Format Overview

> The assetsinfo JSON format allows for advanced control over mod injector behavior.

The most common uses of assetsinfo JSON include:

- Adding brand new textures or .decl files to the game (instead of just replacing another file)
- Controlling which of the game's .resources and .streamdb archives are loaded in a map (this allows modders to use assets that aren't normally loaded in the level/map they're working with).

Format Overview

```
{
// experimental - used to add new map layers
    "layers":[
        {
            "name":""
        }
    ],

    // experimental - used to add new maps
    "maps":[
        {
            "name":""
        }
    ],

    // used for adding, removing, or re-ordering of .resources files
    "resources":[
        {
            "name":"",
            "remove":,
            "placeFirst":,
            "placeBefore":,
            "placeByName":""
        }
```

```
    ],

    // used for adding, removing, or re-ordering of game assets
    "assets":[
        {
            "name":"",
            "mapResourceType":"",
            "resourceType":"",
            "streamDbHash":,
            "version":,
            "remove":,
            "placeBefore":,
            "placeByName":"",
            "placeByType":"",
            "specialByte1":,
            "specialByte2":,
            "specialByte3":
        }
    ]
}
```

"Layers" array

The first part of the assetsinfo JSON format is the `layers` array. This can be used to add additional layers to a map, which can then be referenced via .entities.

Each item in this array contains an object that has the `name` property. This field is required for each entry. In there you can set the name of the layer you want to create. It's that simple.

Here is an example usage that adds three custom layers:

```
{
    "layers":[
        {
            "name":"game/sp/custom/my_custom_layer"
        },
        {
            "name":"game/sp/custom/my_custom_layer_2"
        },
        {
```

```
            "name":"game/sp/custom/my_custom_layer_3"
        }
    ]
}
```

> **Note:** It is unclear what effect this has on gameplay, since it is possible to add new layers to the game without this step, and all existing layers in the game can already be used in any level without additional steps anyway.

"Maps" array

The second part of the assetsinfo JSON format is the `maps` array.

As with the `layers` array, each item contains an object that has the `name` property. In there you can set the name of the map you want to create. This feature is currently of limited use, since we can't create new maps anyway.

Here is an example usage that adds two custom maps:

```
{
    "maps":[
        {
            "name":"maps/game/sp/custom/my_custom_map"
        },
        {
            "name":"maps/game/sp/custom/my_custom_map_2"
        }
    ]
}
```

"Resources" array

The third part of the assetsinfo JSON format is the `resources` array. This one is a bit more complicated, so let's begin by explaining what this is actually for.

In DOOM Eternal, each map has a list of **.resources** and **.streamdb** files that will be loaded in the map itself. With this feature, you can tell the game to load additional .resources and/or .streamdb files in maps. In other words, **you can load files that aren't normally loaded**, giving you access to game assets and entities that normally aren't available. You can also tell the game to stop loading .resources or .streamdb files that are loaded by default in maps.

This feature is what allows us to load things like spirits and other DLC-specific enemies into the base campaign.

Below is an example usage. This would add the `e5m1_spear.resources` file, as well as several .streamdb files, to a map that it is not normally loaded in. Don't worry if you don't understand this - it is explained in more detail on the **Reference: "resources" array** page of the guide**.**

```
{
    "resources":[
        {
            "name":"e5m1_spear.resources"
        },
        {
            "name":"gameresources_6_1.streamdb",
            "placeBefore":true,
            "placeByName":"gameresources_0_4.streamdb"
        },
        {
            "name":"gameresources_0_6.streamdb",
            "placeBefore":true,
            "placeByName":"gameresources_0_4.streamdb"
        },
        {
            "name":"gameresources_1_8.streamdb",
            "placeBefore":true,
            "placeByName":"gameresources_0_4.streamdb"
        },
        {
            "name":"gameresources_2_2.streamdb",
            "placeBefore":true,
            "placeByName":"gameresources_0_4.streamdb"
        },
        {
            "name":"gameresources_3_2.streamdb",
            "placeBefore":true,
            "placeByName":"gameresources_0_4.streamdb"
        },
        {
            "name":"gameresources_4_3.streamdb",
```

```
            "placeBefore": true,
            "placeByName": "gameresources_0_4.streamdb"
        },
        {
            "name": "gameresources_5_3.streamdb",
            "placeBefore": true,
            "placeByName": "gameresources_0_4.streamdb"
        }
    ]
}
```

"Assets" array

The last feature in the assetsinfo JSON is the `assets` array. This is the most complicated one to understand as it has a lot of fields. In most cases, you will only need to use a few of them.

The main purpose of this section is to add entirely new assets to the game. (Most mods work by replacing another asset that already exists, rather than adding something new). For more details, refer to the **Reference: "assets" array** page of the guide.

An example usage is below. This would add a new image file to the game:

```
{
    "assets":[
        {
            "resourceType":"image",
            "version": 21,

"name":"custom_textures/custom_emissive_texture.tga$bc1$streamed$mtlkind=bloommask",
            "mapResourceType":"image"
        }
    ]
}
```

See Also

- **Reference: "resources" Array**

Revision #8
Created 6 October 2021 16:31:47 by SamPT
Updated 21 January 2024 18:44:23 by Alby