

# Infrastructure

## Review

Bundle: an archive of all the tiles and Prefabs for a single Tilemap. TileFab: references a group of Bundles.

Let's review what type of data are in a Bundle asset:

- A List of TilePlus Tiles.
- A List of Unity Tiles.
- A List of Prefabs.
- Indexed Lists of Tile Flags, transforms, and Colors.
- The BoundsInt for the group of tiles as calculated for TileFabs or Chunks.

It's not difficult at all to take all this information and populate a Tilemap with the tiles in the Bundle. Even a caveman could do it! There's a method in the Bundle asset called `TileSet` which unpacks the various Lists in the asset and returns a List of data items, each with the tile reference and Flags, transforms, and Color for each tile. Methods in `TileFabLib` and `TpZoneManager` provide higher-level APIs for both TileFab and Chunk use.

## Caching

Bundle assets cache all non-TilePlus tiles the first time that the `TileSet` method is used. This increases performance when the same Bundle is used repeatedly and most of the tiles are 'normal' Unity tiles (basically, anything that isn't a TilePlus tile). There is a method that clears the cache if you need to.

## Chunkifying

Bundle assets can also subdivide themselves into square chunks of arbitrary size. For example, a 1024 x 1024 Bundle can be divided into smaller chunks ranging from 4 x 4 (65K smaller chunks) to 256 x 256 (4 smaller chunks). After this operation, the Bundle turns this information into data ready to be loaded to Tilemaps using their block move methods. This cache can also be released when no longer needed.

This is the method used by the Layout system, and provides a way to set the loading chunk size at runtime. In other words, you can optimize the system at runtime based on the capabilities of the target system.

# Prefab Caching

Prefabs are never cached since they must be instantiated each time they are placed.

**HOWEVER** if a prefab has a `TpSpawnLink` component (or a derived component) then the `Spawner Service` is used. If the same prefab is used again, it's pooled and *not* repeatedly instantiated.

# Software Components

Support for `TileFabs` comprises `TileFabLib`, `TpZoneManager`, and `TpZoneLayout`. Each of the three components present successively higher-level APIs for using `TileFabs`.

`TileFabLib` is a static library which provides the basic functions for loading `TileFabs` and `Bundles` to `Tilemaps` in a scene, during Editor sessions and at runtime. For example, if you'd created a `TileFab` from a complex scene with 10 layers of `Tilemaps` comprising thousands of tiles, you can paint that `TileFab` to a `Scene` using `Tile+Painter`, or load it to a `Scene` in a running app. It's also used to create `TpZoneManager` instances at runtime.

`TpZoneManager` is a `Chunk Manager`. Instances of `TpZoneManager` `Scriptable Objects` are created at runtime. It has an API for chunk management, and loading/unloading chunks.

`TpZoneLayout` is a `MonoBehaviour` component that queries a single `TpZoneManager` instance: it's a base-class for loading and unloading chunks as they move in and out of the `Camera` range. This is a basic implementation, and `TpZoneLayout` can be subclassed or rewritten by you to work differently if you want.

See the [Side-Scroll layout demo](#) for an example of how to use the `ZoneLayout` component directly: it's pretty easy.

At a higher level, `ChunkedZoneSelectors`, `TpSceneManager`, and `TpSceneList` assets can be used to organize `TileFabs` into `TileScenes` which can be loaded and unloaded in their entirety without loading or additively loading a `Unity scene`.

---

Revision #4

Created 8 July 2025 11:45:16 by Vonchor

Updated 22 July 2025 17:39:10 by Vonchor