# Eternal vs Dark Ages

This page covers the critical differences between Eternal and Dark Ages entities files.

It is assumed you have some experience with editing Doom Eternal entities files. While important differences do exist between the files, there are substantial similarities to Eternal. Check out the book on **Eternal Level Modding** if this subject is new to you. Most of the knowledge you'll gain there is applicable to Dark Ages.

## Serialization

In Doom Eternal, entities files are plaintext and serialized by the game at runtime. In Dark Ages, they are already serialized. See **Serialization and Atlan Mod Packager** for the rules and restrictions they follow, compared to regular decls.

## Submaps

The first difference you'll notice in a Dark Ages entities file, is that every entity has a number assigned to it. This value is it's submap index.

```
entity 3 {
<entity data>
}
entity 4 {
<entity data>
}
entity 4 {
<entity data>
}
```

Doom Eternal entity files were a single, long, uninterrupted list of entities. In Dark Ages, entities are split into multiple lists based on the submap they're located in. Each list starts with it's own `idWorldSpawn` entity named `world`.

idTech8 implements a "World Composition" system that divides the level's world into submaps, loaded and unloaded at will. Think about idTech7's **refmaps** - except they're no longer just a tool for developer workflow.

Due to this World Composition system, Dark Ages does not load every entity on map load. Only entities from the active submaps are loaded. The active submaps depend on your location in the level.

## Identifying Submaps

Now that we know what submaps are, how do we figure out what region of a level they belong to? Many mapentities contain a couple dozen submaps, so it's important to figure this out.

There are a few techniques to help you figure things out.

1. Compare what entities are close by to a given location in the world. (Use EntitySlayer's spawnPosition filter!)
2. Check the names of the other entities in the submaps
3. Check the `entityPrefix` variable in the submap's `world` entity.

The following world entities come from `m4_siege.mapentities`

```
// Prefix: cin_locations_atlan_cockpit
// Conclusion: This submap is for the cinematic at the end of Siege Part 2
// (Many submaps exist solely for cinematics. This makes sense, since many cinematics
// load entirely different worlds you don't see during gameplay.)
entity 1 {
	entityDef world {
		inherit = "entityDef/worldspawn";
		expandInheritance = false;
		systemVars = {
			entityType = "idWorldspawn";
		}
		edit = {
			entityPrefix = "cin_locations_atlan_cockpit";
			levelSoundState = "soundstate/level_mix/none";
			globalAIsettings = "default";
			automapDecl = "automap/default";
		}
	}
}

// Prefix: s4_battlegrounds_north_cave
// Conclusion: This submap is for one of the 2 detached cave segments in Siege Part 1
// Further analyze the submap's other entities to determine which cave it's for.
entity 14 {
	entityDef world {
		inherit = "entityDef/worldspawn";
```

```
	expandInheritance = false;
	systemVars = {
		entityType = "idWorldspawn";
	}
	edit = {
		entityPrefix = "s4_battlegrounds_north_cave";
		levelSoundState = "soundstate/level_mix/none";
		globalAIsettings = "default";
		automapDecl = "automap/default";
	}
}
}
```

## Adding New Entities

When adding new entities to a file, you must assign them a submap index based on what part of the level they'll be appearing in. Use the above techniques to identify the submaps relevant to the part of the level you want to edit. Then, give your modded entities the appropriate submap index.

**Only use a submap index that already exists in the vanilla file.**

**Make sure a submap's `world` entity stays listed before any of it's other entities**

## Submap 0

Entities in most submaps will be loaded/unloaded based on where you are in the level. But Submap 0 functions as the "persistent level". Entities placed in submap 0 will always be loaded and active.

## Multi-Level Entities

As a consequence of this enhanced loading system, levels are a lot bigger. So big, that sometimes two different levels' entities are contained in the same file! For example:

- *Siege Part 1* and *Siege Part 2* are both part of `mapentities/maps/game/sp/m4_siege/m4_siege.mapentities`.
- *Abyssal Forest* and *Ancestral Forge* are both part of `mapentities/maps/game/sp/m5_forge/m5_forge.mapentities`

# Layers

DOOM Eternal made constant use of the layer's system. It still exists in Dark Ages, but only for specific purposes.

If an entity has a layer, it will look something like this:

```
entity 14 {
	layerIndex = 1;
	layers = {
		"spawn_target_layer"
	}
    entityDef some_entity { ... }
}
```

Dark Ages entities can only be added to ONE layer. The `layerIndex` property is important. You may think of it as the "ID" for a layer. It must be included when specifying a layer. **The same layer may have a different layerIndex value across different submaps.** The consequences of using a layer whose layerIndex is not already defined in the submap, is unknown.

# End-of-File Data

If you scroll down to the bottom of a mapentities file, you'll see a massive blob of encoded data:

```
headerchunk {
	<massive amount of alphabetical data>
}
```

This is the header chunk of the original file. It is there intentionally, and **you must not touch it**. This data is critical for correctly serializing the file.

---

Created 2025-11-14 16:14:44 PST by FlavorfulGecko5
Updated 2026-09-12 09:07:58 PDT by FlavorfulGecko5