

SBARDEF formal specification 0.99.1

A way of defining custom status bar layouts has not existed in most Doom source ports. This specification documents a data format and corresponding features that replicates and expands Doom's status bar rendering functionality.

Scope of features

This specification has been written to replicate the Doom and Doom II: Hell on Earth status bar, while expanding its abilities to handle rendering as a full screen overlay. Very little has been added to the featureset that extends beyond these requirements.

JSON lump

This specification uses the JSON Lump 1.0.0 formal specification as the root of its data storage, with a type of "**statusbar**" 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.

Only one lump

Status bars are expected to be defined with the **SBARDEF** lump. If this lump does not exist, then the original behaviour for status bars is to be used. For WAD resolution purposes, this lump follows the standard rules and resolves to the last lump found in the WAD dictionary with that name.

As there is only one lump, any custom status bars are expected to redefine an entire status bar chain and not just redefine the bar they are interested in.

ID24 weapons

To properly support ID24 weapons, id24res.wad contains a SBARDEF lump that adds weapon slots 8 and 9 to the status bar arms section. As a source port should load that WAD when ID24 compatibility is enabled, the status bar will be completely ready for a user to enjoy the new weapons with.

Runtime environment

Status bars are to be rendered in a virtual 320x200 resolution when in fullscreen mode (referred to as the virtual environment), presented at a 4:3 aspect ratio with rectangular pixels at a 1:1.2 aspect ratio (ie 1.2 vertical units are required to maintain the 4:3 aspect ratio). This is identical to the original Doom and Doom II presentation. The virtual resolution is enforced to allow equivalent presentations at any actual device output resolutions.

As non-fullscreen statusbars specify their height, the virtual resolution becomes **320 x <statusbar height>**. While coordinates are specified with a (0,0) origin representing the top-left corner of the status bar, the bar itself is rendered to the screen using a virtual Y coordinate of **200 - <statusbar height>**.

Resources are allowed to render outside of the virtual environment. This allows for the presentation to adapt to widescreen aspect ratios.

Elements are **not allowed** to specify coordinates outside of the virtual environment. Elements are allowed to extend outside of the virtual environment and will be clipped appropriately to the output resolution, but placement of elements must be inside the virtual environment.

Update rate

Any logic that a status bar element requires to run (such as updating animations) must run at the same tic rate as the game simulation (35Hz). Rendering can be handled independently of framerate.

Selecting status bars to render

Selection of which status bar to render is to be handled by expanding the definition of the `screenblocks` value. In the original Doom codebase, values 0 to 10 define how to scale the 3D first-person viewport, while value 11 indicates full screen rendering. Under this new system, assuming zero-indexing of an array the index of a status bar to render is obtained with the following formula:

```
barindex = Max( screenblocks - 10, 0 );
```

Ports that do not allow selecting `screenblocks` above 11 should take care to further clamp that value to a maximum of 11.

Scaling status bars to differing aspect ratios

A rendering target aspect ratio of less than 4:3 is treated differently for fullscreen and non-fullscreen status bars:

- For fullscreen and non-fullscreen status bars, compact mode resolves to true and the status bar definition can provide an alternative layout.
- For non-fullscreen status bars, the bar is scaled down in both X and Y axes until the entire 320 virtual width of the bar fits on the screen.

The 3D viewport is expected to adapt to the scaling of the status bar.

A rendering target aspect ratio of greater than 4:3 has no effect on fullscreen statusbars. For non-fullscreen status bars, the extra columns on either side of the status bar are filled with a flat that is either defined by the status bar itself; or equivalent to the border graphic used in each game (FLOOR7_2 for Doom; GRNROCK for Doom II). This must be rendered using the virtual 320x200 space and not the native render resolution, with the top left corner of the flat occupying the (0, 200 - <statusbar height>) position.

Status bar number fonts

Status bar number fonts exist exclusively to represent numbers. They do not handle any other glyphs outside of the minus and percentage symbols.

A number font uses standard Doom patches to render its glyphs.

The glyphs for numbers 0 through 9 must be resolvable from the WAD dictionary. If a minus or percentage glyph is not defined, it is simply ignored.

A number font can have one of the following types:

- 0 - mono spaced, based off the width of the 0 glyph (default for Doom and Doom II)
- 1 - mono spaced, based off the widest glyph
- 2 - proportional, each glyph takes exactly its width

Monospaced fonts render at the cursor position and increment the cursor by the defined width, giving the impression of gaps on the right side of each glyph for glyphs that are narrower than the monospace width.

Status bar elements

Status bar elements are hierarchical in nature. They exist in a transform tree, where child elements inherit the transform of their parent. The only values that are currently considered a part of the transform are the X and Y position values.

Status bar elements can define an alignment. This exclusively controls how the elements it renders are positioned from the transform. It does not affect the transform for child items. The default of top and left renders the top-left corner of its element at the defined transform; bottom and right renders the bottom-right corner of its element at the defined transform; while the middle values for both horizontal and vertical render the middle of its element at the defined transform. One horizontal and one vertical alignment is allowed to be defined per element.

Each element is capable of containing any number of child elements.

Each element can render with or without a transparency map and a translation. The transparency map is a lookup table as defined by the Boom specification; while a translation is defined by the ID24 Translation formal specification.

Element types

The following element types exist for status bar definitions to use:

- Canvas
- Graphic
- Animation
- Face
- Face Background
- Number
- Percentage

At the data definition level, each element must contain at least one definition for an element type. Only the first entry in an element is considered. Any other element types are simply ignored and not parsed. Editors as such only want to save out the data required for the current element type rather than defining separate entries for each type.

Canvas element

A canvas element is an empty element that exists as a container for children elements.

Graphic element

A graphic element renders the defined patch at the transform's location.

Animation element

An animation element sequentially displays the defined patch at the transform's location. Each frame has a duration which defines the minimum amount of time it must be rendered for.

Face element

A face element is the portrait that animates and renders in the middle of Doom and Doom II's original status bars. Only the face is rendered with this element; its background is handled separately.

Face Background element

A face background element renders the background patch as retrieved from the current player's translation definition.

Number element

A number element renders the specified number, including its negative values if the specified font has a minus glyph defined.

Percentage element

A number element renders the specified number with a percentage glyph on the right-hand side of the number, including its negative values if the specified font has a minus glyph defined.

Number types

The following types of numbers can be resolved by a Number and Percentage element via the **type** field:

- 0 - current player's health
- 1 - current player's armor
- 2 - current player's frags
- 3 - current player's ammo amount for the ammo specified by **param**
- 4 - current player's ammo amount for the currently selected weapon
- 5 - current player's maximum ammo amount for the ammo specified by **param**
- 6 - current player's ammo amount for the weapon specified by **param**
- 7 - current player's maximum ammo amount for the weapon specified by **param**

Element conditions

A status bar element can have any number of conditions that must all resolve to true for both that element and its child elements to render. When zero conditions are defined by specifying null in the JSON document, a resolution of true is determined. Using an empty array to define zero conditions is considered an error condition.

Element conditions **do not** stop the element from updating during the game tic, they only stop the element from rendering.

The following conditions are available to a status bar element:

Type	Description
------	-------------

0	Whether the weapon defined by param is owned
1	Whether the weapon defined by param is selected
2	Whether the weapon defined by param is not selected
3	Whether the weapon defined by param has a valid ammo type
4	Whether the selected weapon has a valid ammo type
5	Whether the ammo type defined by param matches the selected weapon's ammo type
6	Whether any weapon in a slot defined by param is owned
7	Whether any weapon in a slot defined by param not owned
8	Whether any weapon in a slot defined by param is selected
9	Whether any weapon in a slot defined by param is not selected
10	Whether the item defined by param is owned
11	Whether the item defined by param is not owned
12	Whether the current game version is greater than or equal to the feature level defined by param
13	Whether the current game version is less than the feature level defined by param
14	Whether the session type is equal to the type defined by param
15	Whether the session type is not equal to the type defined by param
16	Whether the game mode is equal to the mode defined by param
17	Whether the game mode is equal to the mode defined by param
18	Whether the hud mode is equal to the mode defined by param

The session type has the following values:

- 0 - single player
- 1 - cooperative
- 2 - deathmatch (any kind)

The game version and game mode parameters match those defined in the GAMECONF specification.

The item parameters match those defined by the **Pickup item type** Thing field in the IDHACKED24 specification. Note that the following item types always resolve to **false**:

- -1 - no item

- 0 - message only
- 8 - health bonus
- 9 - stimpack
- 10 - medikit
- 11 - soulsphere
- 12 - megasphere
- 13 - armor bonus

The hud mode has the following values:

- 0 - standard layout
- 1 - compact layout

Data type definitions

root

Name	Type	Description
numberfonts	array of numberfont	An array of number fonts. Must not be null; must contain at least one entry.
statusbars	array of statusbar	An array of status bars. Must not be null; must contain at least one entry.

numberfont

Name	Type	Description
name	string	An identifier used to resolve this number font.
type	integer	One of the values matching the type enumeration described in the number font section.
stem	string	A string representing the first few characters of each glyph's name in the WAD dictionary. Note that 3 characters is the maximum if you want to resolve the minus and percent glyphs; longer stems are allowed thanks to Doom and Doom II requiring it for the STYS number font.

statusbar

Name	Type	Description
height	integer	The height of this status bar in virtual units.
fullscreenrender	boolean	Whether this statusbar is a fullscreen overlay or a standard status bar.
fillflat	string	The name of the flat to fill the extended virtual space with
children	array of sbarelem	An array of child elements.

sbarelem

Name	Type	Description
canvas	canvas	Contents to describe this element as a Canvas. Can be undefined.
graphic	graphic	Contents to describe this element as a Graphic. Can be undefined.
animation	animation	Contents to describe this element as an Animation. Can be undefined.
face	face	Contents to describe this element as a Face. Can be undefined.
facebackground	facebg	Contents to describe this element as a Face Background. Can be undefined.
number	number	Contents to describe this element as a Number. Can be undefined.
percent	percent	Contents to describe this element as a Percent. Can be undefined.

canvas, face, facebg

Name	Type	Description
x	integer	The virtual x position of this element.
y	integer	The virtual y position of this element.
alignment	bitfield	The alignment of this element's rendered graphics.
tranmap	string	The name of a transparency map lump to resolve. Can be null.
translation	string	The name of a translation to resolve. Can be null.
conditions	array of condition	A series of conditions to meet to render both this and all child elements. Can be null; an array length of 0 is considered an error condition.
children	array of sbarelem	An array of child elements. Can be null; an array length of 0 is considered an error condition.

condition

Name	Type	Description
condition	enum	The type of condition to resolve, as described by the table in the "Element conditions" section
param	integer	A parameter as described for each condition type.

graphic

Name	Type	Description
x	integer	The virtual x position of this element.
y	integer	The virtual y position of this element.
alignment	bitfield	The alignment of this element's rendered graphics.
tranmap	string	The name of a transparency map lump to resolve. Can be null.
translation	string	The name of a translation to resolve. Can be null.
conditions	array of condition	A series of conditions to meet to render both this and all child elements. Can be null; an array length of 0 is considered an error condition.
children	array of sbarelem	An array of child elements. Can be null; an array length of 0 is considered an error condition.
patch	string	The name of a patch lump to resolve.

animations

Name	Type	Description
x	integer	The virtual x position of this element.
y	integer	The virtual y position of this element.
alignment	bitfield	The alignment of this element's rendered graphics.
tranmap	string	The name of a transparency map lump to resolve. Can be null.
translation	string	The name of a translation to resolve. Can be null.
conditions	array of condition	A series of conditions to meet to render both this and all child elements. Can be null; an array length of 0 is considered an error condition.
children	array of sbarelem	An array of child elements. Can be null; an array length of 0 is considered an error condition.
frames	array of frame	An array of frames to be rendered in sequential order. Must be an array of at least one element; anything else is an error condition.

frame

Name	Type	Description
lump	string	The name of a patch lump to resolve.
duration	number	The minimum amount of time to display this frame in seconds.

number, percent

Name	Type	Description
x	integer	The virtual x position of this element.
y	integer	The virtual y position of this element.
alignment	bitfield	The alignment of this element's rendered graphics.
tranmap	string	The name of a transparency map lump to resolve. Can be null.
translation	string	The name of a translation to resolve. Can be null.
conditions	array of condition	A series of conditions to meet to render both this and all child elements. Can be null; an array length of 0 is considered an error condition.
children	array of sbarelem	An array of child elements. Can be null; an array length of 0 is considered an error condition.
font	string	The name of the number font to render this number with.
type	enum	The number to resolve.
param	integer	The parameter to use for number resolution.
maxlength	integer	The maximum number of digits to render, including the minus glyph and excluding the percentage glyph.

Reference implementation details

The SBARDEF files that come with extras.wad and id24res.wad contain the classic status bar, a full screen overlay status bar, and a blank full screen. Demos will always use the last status bar in an SBARDEF when loaded, so any custom SBARDEFs should also define a blank fullscreen as the final statusbar entry.