headings in the sidebar, configure the `"packFolders"` in the manifest. Set up the folder name, sorting mode, color (optional), and list which pack names go into each folder (and any subfolders) [19] [5]. This will create the folder structure on first load of a world with your system. Remember that changes here won't retroactively apply to worlds already created without resetting their compendium config [23].
3. **Prepare the pack content:** Create or import the actual data into the compendium. If using Foundry's UI, you might enter items in a world and drag-drop or export them into the compendium pack. If you have data in JSON files, use the Foundry CLI or another method to pack that data into the compendium's database format [13]. For Foundry V13, ensure the data ends up in the LevelDB files under your `packs/` directory (the CLI tool handles this conversion for you). Simply placing JSON files in the folder will **not** populate the compendium at runtime [13].
4. **Verify in Foundry:** Launch Foundry and open a world with your system. In the Compendium Packs sidebar, you should see your compendium pack(s). If you set up folders, you should see the folder names and the packs inside them, instead of a flat list. Open the compendium and verify the entries are present. If the pack is empty or missing data, re-check the packing step. If the folder grouping isn't showing, re-check the `packFolders` configuration and remember that it might require a fresh world or resetting compendium config to take effect [23].
5. **Lock or Release:** By default, system compendium packs are locked (not editable in-game). That's usually fine (you want them read-only unless a GM duplicates them), but know that as a developer you can unlock a compendium to add/remove entries during development. Just be careful: if you modify a system's compendium within a world and then update the system or re-pack data, those changes may get overwritten. It's best to treat the compendium as built from your source files.

Finally, note that **compendium folder organization is a UI convenience**. If by "folders" you were hoping to have actual folders *inside* a single compendium (i.e., grouping the compendium's contents), that is not currently a feature of Foundry's core – compendium contents are always a flat list of documents (there is a community request to allow folders inside compendiums, but as of v13 it's not implemented in core) [26]. The `packFolders` approach discussed above refers to organizing packs in the **sidebar** only.

By following the above steps and guidelines, you should be able to create compendium packs for your custom system and have them appear in Foundry VTT, with the desired folder categories visible in the

Compendium sidebar. This ensures your content is packaged correctly and neatly organized for users of your system. Good luck with your Foundry VTT system development!

Sources: The information above was gathered from the Foundry VTT official documentation and community knowledge bases, including the Foundry VTT Knowledge Base on compendium packs and module development (for manifest structure) [27] [28], the *Foundry VTT Community Wiki* (which explains `packs` and `packFolders` in manifests) [29] [17], as well as community discussions on best practices for packing compendium data in Foundry v11+ [14].

---

[1] [2] [3] [11] [12] Compendium Packs | Foundry Virtual Tabletop
https://foundryvtt.com/article/compendium/

[4] [5] [9] [10] [19] [20] Introduction to Module Development | Foundry Virtual Tabletop
https://foundryvtt.com/article/module-development/

[6] [7] [8] [17] [18] [21] [22] [27] [28] [29] Compendium Collection | Foundry VTT Community Wiki
https://foundryvtt.wiki/en/development/api/CompendiumCollection

[13] [14] [15] [16] Advice for compendium data in a custom system - League of Extraordinary FoundryVTT Developers
https://www.answeroverflow.com/m/1176581388238077981

[23] [25] 03. system.json | Foundry VTT Community Wiki
https://foundryvtt.wiki/en/development/guides/SD-tutorial/SD03-systemjson

[24] Compendium Packs not placed inside specified pack folders #9437
https://github.com/foundryvtt/foundryvtt/issues/9437

[26] [Epic] Allow folder organization inside of Compendium packs #1109
https://github.com/foundryvtt/foundryvtt/issues/1109