MapResources Files

MapResources files are located in the .resource archives. Each map in the game (including global maps like `common` and `warehouse`) have a MapResource file.

Their resource type string is `file` and their paths are `generated/buildgame/<MAP_PATH>.mapresources`

For example: `generated/buildgame/game/sp/m1_intro/m1_intro.mapresources` is the MapResources file for Village of Khalim. You can see the full list of maps at the bottom of the `packagemapspec.json`

Purpose

idTech7 and idTech8 have a 3-layer system that controls what files are loaded by the game:

1. The `packagemapspec.json` controls what .resources and .streamdb archives are used in each level.
2. The **container.mask** determines what files in the available .resources archives can be loaded, and which are disabled.
3. The MapResources Files determines what files a level will actually load from the pool of available files.

Structure

The MapResources files are massive lists of asset paths. There are some small differences in the format between idTech7 and idTech8

```
struct string_t {
	u32 length; // Little Endian
    char* data; // Not null-terminated
}

template<typename TYPE>
struct list_t {
	u32 num; // Big Endian
    TYPE* data;
```

```
}

struct entry_t {
	u32 typeindex;     // Big-Endian. Index into the typeStrings array
    string_t filename;
    u64 layermask[3];  // Little-Endian. See next section for explanation
}

struct MapResources {
	u32 timestamp; // This field is absent in idTech8 mapresources

    // List of all layers used in this level
    // For global maps (i.e. common and warehouse)
    // this list will always be empty.
    list_t<string_t> layers;

    u32 unknown; // Always 0

    // Resource type strings. These are the capitalized versions
    // (i.e. for 'generated/decls/aicomponentlist/.../' the
    // type string will be 'aiComponentList')
    list_t<string_t> typeStrings;

    list_t<entry_t> entries;

    // Names of maps.
    // This list does not exist in idTech8 mapresources
    list_t<string_t> maps;
}
```

Layer Mask

Each entry in the MapResources has a 24-byte "layer mask". Each layer in the file's layer list is assigned one bit in the mask. Layer index 0 is given bit 0, layer index 1 is given bit 1, etc.

The game uses this bitmask to conditionally load (and presumably unload) resources based on what layers are currently active in the level. Using a layer-specific resource in layers it's not assigned to is a potential cause of instability if the original layer is not always active.

The last bit in the layer mask (the most-significant bit of the third u64) is a special flag. If this is flag is set, the layer mask is ignored and the entry's data will always be loaded. Every entry in the global

mapresources (common.mapresources, warehouse.mapresources, etc.) has this bit set, since those have no layers. However, even in level-specific mapresources files, the majority of entries also have this flag set.

In Dark Ages, where the layer system is mostly unused, the only files consistently bound to specific layers are AI animation files. These are bound to the `spawn_target_layer` of each level. In practice, this should mean they're always loaded anyways.

DOOM Eternal makes extensive use of the layer system, with some levels having more than 64 layers. Roughly ~10% of the level-specific mapresource entries use the layermask. Layer-bound entries can include (but aren't limited to) things like:

- AI Animation files (like in Dark Ages)
- Models, material2 decls and images for layer-specific level geometry
- Havok shapes for layer-specific trigger volumes.

Force-Loading Assets

Unlike the packagemapspec and container mask, the MapResources files don't seem to be a strict filter on what files can be loaded. It is evident that idTech can force-load unlisted assets that a listed asset is dependent on. This will occur while parsing the listed assets. Unfortunately, this behavior is not predictable, and understanding of when it occurs is mostly speculation. It *seems* that certain classes of decls have force-loading enabled, while others don't. This would explain why adding new assets to the game via modding sometimes works without editing the mapresources, and sometimes doesn't. The key factor is where those new assets are referenced.

However, idTech8 seems to have stricter rules for force-loading assets. This is likely due to the pre-serialization of entities and entityDef files. Serialized files seem unable to be force-loaded, and don't have forced-loading enabled for their dependencies.

When an asset is force-loaded in this manner, the console will log a message like this:

> WARNING: idResourceStorageDiskStreamer::GetFile failed to find entry for 'generated/decls/damage/damage/dummy.decl' while loading damage:damage/dummy from edit.damageDecls.item from aiDamageDeclCollection:slayer/mace

This message was logged in Dark Ages, and can be interpreted as followed: "While loading an idDeclAIDamageDeclCollection" file, an unlisted damage decl with path "damage/dummy" was encountered in one of it's properties. This file was force-loaded despite not being in any mapresources entry list.

Revision #1
Created 19 August 2026 19:36:14 by FlavorfulGecko5
Updated 19 August 2026 20:33:49 by FlavorfulGecko5