Details

As mentioned before, TpTileBundle (Bundle) assets are used to archive all or a section of a Tilemap. TpTileFab (TileFab) assets combine references to several Bundles, creating an archive of one or more Tilemaps; a multiple-Tilemap prefab of a sort.

In Bundles, cloned tiles in the scene are changed to Locked tiles and are stored as sub-objects of the Bundle asset. In other words, the cloned tiles are changed to Project folder assets and stored as part of the Bundle asset.

When normal Unity tiles are archived, just their transform, color, and flags settings are preserved, along with a reference to the tile asset in the project folder.

Prefabs which are children of the selected Tilemaps are archived by reference only: no asset copying occurs.

Note that Prefabs can be bundled from a Scene hierarchy or from a Project folder. This implies differences in how the transform information is archived:

- When bundling Prefabs from a scene, the stored rotation and scale are the rotation and localScale of the root GameObject of the Prefab in the scene hierarchy.
- When bundling Prefabs from a project folder, the stored rotation and scale are the rotation and lossyScale of the root GameObject of the Prefab in the Project folder.

Just to be clear about naming: Tilemap Prefab (with upper-case P) refers to a created prefab encompassing one grid with one or more child Tilemaps.

It’s encouraged to use a different folder each time you create Bundles, TileFabs, or Tilemap Prefabs, although this isn’t enforced.

Associated Components

You can use the TpBundleLoader component to load a single Bundle to a Tilemap. Place the component on any compatible (same layout, etc.) Tilemap’s GameObject and drag in the Bundle asset reference. Switch to Play mode and loading will happen automatically. Or you can click the Load button if you wish to test or perhaps restore a Tilemap.

If you maintain references to several Archive assets in your Scene (to ensure availability in a build) then you can change the asset reference in TilePlusLoader and call the Load method of the component.

Similarly, you can use the TpFabLoader component to load a TileFab.

Here are two other ways to use TileFabs, plus there's a lot more about them as you read on in this section.

- Paint a TileFab (or a single Bundle) using Tile+Painter. This is discussed in the Painter documentation.
- Load all or some of the Bundles in a TileFab on to different Tilemaps in your scene at runtime.

The second use listed above is particularly useful. Coders can use methods in TileFabLib to load one or more Tilemap sections dynamically from Bundles and TileFabs. For example, the TpAnimZoneLoader tile can be used to specify TileFabs to load when an entity passes into or out of a trigger zone. The TileFabLib static library has comprehensive methods to load TileFabs and Bundles, and enables higher-level elements such as ZoneManagers and ZoneLayout, the key elements of the Layout (Chunking) system.

The Bundling Process

When you use the Bundle Tilemaps menu command, the bundling process asks you some questions, as outlined in the User Guide. Using this information, the bundler creates the TpTileBundle (Archive) assets: one for each Tilemap. Then creates copies of all the TPT tiles and adds them to the asset, saves any asset references to Unity tiles, and saves all the information required to rebuild a Tilemap, including position, transform, color, and flags.

Since the values for transform, color, and flags are often the same for large numbers of tiles (especially so for Unity tiles), these are stored in indexed look-up tables.

After all the Bundle assets are created, a TileFab is created in the same folder. References to the Bundle assets are stored in this new asset.

If you're making a Tilemap Prefab of a Grid and its child Tilemaps, the bundler creates a new prefab with the Grid and the child Tilemaps. These child Tilemaps are empty. The parent Grid of the Tilemaps has the TpPrefabMarker component added with a reference to the created TileFab. The TpPrefabMarker component loads up the empty Tilemaps when the Prefab is dragged into the Scene or otherwise loaded.

If there are any prefabs as children of the Grid or Tilemap GameObjects then the Project folder references to these prefabs are added to the TpTileBundle.

To reiterate:

- When bundling Prefabs from a scene, the stored rotation and scale are the rotation and localScale of the root GameObject of the Prefab in the scene hierarchy.

- When bundling Prefabs from a project folder, the stored rotation and scale are the rotation and lossyScale of the root GameObject of the Prefab in the Project folder.

The completed Tilemap Prefab is placed in the folder that you selected earlier.

If you're archiving a single Tilemap into a Bundle, it's mostly the same process except that a Tilemap Prefab isn't created.

Tilemap Prefabs can be used just like any other prefab with one exception: it's "Locked," and you can't edit the Tilemaps. In Editor sessions, the system will try to stop you opening or making any changes to a Tilemap Prefab.

Why? A Tilemap Prefab contains copies of the Tilemaps, with no tiles. When the Prefab is placed in the Scene, TpPrefabMarker loads the tiles and prefabs from the TileFabs and Bundles. If you modify the Prefab in the Unity Editor by painting a TilePlus tile, then save prefab overrides, the new tile (being a scene asset) can't be saved in the Tilemap Prefab for reasons described earlier.

One could use the Allow Prefab Editing configuration option and paint normal Unity tiles or add GameObjects. That will preserve these changes in the Tilemap Prefab.

However, it's better to unpack the prefab, modify it, and generate a new Tilemap Prefab for maximum compatibility. You still have the original Grid and/or Tilemaps; unlike normal Prefab creation via Drag and Drop, the source isn't linked to the created prefab. If you ever need to recreate the original for editing, drag the Prefab into a scene, use the Unity menu command "Prefab/Unpack Completely."

Why does the Bundle Tilemaps command insist on a parent Grid to make a Prefab? If you make a prefab out of a Tilemap without a parent Grid, then instantiating it won't work properly unless you parent the instance to a Grid (this has nothing to do with TPT tiles). It will look fine in the mock scene created when you edit a prefab, because Unity adds the parent Grid for you in the editing Stage scene.

The created Archive asset contains all the converted, Locked tile assets. A name for the asset is created from the base name that you provide in the dialog box and/or from the scene and Tilemap names.

This is just for convenience so that you can have an idea of where the asset was created from.

If you inspect the asset, you'll notice that it has a few fields:

- Time Stamp: The creation time, in UTC time.
- Scene Path: The Scene path with dot notation.
- Original Scene: The name of the scene that the asset was created from.
- A flag that informs whether this Bundle was created from a Grid Selection.
- A flag that controls whether Tile+Painter includes this asset in its painting sources list.
- A User flag (a boolean) and a user string. Optional use.
- An Icon reference. Used to display an Icon for the asset in Tile+Painter.

- A GUID. This is used in the Layout/Chunking system.
- Various asset lists.

The Time Stamp, Scene Path, and Original Scene aren’t used in this distribution (but are used in the project that TilePlus was developed for).

If you know that you’ve edited a locked Tilemap, then to ensure that there are no issues, use the Tools/TilePlus/Prefabs/Unlocked Tiles test after selecting a single Tilemap. If there are clone tiles in a Prefab, you’ll have issues.

What’s the difference?

- A Tilemap Prefab instantiates Tilemaps and loads all the tiles from Tilefabs.
- A TileFab’s tiles are loaded on to an existing Tilemap and never creates new Tilemaps.

It’s a big difference! When a Tilemap Prefab is instantiated, a new set of Tilemaps is created and parented to a Grid. Therefore, if you instantiate 10 of these you have 10 independent sets of Grids and Tilemaps.

TileFabs require a set of compatible Tilemaps to exist in advance. So, if you have a scene with some Tilemaps and load a TileFab, the tiles load onto the existing Tilemaps. If you load the same TileFab 10 times the same tile would be written 10 times to the same locations.

The power of TileFabs comes when you consider that they can be painted anywhere on the Tilemaps. If you offset the placement of each of the 10 TileFab loads, they can be used to fill-in an area of a Tilemap. That can’t be done with Prefabs. A bonus is that Bundles and TileFabs can be painted with Tile+Painter.

Tilemap Prefabs can be useful as a quick way to load part of a scene. But you can also do that with TileFabs, and they’re way more flexible as they can be edited whilst being loaded.

Revision #1
Created 2025-07-08 10:57:05 UTC by Vonchor
Updated 2025-07-08 11:16:10 UTC by Vonchor