# Selectors

Selectors are used by the layout system to determine what and how to load. Each Selector has a reference to one or more TileFabs (but usually just one) and a LoadFlags field.

There are two types provided with this distribution: `SingleFabChunkSelector` and `ChunkZoneSelector`.

`SingleFabChunkSelector` is very simple and just causes the same TileFab to be painted to each Zone. This Selector does not accommodate variable Chunk Size: Chunk Size is the size of the TileFab referenced by the SingleFabChunkSelector.

`ChunkZoneSelector` is almost the same, however, its `Chunkify` method causes all the child Bundles of the referenced TileFab to subdivide themselves into the specified chunk size.

When Bundles “Chunkify” themselves, they ready their tiles for Tilemap loading using SetTiles(TileChangeDataArray). The tiles from the Bundle asset are reorganized into groups of tiles for each Chunk. For example, if a Bundle has a size of 512 x 512 and the Chunk Size is for the TScene is 128 then the Bundle is subdivided into 16 chunks arranged in a 4 x 4 matrix. Since this is performed at runtime, you can use any applicable chunk size.

This is a big advantage since you can easily tweak the chunk size during development or even in a deployed app for performance optimization on different platforms.

# Useful Selector Methods

`public List<RectInt>? AllLocators` Get all of the locators for this Selector. This subdivides the entire bounds of the TileFab into N RectInts of a size equal to the Selector's Chunk Size.

`BoundsInt SelectorTotalSize(TpZoneLayout layout)` The returned BoundsInt defines the size of a single complete image ie the entirety of the Selector's TileFab. By default this is placed in Quadrant 1 of a 2D plane whose origin is at m_Layout.m_WorldOrigin and the size is always a square.

# Selector Query

```
List<TemplateSelectorQueryResults<T>> GetTilePlusTilesOfType<T>(Func<T, string, bool>?
filter  = null,
                                                                int size = 16,
```

```
                                                                object? options = null)
```

**Note**: the options parameter is reserved.

where `SelectorQueryResults` is:

```
/// <summary>
/// name of TileMap
/// </summary>
public readonly string m_MapName;

/// <summary>
/// the tile instance of type T
/// </summary>
public readonly T m_Tile;

/// <summary>
/// the position
/// </summary>
public readonly Vector3Int m_Position;
```

So the return value is a list of TPT tiles of Type T in the *entire* TileFab, the name of the Tilemap that they're placed on, and the position.

# What's this for?

When you load a new TileScene you will need information from it. For example, if you are using waypoints you might want to find out all the waypoints so you can position your player at a specific location: the most recently enabled waypoint.

But since in general only those chunks near the camera are loaded, not all of the TPT tiles in the TScene are actually loaded yet and hence are not locatable via the TpLib query methods.

You've loaded a save file with a GUID of a 'Start' waypoint but how do you find it if it isn't loaded yet??

Using `GetTilePlusTilesOfType<T>` is an easy way to extract this information. You can see how it's used in the next section.

This is a generic Query method which examines the Selector's TileFab, returning a list of information about each TPT tile.

The method accesses **ALL** the TileFab's bundles to create a list of a particular type of TPT tile, with filtering. Intended to be used during a scene or game init as this can take a while, depending on how many bundles are in the TileFab and how many TPT tiles are in each bundle.

It's NOT something to use inside a Monobehaviour Update: since the results are the same for each method call (unless the TileFab is changed, which is unlikely) , so cache the results if not needed immediately.

If you do cache the results please note that holding on to any tile references after the TScene is unloaded will result in memory leaks, so avoid that if at all possible or ensure that the references are nulled when you change TScenes.

The filter uses the tile ASSET and name of Tilemap as input params, returning a bool. If that bool is false then the tile is excluded from the method's output.

It's important to note that:

- The tile returned is a locked TilePlus tile ASSET in the PROJECT and not an INSTANCE in the scene.
  - So don't mess with it. Reading the contents of fields is A-OK.
- Only the string name of the Tilemap (as preserved in the TileFab) is available.
- The bundles aren't unarchived, the locked TilePlus tile assets are examined directly
- In general it's a bad idea to maintain references to these tiles; doing so will cause a memory leak. So don't cache the TemplateSelectorQueryResults outside of the calling method.

---

Revision #13
Created 2025-07-11 11:49:30 UTC by Vonchor
Updated 2025-09-05 19:38:19 UTC by Vonchor