

GAMECONF formal specification 0.99.1

While there are ways to define how a Doom mod should be set up, a way of explicitly defining the correct way to set up the engine (including IWADs) does not exist outside of specialised functionality in a handful of ports. This specification documents a data format used to configure a Doom session to run a mod successfully and eliminate confusion from the user end.

JSON lump

This specification uses the JSON Lump 1.0.0 formal specification as the root of its data storage, with a type of “**gameconf**” and a version of “**1.0.0**”.

Minimum engine featureset

This specification applies to any **limit-removing** featureset or greater. While it is optional for other valid featuresets, it is a requirement of the **ID24** featureset.

Resolving GAMECONF lumps

Each IWAD and PWAD is allowed to have a GAMECONF lump. These are loaded and resolved in the following order:

- The default IWAD
- Each PWAD listed in order via the **-file** command line parameter (or equivalent)

Most values inside GAMECONF lumps are considered to be destructive, ie they overwrite values previously defined by another GAMECONF. However, PWAD files are considered additive, ie they add to an array of files to resolve by the engine later; options are considered additive; while the **executable** and **mode** values are resolved via a maximum operation.

Every value inside a GAMECONF definition is allowed to be null. This indicates that values defined by a previous GAMECONF lump are not to be altered.

Note that for the purposes of resolving GAMECONF lumps, you must only resolve the IWADs and PWADs defined by the user before engine initialisation. As such, any GAMECONF lump defined by a PWAD is expected to define every value it needs for functioning in a valid manner. Additional IWADs and PWADs defined by a GAMECONF are not to be parsed.

After GAMECONF lumps have been parsed, the engine will have an IWAD and a list of PWADs that it should then load and initialise for the game to continue in the usual manner.

Supported engine versions

Doom engine versions are supported in an additive manner, ie starting with the base feature level set of Doom 1.9 and turning on features the higher version you go. The current supported engine version for the **version** field, starting from the least features and continuing through to most features, are:

- **doom1.9** - Equivalent to the final DOS retail releases of The Ultimate Doom, Doom II, and Final Doom.

- **limitremoving** - Certain limits removed to stop crashes and memory overwrites; includes a select few line actions from Boom that only affect visual features and do not break demo compatibility.
- **bugfixed** - A limit removing mode that fixes well known bugs that outright break demo compatibility with **doom1.9**.
- **boom2.02** - The Boom feature set as originally released in 1998.
- **complevel9** - Sky transfers from MBF, MUSINFO, longtic demos, and a few other bits and bobs that do not exist in Boom 2.02 but are generally accepted by the community to mean “Boom compatible”.
- **mbf** - The MBF 2.03 feature set as released in 1999.
- **mbfextra** - Everything from **mbf** plus DEHEXTRA.
- **mbf21** - The MBF21 feature set as released in 2021.
- **mbf21ex** - Everything from **mbf21** plus DSDHACKED.
- **id24** - The featureset described by this group of specifications.

Any source port not implementing any of those feature sets should error when encountering an unknown featureset. Any feature set not defined here is not considered part of the specification, and encountering any value but the above is to be considered an error condition.

Limit removing ports are allowed to upgrade **doom1.9** to **limitremoving** on load. They should not upgrade the version any higher in order to retain the functionality expected by the mod author.

Supported modes

The Doom engine, and certain specifications such as UMAPINFO, alter their behavior based on the game mode. This is usually defined by the name of the IWAD. However this is not suitable for custom IWADs; and for mods using a lower feature set there may be no other valid way of altering the engine’s behavior.

The three modes supported, from lowest value to highest, are:

- **registered** - Doom with 3 episodes
- **retail** - Doom with 4 episodes
- **commercial** - Doom II with maps 1-32 and all additional enemies and items

Note that while **shareware** is a supported mode inside the Doom source code, this is only a valid method for the original distribution model of shareware Doom. Mod authors can tailor a “demo” version of their mods in a far more flexible fashion with modern tools and features. As such, encountering **shareware** or any other value not defined above is to be considered an error condition.

IWADs

IWADs defined on the command line, or determined by the engine, are to be considered default values. If any GAMECONF lump encountered defines a different IWAD, it will overwrite this default and that original IWAD will not be loaded by the game. This must be the name of a file without any additional path info. Defining a path, be it relative or absolute, is considered an error condition.

PWADs

PWADs defined by the end user need special consideration when building a full PWAD list to load. The list is constructed in the following way:

- For each PWAD declared
 - Add to the new PWAD list
 - Check the PWAD for a valid GAMECONF
 - If a valid GAMECONF is found
 - Add each PWAD defined in the GAMECONF lump to the new PWAD list

This results in a PWAD with a GAMECONF lump having its supporting PWADs loaded in the correct order as it expects.

As with IWADs, defining a path for a PWAD is considered an error condition.

Player translations

The **playertranslations** field allows a mod to override the built-in translation tables defined for player sprites. The order of this table is equivalent to the player number. As such, any less than four entries in this table is to be considered an error condition.

Each encountered GAMECONF lump with a valid player translations entry will completely overwrite this array. Any mod that wants to define player translations should do so in an explicit and complete manner.

Options

The existence of the OPTIONS lump outright indicates an **mbf**-minimum featureset. However, there are a few compatibility options that you will want to configure for lesser feature sets. The options field serves this purpose, by allowing you to set compatibility options that satisfy the minimum feature set requirement.

All normal options configurable by the OPTIONS lump are available to the user. The options you are allowed to set are limited by the version the user specifies. If an option is declared that exists outside of the defined version, it is ignored.

As options are additive, they will accumulate for each encountered GAMECONF lump and will always resolve in that order.

Options defined by the MBF standard are limited to MBF-minimum versions; while options defined by the MBF21 standard are limited to MBF21-minimum versions. The following exceptions apply:

Option	Minimum version	Reason
comp_soul	doom1.9	Lost soul bouncing changes behaviour between Doom and Doom II. Allow explicit controlling of this for doom1.9 versions.
comp_finaldoomteleport	doom1.9	Option did not previously exist; allows

		Final Doom teleport behavior without needing to define a special version.
comp_moveblock	mbfextra	Previously undefined by any standard; exists in prBoom and is being retroactively applied to the mbfextra feature set.
comp_musinfo	complevel9	Previously undefined by any standard; an optional feature that became expected of complevel 9 thanks to prBoom. Defaults to true on complevel9 and higher; false otherwise
comp_thingfloorlight	boom2.02	Previously undefined by any standard; controls whether things are lit by the sector's floor lighting or the sector's normal lighting. Defaults to true in complevel9 and higher; false otherwise.

Note that all new options described above can also be parsed by the OPTIONS lump.

Title, author, description, and version

These fields are descriptors for the given WAD a GAMECONF is describing. These are effectively cosmetic as far as the game is concerned; they are however useful fields for frontend/UI/tools purposes.

Data type definitions

root

Name	Type	Description
title	string	The name of this mod. Can be null.
author	string	The authors of this mod. Can be null.
description	string	A description of this mod. Can be null.
version	string	The release version for this mod. Can be null.
iwad	string	The filename of the IWAD this mod should use as its base. Can be null.
pwads	array of string	A list of additional PWADs to load for this mod to function correctly. Can be null.
playertranslations	array of string	A list of translations to use for player sprites. Can be null.
executable	string	The feature set this mod requires to run. Can be

		null.
mode	string	Which mode this mod should run in. Can be null.
options	string	A string representing options to set. Uses the JSON string convention of newlines being represented as the <code>\n</code> linebreak. Can be null.

Reference implementation details

By default, the game will run in limit-removing mode instead of ID24 mode. It's demonstrably true that most content authored for Doom and Doom II will not be using the ID24 feature set, so it makes no sense for that to be the default mode for the game to run in. Due to needing to remove undefined behavior, limit-removing is the default as that accounts for fixing the various array overflows that the original Doom and Doom II releases were prone to.

From here, features get turned on as requested and as proven by the data itself.

GAMECONF forms a core part of how the feature level of a session is determined. While it is considered the authoritative way for how to set a feature level for the session, an analysis path to determine the true feature set exists.

The flow for how a session is set up at a feature and a data level is as follows:

- For each IWAD and PWAD passed to the game session
 - Load the WAD directory
 - Search for and parse GAMECONF as defined above
 - Unload the WAD directory
- For each IWAD and PWAD obtained from the GAMECONF pass
 - Load the WAD directory
 - Search for and parse a DEHACKED lump
- Load any available map info definitions in the following order of preference:
 - UMAPINFO
 - DMAPINFO
- Determine current session feature level by the maximum value encountered from the following checks
 - Current gameconf declared level
 - DeHackEd maximum encountered features
 - Feature level found in COMPLVL lump
 - Feature level defined by DEMOx lumps
 - Feature level of Boom 2.02 if ANIMATED, SWITCHES, C_BEGIN, or C_END lumps are found
 - Feature level of MBF if OPTIONS lump is found
 - For each map
 - Maximum feature level of encountered line special, sector special, and thing encountered
- Unload every WAD directory
- Ensure all features required by the determined feature level are enabled or disabled according to spec

- Load extras.wad
- If ID24 feature level, load id24res.wad
- Load each IWAD and PWAD defined by the current gameconf

In the case of linedef specials, while it is well defined which line special belongs to which standard the Boom line special number 85 - which is undefined in vanilla - is treated as limit removing. Some mapsets that were authored to be vanilla-compatible in the 90s and early 00s used this special to improve the presentation in Boom-compatible ports, and the autodetect path incorrectly elevates the feature level to Boom thanks to this one linedef. As it is a texture scroller and has no impact on the simulation, it is considered a limit-removing line.

In the case of thing flags, the Boom “not in deathmatch” and “not in coop” flags are not tested for similar reasons as the linedef special number 85.

The MBF reserved thing flag is respected and limits thing flag checks to those defined in vanilla Doom if encountered. A similar flag (0x0800, 2048, 1 << 11) is used for linedef flags for the same justifications provided in the MBF specification for thing flags.