

## ID24 formal specification 0.99.1

As a part of developing features for the new Doom + Doom II release, certain features were needed in order to make development easier and to provide a solid foundation for any and all future releases. To collect all these features in one place that source ports can implement and use, a new feature specification dubbed ID24 has been created.

### Baseline features

ID24 is a superset of the following standards and features:

- Vanilla Doom
- Boom
- MUSINFO
- MBF
- DEHEXTRA
- MBF21
- DSDHACKED
- UMAPINFO

A complete ID24 implementation requires support for those standards and features.

For simplicity's sake, these features are also enumerated in the following order, from the least amount of features to the most amount of features and referred to as a **feature level**:

- Doom 1.9
- Limit Removing (with UMAPINFO)
- Limit Removing With Bug Fixes
- Boom 2.02
- complevel 9 (with MUSINFO)
- MBF
- DEHEXTRA
- MFB21
- DSDHACKED
- ID24

This ordering is important for a few features described by the ID24 specification.

### JSON lumps

A key feature that all new data formats use is the JSON Lump specification. Rather than create a new data format for each new data type, a standard JSON root element has been created that can be parsed and interpreted by both tools and source ports. As a standardised format, this cuts down on the amount of work required to implement new datatypes and provides a simple way to support new datatypes in the future.

### Major features

Each major feature of ID24 is covered in a separate document. These features are:

- DEMOLOOP - a way to customise the Doom demo loop
- Finale lumps - a way to customise Doom finales, and allow continuing to another map from any finale

- GAMECONF - a way to describe how to set up the runtime environment of any WAD
- Mapping additions - new linedef types and thing numbers for mappers to use
- ID24HACKED - extensions to DeHackEd, plus a way to verify valid DeHackEd states before parsing additional DeHackEd lumps
- Interlevel lumps - a way to define background animations on victory screens
- SBARDEF - a way to create custom status bars
- SKYDEFS - a way to create custom skies, including Doom 64 fire skies
- Translations - a way to create new translations for palette swapping

### New placeable things

Perhaps of most interest to the community, Doom + Doom II's new episode Legacy of Rust includes a suite of new monsters, decorations, and weapons for use in your maps. These new things are exposed to everyone via a new set of table entries that do not conflict with any previous community standard or usage of DeHackEd indices.

### Required data

All required data to support the **ID24** specification is contained within the **id24res.wad** file. As this is a commercial product, note that illegal redistribution of this file is covered by normal copyright laws. Source ports should use their normal IWAD resolution rules to locate and load this WAD before the main IWAD when **ID24** compatibility is enabled.

### Music formats

A complete **ID24** implementation supports the following additional audio formats for music playback:

- OGG Vorbis
- Tracker

### Future revisions

Future revisions of any part of the ID24 specification and feature set will incur the following version number changes for the following reasons:

- Major - breaking changes, not backward compatible
- Minor - additions and minor changes
- Revision - bug fixes and clarifications

As ID24 aims to be a stable feature set, major revisions will aim to be restricted to a new feature set specification.

### Guiding principles for ID24 specifications

As a set of features developed for a commercial product, ID24 needed to satisfy the requirements of shipping on multiple vendor-controlled platforms, compatibility with community standards, and ensuring that the future commercial viability of Doom and Doom II would be secured.

Rather than assume generic names for data fields, explicit naming for ID24 has been taken in cases where a conflict is likely to occur. This is most pertinent in the DeHackEd features that have been added, where source ports (and even the original Heretic) have taken to calling additional flags fields "flags2", "flags3", etc. Naming such fields for ID24 at the code

and data level makes it explicit what functionality set these fields should be used for, and avoids conflicting with prior or future functionality.

There are many standards over the years that take the assumption of the implementation before it as a part of its standard. A primary example is UMAPINFO, which changes behavior depending on game mode, and even relies on undocumented behavior on how an episode number is resolved. Rather than assume a prior implementation, an explicit complete implementation is defined and specced. This includes the JSON data level, which have all been authored as reflections of data structures rather than a way to describe modifying existing structures.

When creating a new feature and specification, an attempt was made to see if there were suitable community standards. While some source ports have solutions for what has been specced, there is nothing standard and widely adopted amongst a broad range of source ports. These newly developed specifications will be the basis for any and all official content that may or may not be commissioned in the future and as such have been developed to replicate functionality that was already there, add anything required for this release, and leave a path open for future development and iteration on these features and specifications.