Why Use New GUIDs?

If you want to load the same TileFab multiple times programmatically you need to set the newGuids option true when using LoadTilefab. This is because a TilePlus Tile’s GUID is used in TpLib, and unsurprisingly, GUIDs are expected to be unique.

Recall: if the same TileFab (or Bundle) is used repeatedly the TilePlus tile’s GUID will also be reused, and that tile is ignored by the TilePlus system.

Why does this matter? It might not. If you’re not using TilePlus tiles in a particular TileFab then it probably doesn’t. But if you want to use TilePlus features like sending messages to TPT tiles or dealing with TilePlus events, or finding a tile by tag, Type, GUID, or interface, you’ll find that these unregistered tiles can’t be located.

If newGuids is set true when using LoadTilefab, all the TilePlus tiles have their GUIDs replaced with new ones. This means that they will register properly. Note that true is the default value for this parameter. Ordinary tiles are not affected by this option at all.

Note that the NewGuids feature isn't used by the Layout system since it does not repeat the use of any TileFabs. However, if you're not using the Layout system and are repeatedly loading the same TileFab to many places on one or more Tilemaps, it does matter only if these TileFabs include TilePlus Tiles.

TANSTAFL Dep’t

`(There Ain’t No Such Thing as A Free Lunch, originally coined by noted SF author Larry Niven)`

There’s another issue, and it’s not strictly a TilePlus issue, but more of a design issue regarding data persistence in saved game information.

Let’s say that your game has some TileFabs containing Waypoints, like what’s seen in TopDownDemo. This isn’t about the chunking system where chunks are added and deleted frequently. Your game loads chunks of gameplay tiles, and some of the tiles are TPT tiles with data that you might want to save, e.g., the most recent Waypoint.

As the game progresses and the Player moves around, new sections are loaded as needed. When the Player moves over a Waypoint then that Waypoint is enabled (perhaps it changes appearance) and a game save is created: it’s reasonable to persist the GUID of the Waypoint TPT tile.

The next time you run the game, it looks in its save data for the GUID of the most recent Waypoint and tries to enable it. But what would happen if the Waypoint was in one of the dynamically loaded

TileFab sections? And how do you know what TileFabs to load and where to place them?

The next time the game is played, how do you restore the sections already loaded up till the most recently used waypoint and then place the Player at that proper Waypoint? You can't preserve the GUID of the waypoint in a loaded section since that GUID changes each time that the TileFab is loaded.

This leads us to a great use for the ZoneRegistrations accumulated in a ZM: they contain all the information that you need to save to reconstruct the game world's TileFabs and remap GUIDs correctly.

When you load a TileFab using LoadTileFab and provide the ZM instance as one of the parameters, the ZM is sent the results of the load. It adds this information to a "breadcrumbs" list in the form of ZoneReg instances. The breadcrumbs are therefore a list of information about which TileFab assets were loaded and where they were placed.

This list has instances of serializable type ZoneReg, and each instance contains an index, the asset GUID, and the offset and rotation parameters which were used when loading. All the ZoneReg instances, in load order, can be retrieved using the GetAllZoneRegistrations property. The last N Registrations can be obtained with GetLastRegistrations.

However, there's no one way for these to be used since each game's requirements are different. However, each ZoneManager instance has other two useful methods that you can build on or use as an example:

- GetZoneRegJson creates a JSON string with all the serialized ZoneReg instances that you can save.
- RestoreFromZoneRegJson takes that exact JSON string and recreates all the TileFabs specified within.

The serialized ZoneReg instances are in ascending order of creation.

Get/Restore works for the simple (but common) use case of wanting to restore everything. It's possible to only save some of the ZoneRegs, but that's not handled in any built-in way.

A good starting point to develop a different approach is to use GetAllZoneRegistrations and process it yourself.

The Remapping

RestoreFromZoneRegJson also handles remapping GUIDs, using TpZoneManagerUtils.UpdateGuidLookup. That code scans the asset registrations and creates a lookup table mapping the GUIDs from the TilePlus tiles in the loaded ZoneReg entries with those in the tiles placed from the Bundles. In other words, create a mapping from the GUIDs you may have

saved as part of the game state to the new GUIDs that were created when the TileFab was loaded during the execution of RestoreFromZoneRegJson.

The method TpLib.GetTilePlusBaseFromGuidString will try to use the lookup table if it can’t find a match in the primary lookup contained in the TpLib static class. That solves the changed GUID issue. So that saved Waypoint GUID ‘points’ to the proper Waypoint if it was in a dynamically loaded section of the Tilemap.

If you’re ‘rolling your own’ there’s a property in TpLib which allows you to provide your own Guid-to-TilePlusBase mapping. TpZoneMangagerUtils.UpdateGuidLookup can be used to create the mapping. It also creates a reverse mapping, which is needed when Zones are deleted.

It should be noted that the TileFab and Bundle assets must be part of the build. For example, when a TpAnimZoneLoader tile is part of a scene, the asset reference in that tile causes the TileFab, referenced Bundles, and other associated assets such as textures for the tile sprites, to be included in the build.

However, if you’re loading everything dynamically you need to ensure that the assets you need are available in the build. If you’re reading this far you probably know how to do that, but you could place them in a Resources folder or reference them all somehow in a Monobehaviour component attached to some GameObject in at least one scene.

Finally, if you are using the TPT persistence scheme, do not try to Restore to TPT tiles prior to remapping GUIDs. This is important since the restore process depends on the GUIDs being mapped correctly.

Revision #2
Created 2025-07-08 14:49:10 UTC by Vonchor
Updated 2025-07-08 14:55:21 UTC by Vonchor