# libpvf Reference



PlayStation®Vita Programmer Tool Runtime Library Release 3.50



Document serial number: 000004892117

Document serial number: 000004892117

# Table of Contents

# Constant Definitions

# ScePvfFamilyCode

Font family codes

### Definition

```
#include <font/libpvf.h>
typedef enum ScePvfFamilyCode {
        SCE_PVF_DEFAULT_FAMILY_CODE=(0),
        SCE_PVF_FAMILY_SANSERIF=(1),
        SCE_PVF_FAMILY_SERIF=(2),
        SCE_PVF_FAMILY_ROUNDED=(3)
} ScePvfFamilyCode;
```

### Description

This is one piece of font design information.

It is the value that is returned in the *familyCode* member of the ScePvf_t_fontStyleInfo type structure by the scePvfGetFontList() function for getting font information for vector fonts that are installed on the PlayStation®Vita.

| | |
|---|---|
| SCE_PVF_FAMILY_SANSERIF | Font of the sans serif family |
| SCE_PVF_FAMILY_SERIF | Font of the serif family |
| SCE_PVF_FAMILY_ROUNDED | Reserved value |

If the following value is assigned to the *familyCode* member of the ScePvf_t_fontStyleInfo type structure that is provided as an argument to the scePvfFindOptimumFont() or scePvfFindFont() function for finding a font, the font family that is the system default is selected.

| | |
|---|---|
| SCE_PVF_DEFAULT_FAMILY_CODE | Default |

### See Also

scePvfGetFontList(), scePvfFindOptimumFont(), scePvfFindFont(), ScePvf_t_fontStyleInfo, ScePvfTFontStyleInfo

# ScePvfStyleCode

Style codes

### Definition

```
#include <font/libpvf.h>
typedef enum ScePvfStyleCode {
        SCE_PVF_DEFAULT_STYLE_CODE=(0),
        SCE_PVF_STYLE_REGULAR=(1),
        SCE_PVF_STYLE_OBLIQUE=(2),
        SCE_PVF_STYLE_NARROW=(3),
        SCE_PVF_STYLE_NARROW_OBLIQUE=(4),
        SCE_PVF_STYLE_BOLD=(5),
        SCE_PVF_STYLE_BOLD_OBLIQUE=(6),
        SCE_PVF_STYLE_BLACK=(7),
        SCE_PVF_STYLE_BLACK_OBLIQUE=(8),
        SCE_PVF_STYLE_L=(101),
        SCE_PVF_STYLE_M=(102),
        SCE_PVF_STYLE_DB=(103),
        SCE_PVF_STYLE_B=(104),
        SCE_PVF_STYLE_EB=(105),
        SCE_PVF_STYLE_UB=(106)
} ScePvfStyleCode;
```

### Description

This is one piece of font design information.

It is the value that is returned in the *style* member variable of the ScePvf_t_fontStyleInfo type structure by the scePvfGetFontList() function for getting font information for vector fonts that are installed on the PlayStation®Vita.

The following style codes are mainly used for Latin fonts.

| | |
|---|---|
| SCE_PVF_STYLE_REGULAR | Standard design |
| SCE_PVF_STYLE_OBLIQUE | Italic |
| SCE_PVF_STYLE_NARROW | Narrow |
| SCE_PVF_STYLE_NARROW_OBLIQUE | Narrow italic |
| SCE_PVF_STYLE_BOLD | Bold |
| SCE_PVF_STYLE_BOLD_OBLIQUE | Bold italic |
| SCE_PVF_STYLE_BLACK | Thicker bold |
| SCE_PVF_STYLE_BLACK_OBLIQUE | Thicker bold italic |

The following style codes are mainly used for Japanese fonts.

| | |
|---|---|
| SCE_PVF_STYLE_L | Narrower |
| SCE_PVF_STYLE_M | ↑ |
| SCE_PVF_STYLE_DB | |
| SCE_PVF_STYLE_B | |
| SCE_PVF_STYLE_EB | ↓ |
| SCE_PVF_STYLE_UB | Bolder |

If the following value is assigned to the *style* member of the ScePvf_t_fontStyleInfo type structure that is provided as an argument to the scePvfFindOptimumFont() or scePvfFindFont() function for finding a font, the style that is the system default is selected.

| | |
|---|---|
| SCE_PVF_DEFAULT_STYLE_CODE | System standard |

**See Also**

`scePvfGetFontList()`, `scePvfFindOptimumFont()`, `scePvfFindFont()`, `ScePvf_t_fontStyleInfo`, `ScePvfTFontStyleInfo`

# ScePvfImageBufferPixelFormatType

Pixel formats

### Definition

```
#include <font/libpvf.h>
typedef enum ScePvfImageBufferPixelFormatType {
        SCE_PVF_USERIMAGE_DIRECT4_L=(0),
        SCE_PVF_USERIMAGE_DIRECT8=(2)
} ScePvfImageBufferPixelFormatType;
```

### Description

This is the value that is assigned in the *pixelFormat* member variable of the ScePvf_t_userImageBufferRec type structure that is passed to the scePvfGetCharGlyphImage() function for copying the font glyph images to the user memory space.

It specifies the format of the pixels that constitute a glyph image.

The scePvfGetCharGlyphImage() function copies the font glyph images to the memory that was specified by the user according to this format value.

| | |
|---|---|
| SCE_PVF_USERIMAGE_DIRECT4_L | The low-order 4 bytes within 8 bits are the direct color grayscale 4 bits that are placed on the left side of the screen. The pixel value 0x0 means the minimum brightness and the pixel value 0xf means the maximum brightness. |
| SCE_PVF_USERIMAGE_DIRECT8 | 256-color direct color grayscale 8 bits. The pixel value 0x00 means minimum brightness and the pixel value 0xff means maximum brightness. |

### See Also

scePvfGetCharGlyphImage(), ScePvf_t_userImageBufferRec, ScePvfTUserImageBufferRec

# ScePvfLanguageCode

Language codes (for fonts that use language)

## Definition

```
#include <font/libpvf.h>
typedef enum ScePvfLanguageCode {
        SCE_PVF_DEFAULT_LANGUAGE_CODE=(0),
        SCE_PVF_LANGUAGE_J=(1),
        SCE_PVF_LANGUAGE_LATIN=(2),
        SCE_PVF_LANGUAGE_K=(3),
        SCE_PVF_LANGUAGE_C=(4),
        SCE_PVF_LANGUAGE_CJK=(5)
} ScePvfLanguageCode;
```

## Description

This is information about the language or languages corresponding to the font.

It is the value that is returned in the *languageCode* member variable of the ScePvf_t_fontStyleInfo type structure by the scePvfGetFontList() function for getting font information for vector fonts that are installed on the PlayStation®Vita.

| | |
|---|---|
| SCE_PVF_LANGUAGE_J | Japanese |
| SCE_PVF_LANGUAGE_LATIN | English |
| SCE_PVF_LANGUAGE_K | Korean |
| SCE_PVF_LANGUAGE_C | Chinese |
| SCE_PVF_LANGUAGE_CJK(*) | Corresponds to Japanese, Korean, and Chinese |

(*) SCE_PVF_LANGUAGE_CJK fonts (single fonts supporting Japanese, Korean and Chinese) do not exist in the vector fonts installed on the PlayStation®Vita.

If the following value is assigned for the *languageCode* member of the ScePvf_t_fontStyleInfo type structure that is provided as an argument to the scePvfFindOptimumFont() or scePvfFindFont() functions for finding a font, the font language code that is the system default is selected.

| | |
|---|---|
| SCE_PVF_DEFAULT_LANGUAGE_CODE | Default |

## See Also

scePvfGetFontList(), scePvfFindOptimumFont(), scePvfFindFont(), ScePvf_t_fontStyleInfo, ScePvfTFontStyleInfo

# ScePvfRegionCode

Region codes (for fonts that use region)

## Definition

```
#include <font/libpvf.h>
typedef enum ScePvfRegionCode {
        SCE_PVF_GENERIC_REGION_CODE=(0),
        SCE_PVF_REGION_001=(1),
        SCE_PVF_REGION_002=(2),
        SCE_PVF_REGION_003=(3),
        SCE_PVF_REGION_004=(4),
        SCE_PVF_REGION_005=(5),
        SCE_PVF_REGION_006=(6),
        SCE_PVF_REGION_007=(7)
} ScePvfRegionCode;
```

## Description

This is information about the region corresponding to the font.

It is the value that is returned in the *regionCode* member variable of the ScePvf_t_fontStyleInfo type structure by the scePvfGetFontList() function for getting font information for vector fonts that are installed on the PlayStation®Vita.

| | |
|---|---|
| SCE_PVF_REGION_001 | Reserved value |
| SCE_PVF_REGION_002 | Reserved value |
| SCE_PVF_REGION_003 | Reserved value |
| SCE_PVF_REGION_004 | Reserved value |
| SCE_PVF_REGION_005 | Reserved value |
| SCE_PVF_REGION_006 | Reserved value |
| SCE_PVF_REGION_007 | Reserved value |

If the following value is assigned for the *regionCode* member of the ScePvf_t_fontStyleInfo type structure that is provided as an argument to the scePvfFindOptimumFont() or scePvfFindFont() functions for finding a font, a font for which the region is unspecified is selected.

| | |
|---|---|
| SCE_PVF_GENERIC_REGION_CODE | Unspecified |

## See Also

scePvfGetFontList(), scePvfFindOptimumFont(), scePvfFindFont(), ScePvf_t_fontStyleInfo, ScePvfTFontStyleInfo

# ScePvfFontVendorCountryCode

Font vendor country codes

## Definition

```
#include <font/libpvf.h>
typedef enum ScePvfFontVendorCountryCode {
        SCE_PVF_GENERIC_COUNTRY_CODE=(0),
        SCE_PVF_COUNTRY_JAPAN=(1),
        SCE_PVF_COUNTRY_USA=(2)
        SCE_PVF_COUNTRY_KOREA=(3)
} ScePvfFontVendorCountryCode;
```

## Description

This is the country code of the font vendor.

It is the value that is returned in the *countryCode* member variable of the ScePvf_t_fontStyleInfo type structure by the scePvfGetFontList() function for getting font information for vector fonts that are installed on the PlayStation®Vita.

| | |
|---|---|
| SCE_PVF_COUNTRY_JAPAN | Japan |
| SCE_PVF_COUNTRY_USA | United States of America |
| SCE_PVF_COUNTRY_KOREA | Korea |

If the following value is assigned for the *countryCode* member of the ScePvf_t_fontStyleInfo type structure that is provided as an argument to the scePvfFindOptimumFont() or scePvfFindFont() functions for finding a font, a font for which the vendor's country is unspecified is selected.

| | |
|---|---|
| SCE_PVF_GENERIC_COUNTRY_CODE | Country unspecified |

## See Also

scePvfGetFontList(), scePvfFindOptimumFont(), scePvfFindFont(), ScePvf_t_fontStyleInfo, ScePvfTFontStyleInfo

# ScePvfBoolValue

Boolean values

### Definition

```
#include <font/libpvf.h>
typedef enum ScePvfBoolValue {
        SCE_PVF_FALSE=(0),
        SCE_PVF_TRUE=(1)
} ScePvfBoolValue;
```

### Description

This is used to specify the Boolean values that are handled by libpvf.

| | |
|---|---|
| SCE_PVF_FALSE | False |
| SCE_PVF_TRUE | True |

# ScePvfDataAccessMode

Access modes

### Definition

```
#include <font/libpvf.h>
typedef enum ScePvfDataAccessMode {
        SCE_PVF_FILEBASEDSTREAM=(0),
        SCE_PVF_MEMORYBASEDSTREAM=(1)
} ScePvfDataAccessMode;
```

### Description

This value is used to specify the mode for accessing font data.

| | |
|---|---|
| `SCE_PVF_FILEBASEDSTREAM` | Font data in a file is handled directly as a file. |
| `SCE_PVF_MEMORYBASEDSTREAM` | All font data in a file is read into memory and handled as data in memory.<br>The file itself is closed when the data is read into memory.<br>Memory that was allocated for file reading by using the `scePvfOpen()`, `scePvfOpenUserFile()`, or `scePvfOpenUserFileWithSubfontIndex()` function is released when the `scePvfClose()` function is executed. |

### See Also

`scePvfOpen()`, `scePvfOpenUserFile()`, `scePvfOpenUserMemory()`
`scePvfOpenUserFileWithSubfontIndex()`, `scePvfOpenUserMemoryWithSubfontIndex()`

# SCE_PVF_SUBSTYLE_xxx

Substyle attribute mask value (for specifying emboldened and italicized typefaces)

## Definition

```
#include <font/libpvf.h>
#define SCE_PVF_SUBSTYLE_VERTICALLAYOUT (0x0001)
#define SCE_PVF_SUBSTYLE_PSEUDO_BOLD (0x0002)
#define SCE_PVF_SUBSTYLE_PSEUDO_SLANT (0x0004)
```

## Description

This is one piece of font design information.

It is the value that is returned in the *subStyle* member variable of the ScePvf_t_fontStyleInfo type structure by the scePvfGetFontList() function for getting font information for vector fonts that are installed on the PlayStation®Vita.

| | |
|---|---|
| SCE_PVF_SUBSTYLE_VERTICALLAYOUT | Typeface for vertical layout |
| SCE_PVF_SUBSTYLE_PSEUDO_BOLD | Style for which pseudo boldface processing was executed |
| SCE_PVF_SUBSTYLE_PSEUDO_SLANT | Style for which pseudo italic processing was executed |

(*) The vector fonts that are installed on the PlayStation®Vita do not have the SCE_PVF_SUBSTYLE_VERTICALLAYOUT, SCE_PVF_SUBSTYLE_PSEUDO_BOLD, and SCE_PVF_SUBSTYLE_PSEUDO_SLANT substyles. Any vector font can be emboldened (or unemboldened) at any time using scePvfSetEmboldenRate() and any vector font can be italicized at any time using scePvfSetSkewValue().

## See Also

scePvfGetFontList(), scePvfFindOptimumFont(), scePvfFindFont(), ScePvf_t_fontStyleInfo, ScePvfTFontStyleInfo, scePvfSetEmboldenRate(), scePvfSetSkewValue()

# SCE_PVF_FONTNAME_LENGTH

Font name maximum length

### Definition

```
#include <font/libpvf.h>
#define SCE_PVF_FONTNAME_LENGTH (64)
```

### Description

This is the maximum length of string data that is used when libpvf handles a font name.
The font name is the name of the font, not the filename of the font.
Different font files may have the same font name.

### See Also

scePvfGetFontList(), scePvfFindOptimumFont(), scePvfFindFont(),
ScePvf_t_fontStyleInfo, ScePvfTFontStyleInfo

# SCE_PVF_FONTFILENAME_LENGTH

Font filename maximum length

### Definition

```
#include <font/libpvf.h>
#define SCE_PVF_FONTFILENAME_LENGTH (64)
```

### Description

This is the maximum length of string data that includes a device name and directory name, which is used when libpvf handles a font filename.

### See Also

scePvfGetFontList(), scePvfFindOptimumFont(), scePvfFindFont(), ScePvf_t_fontStyleInfo, ScePvfTFontStyleInfo

# SCE_PVF_MAX_OPEN

Maximum number of fonts that can be open simultaneously

### Definition

```
#include <font/libpvf.h>
#define SCE_PVF_MAX_OPEN (18)
```

### Description

This is the maximum number of fonts that can be open simultaneously, which has been determined within libpvf.

When multiple library instances of libpvf are generated, the maximum number of fonts that can be open simultaneously per library instance is less than the `SCE_PVF_MAX_OPEN` value. The upper bound of the file descriptor resources assigned by the kernel for accessing the filesystem is related to this.

### See Also

`scePvfNewLib(),scePvfOpen(),scePvfClose()`

# SCE_PVF_MIN_EMBOLDEN_RATE, SCE_PVF_MAX_EMBOLDEN_RATE

Maximum and minimum values for emboldening (or unemboldening)

### Definition

```
#include <font/libpvf.h>
#define SCE_PVF_MIN_EMBOLDEN_RATE (-20.0f)
#define SCE_PVF_MAX_EMBOLDEN_RATE (40.0f)
```

### Description

Minimum and maximum values passed to the `scePvfSetEmboldenRate()` function.

### See Also

`scePvfSetEmboldenRate()`

# SCE_PVF_MIN_SKEW_VALUE, SCE_PVF_MAX_SKEW_VALUE

Minimum and maximum skew values

### Definition

```
#include <font/libpvf.h>
#define SCE_PVF_MIN_SKEW_VALUE (-30.0f)
#define SCE_PVF_MAX_SKEW_VALUE (30.0f)
```

### Description

Minimum and maximum values passed to the `scePvfSetSkewValue()` function.

A positive skew value indicates a clock-wise direction.

### See Also

`scePvfSetSkewValue()`

# Variable Types

# List of Variable Types

Simple variable types defined in libpvf.h

**Definition**

| Type Name | Entity | Use |
|---|---|---|
| ScePvf_t_u64 | unsigned long long | Unsigned 64-bit type |
| ScePvf_t_s64 | signed long long | Signed 64-bit type |
| ScePvf_t_u32 | unsigned long | Unsigned 32-bit type |
| ScePvf_t_s32 | signed long | Signed 32-bit type |
| ScePvf_t_u16 | unsigned short | Unsigned 16-bit type |
| ScePvf_t_s16 | signed short | Signed 16-bit type |
| ScePvf_t_u8 | unsigned char | Unsigned 8-bit type |
| ScePvf_t_s8 | signed char | Signed 8-bit type |
| ScePvf_t_f32 | float | 32-bit floating-point type |
| ScePvf_t_f64 | double | 64-bit floating-point type |
| ScePvf_t_bool | ScePvf_t_u32 | Boolean type |
| ScePvf_t_libId | void * | ScePvf library handle pointer |
| ScePvf_t_fontId | void * | ScePvf font handle |
| ScePvf_t_pointer | void * | General pointer type |
| ScePvf_t_handle | void * | General ID type (same meaning as ScePvf_t_pointer) |
| ScePvf_t_error | ScePvf_t_s32 | For error values |
| ScePvf_t_int | ScePvf_t_s32 | Integer type |
| ScePvf_t_charCode | ScePvf_t_u16 | Character code type |
| ScePvf_t_string | ScePvf_t_charCode * | String type |
| ScePvf_t_fontIndex | ScePvf_t_s32 | Font number determined by system |

# Datatypes

# ScePvf_t_irect, ScePvfTIRect

General rectangle data type

### Definition

```
#include <font/libpvf.h>
typedef struct ScePvf_t_irect {
        ScePvf_t_u16 width;
        ScePvf_t_u16 height;
} ScePvf_t_irect, ScePvfTIRect;
```

### Members

| | |
|---|---|
| *width* | Width represented by a 16-bit integer |
| *height* | Height represented by a 16-bit integer |

### Description

This is a data type (structure) for handling the width and height of a general rectangle. Both type names (`ScePvf_t_irect` and `ScePvfTIRect`) can be used in the same way.

It is used by data types or functions that are used for processing in which the units are the pixels of a glyph image.

### See Also

`ScePvf_t_userImageBufferRec`, `ScePvfTUserImageBufferRec`, `scePvfGetCharImageRect()`

# ScePvf_t_rect, ScePvfTRect

General rectangle data type

### Definition

```
#include <font/libpvf.h>
typedef struct ScePvf_t_rect {
        ScePvf_t_u32 width;
        ScePvf_t_u32 height;
} ScePvf_t_rect, ScePvfTRect;
```

### Members

| | |
|---|---|
| *width* | Width represented by a 32-bit integer |
| *height* | Height represented by a 32-bit integer |

### Description

This is a data type (structure) for handling the width and height of a general rectangle. Both type names (`ScePvf_t_rect` and `ScePvfTRect`) can be used in the same way.

It is used by data types or functions that are used for processing in which the units are the pixels of a glyph image.

# ScePvf_t_cacheSystemInterface, ScePvfTCacheSystemInterface

Data type of interface with font cache system

**Definition**

```
#include <font/libpvf.h>
typedef struct ScePvf_t_cacheSystemInterface {
        ScePvf_t_pointer *cacheInstance;
        ScePvf_t_s32 (*lockFunc)
              (
              ScePvf_t_pointer
              );
        ScePvf_t_s32 (*unlockFunc)
              (
              ScePvf_t_pointer
              );
        ScePvf_t_pointer (*findFunc)
              (
              ScePvf_t_pointer,
              ScePvf_t_u32,
              ScePvf_t_pointer,
              ScePvf_t_bool *
              );
        ScePvf_t_s32 (*writeKeyValueToCacheFunc)
              (
              ScePvf_t_pointer,
              ScePvf_t_pointer,
              ScePvf_t_pointer
              );
        ScePvf_t_s32 (*write0ToCacheFunc)
              (
              ScePvf_t_pointer,
              ScePvf_t_pointer,
              ScePvf_t_pointer,
              ScePvf_t_int
              );
        ScePvf_t_s32 (*write1ToCacheFunc)
              (
              ScePvf_t_pointer,
              ScePvf_t_pointer,
              ScePvf_t_pointer,
              ScePvf_t_int
              );
        ScePvf_t_s32 (*read0FromCacheFunc)
              (
              ScePvf_t_pointer,
              ScePvf_t_pointer,
              ScePvf_t_pointer
              );
        ScePvf_t_s32 (*read1FromCacheFunc)
              (
              ScePvf_t_pointer,
              ScePvf_t_pointer,
              ScePvf_t_pointer
              );
} ScePvf_t_cacheSystemInterface, ScePvfTCacheSystemInterface;
```

### Members

| | |
|---|---|
| *cacheInstance* | Value passed to first argument of cache interface function group |
| *lockFunc* | Pointer to function for locking the cache |
| *unlockFunc* | Pointer to function for unlocking the cache |
| *findFunc* | Pointer to function for checking whether or not the cache exists |
| *writeKeyValueToCacheFunc* | Pointer to function for writing a key value to the cache |
| *write0ToCacheFunc* | Pointer to function for writing data0 to the cache |
| *write1ToCacheFunc* | Pointer to function for writing data1 to the cache |
| *read0FromCacheFunc* | Pointer to function for reading data0 from the cache |
| *read1FromCacheFunc* | Pointer to function for reading data1 from the cache |

### Description

libpvf can be assigned a cache system from an outside source.

libpvf communicates with the cache system via the function group assigned by this data type. Both type names (`ScePvf_t_cacheSystemInterface` and `ScePvfTCacheSystemInterface`) can be used in the same way. When libpvf calls this function group, it passes *cacheInstance* in the first argument.

The cache system function group specifications expected by libpvf are as follows.

Cache system initialization and termination processing are not performed from within libpvf. Make sure that the application performs processing as necessary before libpvf initialization and after libpvf termination processing.

**result = lockFunc ( cacheInstance )**

This function locks the cache.

The cache system must not accept requests from elsewhere while the cache is locked.

When locking is successful, 0 is returned for *result*. When locking fails, -1 is returned.

**result = unlockFunc ( cacheInstance )**

This function unlocks the cache.

When unlocking is successful, 0 is returned for *result*. When unlocking fails, -1 is returned.

**cacheSlot = findFunc ( cacheInstance, hashValue, key, result )**

This function uses the value of *hashValue* to check whether or not the data indicated by *key* exists within the cache.

*hashValue* is a hash value generated by libpvf. If this value is used, comparison processing with data in the cache can be performed quickly. However, libpvf does not expect that the cache system always implements and uses this value. The cache system itself is permitted to generate a hash value from *key*.

*key* is a pointer to a `ScePvfCacheKey` type variable (structure), and `ScePvfCacheKey` has four member variables. This function compares data within the cache with *key* and if all four member variables match, the cache data and key are considered to be the same.

When the same data is not found in the cache, this function writes `SCE_PVF_FALSE` in `*`*result* and returns NULL in *cacheSlot*.

When the same data is found in the cache, this function writes `SCE_PVF_TRUE` in `*`*result* and returns a pointer to the cache slot where it was found in *cacheSlot*.

**result = writeKeyValueToCacheFunc ( cacheInstance, cacheSlot, key )**

This function stores the value of *key* in the specified cache slot *cacheSlot*.

If processing fails for some reason, -1 is returned for *result*. Otherwise, 0 is returned for *result*.

**result = write0ToCacheFunc ( cacheInstance, cacheSlot, data, dataSize )**

This function stores data with a size of the number of bytes indicated by *dataSize* from the area indicated by the pointer *data* as one data of the cache slot specified by *cacheSlot*.

This *cacheSlot* is the value that was returned as the return value by *findFunc*.

libpvf expects the cache system to manage four data blocks in one cache slot.

This function stores data in one (the 0th one) of those data blocks.

If processing fails for some reason, -1 is returned for *result*. Otherwise, 0 is returned for *result*.

**result = write1ToCacheFunc ( cacheInstance, cacheSlot, data, dataSize )**

This function stores data with a size of the number of bytes indicated by dataSize from the area indicated by the pointer *data* as one data of the cache slot specified by *cacheSlot*.

This *cacheSlot* is the value that was returned as the return value by *findFunc*.

libpvf expects the cache system to manage four data blocks in one cache slot.

This function stores data in one (the 1st one) of those data blocks.

If processing fails for some reason, -1 is returned for *result*. Otherwise, 0 is returned for *result*.

**result = read0FromCacheFunc ( cacheInstance, cacheSlot, dst )**

This function copies one (the 0th one) of the data from the cache slot specified by *cacheSlot* to the area indicated by the pointer *dst*.

This *cacheSlot* is the value that was returned as the return value by *findFunc*.

libpvf expects the cache system to manage four data blocks in one cache slot.

This function copies the data that was stored in one (the 0th one) of those data blocks to *dst*.

If processing fails for some reason, -1 is returned for *result*. Otherwise, 0 is returned for *result*.

**result = read1FromCacheFunc ( cacheInstance, cacheSlot, dst )**

This function copies one (the 1st one) of the data from the cache slot specified by *cacheSlot* to the area indicated by the pointer *dst*.

This *cacheSlot* is the value that was returned as the return value by *findFunc*.

libpvf expects the cache system to manage four data blocks in one cache slot.

This function copies the data that was stored in one (the 1st one) of those data blocks to *dst*.

If processing fails for some reason, -1 is returned for *result*. Otherwise, 0 is returned for *result*.

libpvf operates even if no cache system is assigned. If NULL is assigned for the member variable *cache* of the `ScePvf_t_initRec` structure that is assigned by an argument in the `scePvfNewLib()` function, libpvf operates with no cache system.

If a cache system is assigned, the font cache is also used when the font data is accessed in `SCE_PVF_MEMORYBASEDSTREAM` mode.

## See Also

`scePvfNewLib()`, `ScePvf_t_initRec`, `ScePvfTInitRec`, sample program "fontcache", `SCE_PVF_MEMORYBASEDSTREAM`, `SCE_PVF_FILEBASEDSTREAM`

# ScePvf_t_initRec, ScePvfTInitRec

Data type of information specified when use of libpvf begins

## Definition

```
#include <font/libpvf.h>
typedef struct ScePvf_t_initRec {
        ScePvf_t_pointer userData;
        ScePvf_t_u32 maxNumFonts;
        ScePvf_t_cacheSystemInterface *cache;
        ScePvf_t_pointer reserved;
        ScePvf_t_pointer (*allocFunc)
              (
              ScePvf_t_pointer,
              ScePvf_t_u32
              );
        ScePvf_t_pointer (*reallocFunc)
              (
              ScePvf_t_pointer,
              ScePvf_t_pointer,
              ScePvf_t_u32
              );
        void (*freeFunc)
              (
              ScePvf_t_pointer,
              ScePvf_t_pointer
              );
} ScePvf_t_initRec, ScePvfTInitRec;
```

## Members

| | |
|---|---|
| *userData* | Pointer to user data |
| *maxNumFonts* | Maximum number of fonts that are open simultaneously |
| *cache* | Cache system instance handle |
| *reserved* | Reserved area. 0 must be set. |
| *allocFunc* | Memory allocation function |
| *reallocFunc* | Memory reallocation function |
| *freeFunc* | Memory release function |

## Description

Both names of this data type (ScePvf_t_initRec and ScePvfTInitRec) can be used in the same way.

For *userData*, specify the value that is passed in the first argument when the user-provided memory allocation and release functions or file access functions are called by libpvf.

For *maxNumfonts*, specify the number of fonts that can be open simultaneously. The maximum value that can be specified is SCE_PVF_MAX_OPEN. This value cannot be exceeded.

For *cache*, specify a user-provided cache system instance.

The memory allocation and release function specifications expected by libpvf are as follows.

### p = allocFunc ( userData, size )

*userData* is the value that was assigned for the member variable *userData* of this structure.

This function allocates memory with a size of *size* bytes (0 <= *size*) aligned to a 4-byte boundary and returns p. When memory allocation fails, NULL is returned.

**p = reallocFunc ( userData, old_p, size )**

*userData* is the value that was assigned for the member variable *userData* of this structure.

This function resizes the area specified in *old_p* to the memory of *size* bytes (0 <= *size*) aligned to a 4-byte boundary, and returns p. When memory allocation fails, NULL is returned.

**freeFunc ( userData, p )**

*userData* is the value that was assigned for the member variable *userData* of this structure.

This function releases the memory of the area indicated by *p*.

## See Also

`ScePvf_t_cacheSystemInterface`, `ScePvfTCacheSystemInterface`, `scePvfNewLib()`

# ScePvf_t_fontStyleInfo, ScePvfTFontStyleInfo

Data type used for getting installed font information and for finding fonts

## Definition

```
#include <font/libpvf.h>
typedef struct ScePvf_t_fontStyleInfo {
        ScePvf_t_f32 weight;
        ScePvf_t_u16 familyCode;
        ScePvf_t_u16 style;
        ScePvf_t_u16 subStyle;
        ScePvf_t_u16 languageCode;
        ScePvf_t_u16 regionCode;
        ScePvf_t_u16 countryCode;
        ScePvf_t_u8 fontName [SCE_PVF_FONTNAME_LENGTH ];
        ScePvf_t_u8 styleName [SCE_PVF_STYLENAME_LENGTH ];
        ScePvf_t_u8 fileName [SCE_PVF_FONTFILENAME_LENGTH ];
        ScePvf_t_u32 extraAttributes;
        ScePvf_t_u32 expireDate;
} ScePvf_t_fontStyleInfo, ScePvfTFontStyleInfo;
```

## Members

| | |
|---|---|
| *weight* | Weight value |
| *familyCode* | Family code |
| *style* | Style |
| *subStyle* | Substyle |
| *languageCode* | Language code |
| *regionCode*(*) | Region code |
| *countryCode* | Font vendor country code |
| *fontName* | Font name string |
| *styleName* | Style name string |
| *fileName* | Font filename string |
| *extraAttributes* | Additional attribute information |
| *expireDate*(*) | Expiration date |

(*) There are no vector fonts installed on the PlayStation®Vita for which a region code is specified or for which an expiration date is specified.

## Description

This is a data type (structure) that is handled by the `scePvfGetFontList()` function for getting font information about vector fonts that are installed on the PlayStation®Vita. Both type names (`ScePvf_t_fontStyleInfo` and `ScePvfTFontStyleInfo`) can be used in the same way.

Before an application program that uses libpvf uses `scePvfOpen()` to open a libpvf font, it should use `scePvfGetFontList()` to check the fonts that are actually installed on the PlayStation®Vita to determine an appropriate font or it should use `scePvfFindOptimumFont()` to determine the font that should be selected.

**See Also**

ScePvfFamilyCode, ScePvfStyleCode, ScePvfLanguageCode, ScePvfRegionCode, ScePvfFontVendorCountryCode, SCE_PVF_SUBSTYLE_xxx, SCE_PVF_FONTNAME_LENGTH, SCE_PVF_FONTFILENAME_LENGTH, ScePvf_t_fontInfo, ScePvfTFontInfo, scePvfGetFontList(), scePvfFindOptimumFont(), scePvfFindFont(), scePvfGetFontInfoByIndexNumber()

# ScePvf_t_userImageBufferRec, ScePvfTUserImageBufferRec

Data type used when libpvf copies glyph images to user memory area

### Definition

```
#include <font/libpvf.h>
typedef struct ScePvf_t_userImageBufferRec {
        ScePvf_t_u32 pixelFormat;
        ScePvf_t_s32 xPos64;
        ScePvf_t_s32 yPos64;
        ScePvf_t_irect rect;
        ScePvf_t_u16 bytesPerLine;
        ScePvf_t_u16 reserved;
        ScePvf_t_u8 *buffer;
} ScePvf_t_userImageBufferRec, ScePvfTUserImageBufferRec;
```

### Members

| | |
|---|---|
| *pixelFormat* | Pixel format |
| *xPos64* | X-position of reference point for writing |
| *yPos64* | Y-position of reference point for writing |
| *rect* | Buffer horizontal and vertical size<br>Specifies the horizontal and vertical size of the image *buffer*.<br>The units of this value are pixels |
| *bytesPerLine* | Number of bytes per horizontal line of *buffer*<br>Specifies the number of bytes per horizontal line of the image *buffer*. |
| *reserved* | Padding (always set 0) |
| *buffer* | Pointer to *buffer* area<br>Specifies the address of memory allocated by the user application.<br>Be sure to specify an area that was allocated with a size of at least *bytesPerLine* * *rect.height* bytes. |

Specify the format of one pixel for *pixelFormat*. The following SCE_PVF_USERIMAGE_xxx values can be specified.

| | |
|---|---|
| SCE_PVF_USERIMAGE_DIRECT4_L | This is a pixel format in which 2 pixels are contained in 1 byte. The high-order 4 bits are assumed to be placed at the left side of the screen and the low-order four bits are assumed to be placed at the right side of the screen. The value 0x0 means the minimum brightness, and the value 0xf means the maximum brightness. |
| SCE_PVF_USERIMAGE_DIRECT8 | This is a pixel format in which 1 pixel is contained in 1 byte. The value 0x00 means the minimum brightness and the pixel value 0xff means the maximum brightness. |

For *xPos64* and *yPos64*, specify the position on the character base line that is to be the starting point when the scePvfGetCharGlyphImage() or scePvfGetCharGlyphImage_Clip() function copies a character. The units of this value are 1/64 of a pixel. libpvf, which also processes numeric values less than 1 pixel, calculates the position and brightness for copying the glyph image of a character.

## Description

libpvf copies the glyph image of a character to the user memory space according to the `scePvfGetCharGlyphImage()` or `scePvfGetCharGlyphImage_Clip()` function.

The destination memory to which both of these functions copy the glyph image is determined according to this data type. Both type names (`ScePvf_t_userImageBufferRec` and `ScePvfTUserImageBufferRec`) can be used in the same way.

By setting an appropriate CLUT, the user application can handle *`buffer`* as a texture and display it by mapping it to a polygon.

## See Also

`ScePvfImageBufferPixelFormatType`, `scePvfGetCharGlyphImage()`, `scePvfGetCharGlyphImage_Clip()`

# ScePvf_t_iGlyphMetricsInfo, ScePvfTIGlyphMetricsInfo

Data type of character glyph metric information (fixed-point format)

## Definition

```
#include <font/libpvf.h>
typedef struct ScePvf_t_iGlyphMetricsInfo {
        ScePvf_t_u32 width64;
        ScePvf_t_u32 height64;
        ScePvf_t_s32 ascender64;
        ScePvf_t_s32 descender64;
        ScePvf_t_s32 horizontalBearingX64;
        ScePvf_t_s32 horizontalBearingY64;
        ScePvf_t_s32 verticalBearingX64;
        ScePvf_t_s32 verticalBearingY64;
        ScePvf_t_s32 horizontalAdvance64;
        ScePvf_t_s32 verticalAdvance64;
} ScePvf_t_iGlyphMetricsInfo, ScePvfTIGlyphMetricsInfo;
```

## Members

| | |
|---|---|
| *width64* | Indicates the character width. The units are 1/64 pixel. |
| *height64* | Indicates the character height. The units are 1/64 pixel. |
| *ascender64* | Indicates the character ascender. The units are 1/64 pixel.<br>Normally, this is the same value as *horizontalBearingY64*. |
| *descender64* | Indicates the character *descender*.<br>The units are 1/64 pixel.<br>Normally, this is the same value as<br>*horizontalBearingY64* - *height64*. |
| *horizontalBearingX64* | This is the bearing value in the X-axis direction for horizontal character layout.<br>The units are 1/64 pixel. |
| *horizontalBearingY64* | This is the bearing value in the Y-axis direction for horizontal character layout.<br>The units are 1/64 pixel. |
| *verticalBearingX64*(*) | This is the bearing value in the X-axis direction for vertical character layout.<br>The units are 1/64 pixel. |
| *verticalBearingY64*(*) | This is the bearing value in the Y-axis direction for vertical character layout.<br>The units are 1/64 pixel. |
| *horizontalAdvance64* | This is the advance in the X-axis direction for horizontal character layout.<br>The units are 1/64 pixel. |
| *verticalAdvance64*(*) | This is the advance in the Y-axis direction for vertical character layout.<br>The units are 1/64 pixel. |

(*) Although all characters in the fonts that are installed on the PlayStation®Vita have numeric values for vertical layout metrics, vertical layout is not guaranteed.

### Description

This is a data type for representing glyph metrics information for one character. Both type names (`ScePvf_t_iGlyphMetricsInfo` and `ScePvfTIGlyphMetricsInfo`) can be used in the same way.

It is used by the `scePvfGetFontInfo()` function for indicating the maximum values for glyph images of all characters that are included in a font as typical values for the entire font.

### See Also

`ScePvf_t_fGlyphMetricsInfo`, `ScePvfTFGlyphMetricsInfo`, `ScePvf_t_charInfo`, `ScePvfTCharInfo`, `ScePvf_t_fGlyphMetricsInfo`, `ScePvfTFGlyphMetricsInfo`, `scePvfGetFontInfo()`, `scePvfGetCharInfo()`

# ScePvf_t_charInfo, ScePvfTCharInfo

Data type of character-specific information

## Definition

```
#include <font/libpvf.h>
typedef struct ScePvf_t_charInfo {
        ScePvf_t_u32 bitmapWidth;
        ScePvf_t_u32 bitmapHeight;
        ScePvf_t_s32 bitmapLeft;
        ScePvf_t_s32 bitmapTop;
        ScePvf_t_iGlyphMetricsInfo glyphMetrics;
        ScePvf_t_u8 reserved [4];
} ScePvf_t_charInfo, ScePvfTCharInfo;
```

## Members

| | |
|---|---|
| *bitmapWidth* | Always 0 |
| *bitmapHeight* | Always 0 |
| *bitmapLeft* | Always 0 |
| *bitmapTop* | Always 0 |
| *glyphMetrics* | Character metrics information<br>Indicates glyph metrics information. |
| *reserved* | Undefined |

## Description

This is a data type for indicating the numeric values of the bitmap size of the glyph image of a certain character and the offset from the baseline origin of that image and character metrics information. Both type names (`ScePvf_t_charInfo` and `ScePvfTCharInfo`) can be used in the same way.

In libpgf, *xPos64*, *yPos64* of `SceFont_t_userImageBufferRec` must be set while taking into consideration the values of *bitmapLeft* and *bitmapTop*. But in libpvf, this is processed within the library, so the application program does not need to take the values of *bitmapLeft*, *bitmapTop* into consideration.

Each of the member variables *bitmapWidth*, *bitmapHeight*, *bitmapLeft*, and *bitmapTop* are dummy variables created for holding structural compatibility with libpgf. The bitmap width and height after rasterizing can be obtained using the `scePvfGetCharImageRect()` function.

## See Also

`ScePvf_t_iGlyphMetricsInfo`, `ScePvfTIGlyphMetricsInfo`, `ScePvf_t_userImageBufferRec`, `ScePvfTUserImageBufferRec`, `scePvfGetCharInfo()`, `scePvfGetCharImageRect()`

# ScePvf_t_fGlyphMetricsInfo, ScePvfTFGlyphMetricsInfo

Data type of character metrics information (floating-point format)

## Definition

```
#include <font/libpvf.h>
typedef struct ScePvf_t_fGlyphMetricsInfo {
        ScePvf_t_f32 width;
        ScePvf_t_f32 height;
        ScePvf_t_f32 ascender;
        ScePvf_t_f32 descender;
        ScePvf_t_f32 horizontalBearingX;
        ScePvf_t_f32 horizontalBearingY;
        ScePvf_t_f32 verticalBearingX;
        ScePvf_t_f32 verticalBearingY;
        ScePvf_t_f32 horizontalAdvance;
        ScePvf_t_f32 verticalAdvance;
} ScePvf_t_fGlyphMetricsInfo, ScePvfTFGlyphMetricsInfo;
```

## Members

| | |
|---|---|
| *width* | Indicates the character width. The units are pixels. |
| *height* | Indicates the character height. The units are pixels. |
| *ascender* | Indicates the character ascender. The units are pixels.<br>Normally, this is the same value as *horizontalBearingY64*. |
| *descender* | Indicates the character *descender*. The units are pixels.<br>Normally, this is the same value as *horizontalBearingY64* - *height64*. |
| *horizontalBearingX* | This is the bearing value in the X-axis direction for horizontal character layout.<br>The units are pixels. |
| *horizontalBearingY* | This is the bearing value in the Y-axis direction for horizontal character layout.<br>The units are pixels. |
| *verticalBearingX* (*) | This is the bearing value in the X-axis direction for vertical character layout. The units are pixels. |
| *verticalBearingY* (*) | This is the bearing value in the Y-axis direction for vertical character layout. The units are pixels. |
| *horizontalAdvance* | This is the advance in the X-axis direction for horizontal character layout. The units are pixels. |
| *verticalAdvance* (*) | This is the advance in the Y-axis direction for vertical character layout. The units are pixels. |

(*) Although all characters in the fonts that are installed on the PlayStation®Vita have numeric values for vertical layout metrics, vertical layout is not guaranteed.

## Description

This is a data type for representing glyph metrics information for one character. Both type names (`ScePvf_t_fGlyphMetricsInfo` and `ScePvfTFGlyphMetricsInfo`) can be used in the same way.

It is used by the `scePvfGetFontInfo()` function for indicating the maximum values for glyph images of all characters that are included in a font as typical values for the entire font.

**See Also**

ScePvf_t_iGlyphMetricsInfo, ScePvfTIGlyphMetricsInfo, ScePvf_t_charInfo, ScePvfTCharInfo, scePvfGetFontInfo(), scePvfGetCharInfo()

# ScePvf_t_fontInfo, ScePvfTFontInfo

Data type of information related to font data in general

## Definition

```
#include <font/libpvf.h>
typedef struct ScePvf_t_fontInfo {
        ScePvf_t_iGlyphMetricsInfo maxIGlyphMetrics;
        ScePvf_t_fGlyphMetricsInfo maxFGlyphMetrics;
        ScePvf_t_u32 numChars;
        ScePvf_t_fontStyleInfo fontStyleInfo;
        ScePvf_t_u8 reserved [4];
} ScePvf_t_fontInfo, ScePvfTFontInfo;
```

## Members

| | |
|---|---|
| *maxIGlyphMetrics* | Maximum metrics value (fixed-point representation)<br>This is the maximum value of metrics information of all characters included in the font data represented as a fixed-point numeric value.<br>The units of the member variable are 1/64 pixel. |
| *maxFGlyphMetrics* | Maximum metrics value (floating-point representation)<br>This is the maximum value of metrics information of all characters included in the font data represented as a floating-point numeric value.<br>The units of the member variable are pixels. |
| *numChars* | Number of recorded character types<br>Indicates the number of all character types included in the font data. |
| *fontStyleInfo* | Indicates font style information. |
| *reserved* | Undefined |

## Description

This is a data type for handling information related to one font in general. Both type names (`ScePvf_t_fontInfo` and `ScePvfTFontInfo`) can be used in the same way.

*maxIGlyphMetrics* and *maxFGlyphMetrics* indicate maximum values over a typical range for a given character set. Note that a font may contain characters with glyphs that exceed these maximum values. *maxIGlyphMetrics* and *maxFGlyphMetrics* contained in this data type should be handled as target values for laying out glyphs in the given font (such as for line spacing). Accurate values for individual characters can be obtained from the `ScePvf_t_charInfo` structure via the `scePvfGetCharInfo()` function.

## See Also

`scePvfGetFontInfo()`, `ScePvf_t_iGlyphMetricsInfo`, `ScePvfTIGlyphMetricsInfo`, `ScePvf_t_fGlyphMetricsInfo`, `ScePvfTFGlyphMetricsInfo`, `ScePvf_t_fontStyleInfo`, `ScePvfTFontStyleInfo`

# ScePvfCacheKey

Data type used as key in font cache system

### Definition

```
#include <font/libpvf.h>
typedef struct ScePvfCacheKey {
        int keyValue0;
        int keyValue1;
        int keyValue2;
        int keyValue3;
        int keyValue4;
        int keyValue5;
        int keyValue6;
        int keyValue7;
        int keyValue8;
} ScePvfCacheKey;
```

### Members

| | |
|---|---|
| *keyValue0* | Comparison key 0 |
| *keyValue1* | Comparison key 1 |
| *keyValue2* | Comparison key 2 |
| *keyValue3* | Comparison key 3 |
| *keyValue4* | Comparison key 4 |
| *keyValue5* | Comparison key 5 |
| *keyValue6* | Comparison key 6 |
| *keyValue7* | Comparison key 7 |
| *keyValue8* | Comparison key 8 |

### Description

This is a data type assigned by libpvf as comparison keys in the font cache system.

When these nine comparison keys are all the same value, libpvf has the font cache system behave so that they are handled as the same data.

### See Also

ScePvf_t_cacheSystemInterface, ScePvfTCacheSystemInterface, ScePvf_t_initRec, ScePvfTInitRec, scePvfNewLib()

# ScePvf_t_iKerningInfo, ScePvfTIKerningInfo

Integer value data type for kerning information

### Definition

```
#include <font/libpvf.h>
typedef struct ScePvf_t_iKerningInfo {
        ScePvf_t_s32 xOffset64;
        ScePvf_t_s32 yOffset64;
}ScePvf_t_iKerningInfo, ScePvfTIKerningInfo;
```

### Members

| | |
|---|---|
| *xOffset64* | Offset in the X direction<br>The units are 1/64 pixel. |
| *yOffset64* | Offset in the Y direction<br>The units are 1/64 pixel. |

### Description

This is a data type for handling kerning values defined by the combination of 2 characters as integer values.

This data type is used as a member variable of the `ScePvf_t_kerningInfo` and `ScePvfTKerningInfo` types.

### See Also

`ScePvf_t_fKerningInfo`, `ScePvfTFKerningInfo`,
`scePvfGetKerningInfo()`, `scePvfOpenUserFile()`, `scePvfOpenUserMemory()`,
`scePvfOpenUserFileWithSubfontIndex()`, `scePvfOpenUserMemoryWithSubfontIndex()`

# ScePvf_t_fKerningInfo, ScePvfTFKerningInfo

Floating-point value data type for kerning information

### Definition

```
#include <font/libpvf.h>
typedef struct ScePvf_t_fKerningInfo {
        ScePvf_t_s32 xOffset;
        ScePvf_t_s32 yOffset;
}ScePvf_t_fKerningInfo, ScePvfTFKerningInfo;
```

### Members

| | |
|---|---|
| *xOffset* | Offset in the X direction<br>The units are pixels. |
| *yOffset* | Offset in the Y direction<br>The units are pixels. |

### Description

This is a data type for handling kerning values defined by the combination of 2 characters as floating-point values.

This data type is used as a member variable of the `ScePvf_t_kerningInfo` and `ScePvfTKerningInfo` types.

### See Also

`ScePvf_t_iKerningInfo`, `ScePvfTIKerningInfo`,
`scePvfGetKerningInfo()`, `scePvfOpenUserFile()`, `scePvfOpenUserMemory()`

# ScePvf_t_kerningInfo, ScePvfTKerningInfo

Data type for kerning information

### Definition

```
#include <font/libpvf.h>
typedef struct ScePvf_t_kerningInfo {
        ScePvf_t_iKerningInfo iKerningInfo;
        ScePvf_t_fKerningInfo fKerningInfo;
}ScePvf_t_kerningInfo, ScePvfTKerningInfo;
```

### Members

| | |
|---|---|
| *iKerningInfo* | Integer type kerning information |
| *fKerningInfo* | Floating-point type kerning information |

### Description

This is a data type for handling kerning values defined by the combination of 2 characters.

Kerning values are stored to this data type variable by calling the `scePvfGetKerningInfo()` function.

The same value will be stored in *iKerningInfo* and *fKerningInfo* in their respective types.

### See Also

`ScePvf_t_iKerningInfo`, `ScePvfTIKerningInfo`,
`ScePvf_t_fKerningInfo`, `ScePvfTFKerningInfo`,
`scePvfGetKerningInfo()`, `scePvfOpenUserFile()`, `scePvfOpenUserMemory()`,
`scePvfOpenUserFileWithSubfontIndex()`, `scePvfOpenUserMemoryWithSubfontIndex()`

# Functions

# scePvfNewLib

Generate libpvf library instance

### Definition

```
#include <font/libpvf.h>
ScePvf_t_libId libID = scePvfNewLib(
        ScePvf_t_initRec *initParam,
        ScePvf_t_error *errorCode
);
```

### Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

### Arguments

| | |
|---|---|
| *initParam* | Parameters such as pointer to callback function to be used |
| *errorCode* | Address for storing error code |

### Return Values

If the function completes normally, a pointer to the library instance is returned in *libID*.

If an error occurs, NULL is returned in *libID*, and either SCE_OK, SCE_PVF_ERROR_NOMEMORY, or SCE_PVF_ERROR_ARG is stored in **errorCode*.

### Description

This function generates a libpvf library instance.

Multiple library instances can be generated at the same time.

### Examples

```
static ScePvf_t_pointer cb_Alloc (
        ScePvf_t_pointer pMyData,
        ScePvf_t_u32 size
);
static ScePvf_t_pointer cb_Realloc (
        ScePvf_t_pointer pMyData,
        ScePvf_t_pointer old_p,
        ScePvf_t_u32 size
);
static void cb_Free (
        ScePvf_t_pointer pMyData,
        ScePvf_t_pointer p
);
ScePvf_t_error errorCode;
ScePvf_t_libId libID;
ScePvf_t_initRec initParam = {
        NULL, /* Pointer to user data */
        4, /* Maximum number of fonts that are open simultaneously */
        NULL, /* Handle for cache instance */
        0, /* Reserved area (must be 0) */
```

```
        cb_Alloc, /* Memory allocation function */
        cb_Realloc, /* Memory reallocation function */
        cb_Free,     /* Memory release function */
};

/* Generate a libpvf library instance */
libID = scePvfNewLib (&initParam, &errorCode);
if ( errorCode != SCE_OK ) {
        printf ("Error (scePvfNewLib): 0x%8.8x\n", (int)errorCode);
}
```

### Notes

Several libpvf functions receive this *libID* in the first argument.

### See Also

ScePvf_t_libId, ScePvf_t_initRec, ScePvfTInitRec, ScePvf_t_error, scePvfDoneLib()

# scePvfDoneLib

Destroy libpvf library instance

### Definition

```
#include <font/libpvf.h>
ScePvf_t_error errorCode = scePvfDoneLib(
        ScePvf_t_libId libID
);
```

### Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

### Arguments

| | |
|---|---|
| *libID* | Pointer to library instance |
| *errorCode* | Error code |

### Return Values

If the function completes normally, SCE_OK is returned.

If an error occurs, one of the following is returned.

SCE_PVF_ERROR_NOMEMORY, SCE_PVF_ERROR_LIBID, SCE_PVF_ERROR_ARG,
SCE_PVF_ERROR_NOFILE, SCE_PVF_ERROR_FILEOPEN, SCE_PVF_ERROR_FILECLOSE,
SCE_PVF_ERROR_FILEREAD, SCE_PVF_ERROR_FILESEEK, SCE_PVF_ERROR_TOOMANYOPENED,
SCE_PVF_ERROR_ILLEGALVERSION, SCE_PVF_ERROR_DATAINCONSISTENT,
SCE_PVF_ERROR_EXPIRED, SCE_PVF_ERROR_NOSUPPORT, SCE_PVF_ERROR_UNKNOWN

### Description

This function forcibly closes all fonts that remain open in relation to the one library instance specified by *libID*, releases all memory that had been allocated, and terminates that library instance.

### Examples

```
ScePvf_t_error errorCode;

errorCode = scePvfDoneLib (libID);
if ( errorCode != SCE_OK ) {
        printf ("Error (scePvfDoneLib): 0x%8.8x\n", (int)errorCode);
}
```

### See Also

ScePvf_t_libId, ScePvf_t_error, scePvfNewLib()

# scePvfSetEM

Set em square value

### Definition

```
#include <font/libpvf.h>
ScePvf_t_error errorCode = scePvfSetEM(
        ScePvf_t_libId libID,
        ScePvf_t_f32 emValue
);
```

### Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

### Arguments

| | |
|---|---|
| *libID* | Pointer to library instance |
| *emValue* | Em square value |

### Return Values

If the function completes normally, SCE_OK is returned.

If an error occurs, one of the following is returned.

SCE_PVF_ERROR_LIBID, SCE_PVF_ERROR_ARG

### Description

This function sets the em square value that libpvf uses as the point of reference for the metrics information.

If the em square value is set as 72.0/ (10.125 * 128.0), the numerical value of the metrics information will be a value compatible with the PSP™-compatible grayscale dot font handled by libpgf.

### Examples

```
ScePvf_t_error errorCode;

errorCode = scePvfSetEM
        (libID, (ScePvf_t_f32)(72.0f / (10.125f * 128.0f));
if ( errorCode != SCE_OK ) {
        printf ("Error (scePvfSetEM): 0x%8.8x\n", (int)errorCode);
}
```

### Notes

This em square value will affect all processes of the libpvf instance after this function call.

### See Also

ScePvf_t_libId, ScePvf_t_error

# scePvfSetResolution

Set expected resolution

### Definition

```
#include <font/libpvf.h>
ScePvf_t_error errorCode = scePvfSetResolution(
        ScePvf_t_libId libID,
        ScePvf_t_f32 hResolution,
        ScePvf_t_f32 vResolution
);
```

### Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

### Arguments

| | |
|---|---|
| *libID* | Library instance pointer |
| *hResolution* | Horizontal resolution value (dpi value) |
| *vResolution* | Vertical resolution value (dpi value) |

### Return Values

If the function completes normally, SCE_OK is returned.

If an error occurs, one of the following is returned.

SCE_PVF_ERROR_LIBID, SCE_PVF_ERROR_ARG

### Description

The resolution values set with this function will be used as variables to associate point values with pixel values, both of which are used by libpvf.

### Examples

```
ScePvf_t_error errorCode;

errorCode = scePvfSetResolution
        (libID, (ScePvf_t_f32)128.0f, (ScePvf_t_f32)128.0f);
if ( errorCode != SCE_OK ) {
        printf ("Error (scePvfSetResolution): 0x%8.8x\n", (int)errorCode);
}
```

### Notes

These resolution values will affect all processes of the libpvf instance after this function call.

### See Also

ScePvf_t_libId, ScePvf_t_error

# scePvfGetNumFontList

Get number of fonts installed on PlayStation®Vita

### Definition

```
#include <font/libpvf.h>
ScePvf_t_int numFontLists = scePvfGetNumFontList(
        ScePvf_t_libId libID,
        ScePvf_t_error *errorCode
);
```

### Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

### Arguments

| | |
|---|---|
| *libID* | Library instance pointer |
| *errorCode* | Address for storing the error code |

### Return Values

If the function completes normally, the number of fonts that are installed on the PlayStation®Vita is returned.

If an error occurs, 0 is returned in *numFontLists* and one of the following is stored in **errorCode*.

SCE_PVF_ERROR_NOMEMORY, SCE_PVF_ERROR_LIBID, SCE_PVF_ERROR_ARG,
SCE_PVF_ERROR_NOSUPPORT, SCE_PVF_ERROR_UNKNOWN

### Description

This function returns the number of vector fonts installed on the PlayStation®Vita system.

### Examples

```
ScePvf_t_error errorCode;
int numFontList;

/* Get number of font lists */
numFontList = scePvfGetNumFontList (libID, &errorCode);
if ( errorCode != SCE_OK ) {
        printf ("Error (scePvfGetNumFontList): 0x%8.8x\n", (int)errorCode);
}
```

### See Also

ScePvf_t_libId, ScePvf_t_error, scePvfGetFontList()

# scePvfGetFontList

Create list related to fonts installed on PlayStation®Vita

### Definition

```
#include <font/libpvf.h>
ScePvf_t_error errorCode = scePvfGetFontList(
        ScePvf_t_libId libID,
        ScePvf_t_fontStyleInfo *fontStyleInfo,
        ScePvf_t_int arraySize
);
```

### Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

### Arguments

| | |
|---|---|
| *libID* | Library instance pointer |
| *fontStyleInfo* | Address of font style information array |
| *arraySize* | Size of *fontStyleInfo* array |

### Return Values

If the function completes normally, SCE_OK is returned.

If an error occurs, one of the following is returned.

SCE_PVF_ERROR_NOMEMORY, SCE_PVF_ERROR_LIBID, SCE_PVF_ERROR_ARG, SCE_PVF_ERROR_NOFILE, SCE_PVF_ERROR_FILEOPEN, SCE_PVF_ERROR_FILECLOSE, SCE_PVF_ERROR_FILEREAD, SCE_PVF_ERROR_FILESEEK, SCE_PVF_ERROR_TOOMANYOPENED, SCE_PVF_ERROR_ILLEGALVERSION, SCE_PVF_ERROR_DATAINCONSISTENT, SCE_PVF_ERROR_EXPIRED, SCE_PVF_ERROR_NOSUPPORT, SCE_PVF_ERROR_UNKNOWN

### Description

This function obtains information about vector fonts installed on the PlayStation®Vita system

### Examples

```
ScePvf_t_error errorCode;
ScePvf_t_fontStyleInfo aFontStyleInfoList [30];

/* Get font list */
errorCode = scePvfGetFontList
        (libID, aFontStyleInfoList,
        sizeof (aFontStyleInfoList ) / sizeof (aFontStyleInfoList  [0]));
if ( errorCode != SCE_OK ) {
        printf ("Error (scePvfGetFontList): 0x%8.8x\n", (int)errorCode);
}
```

### Notes

If the number of vector fonts that are installed on the PlayStation®Vita system is greater than *arraySize*, *arraySize* pieces of information from the start of the fonts managed by the PlayStation®Vita system are stored in *fontStyleInfo*.

### See Also

`ScePvf_t_libId`, `ScePvf_t_fontStyleInfo`, `ScePvfTFontStyleInfo`, `ScePvf_t_error`, `scePvfGetNumFontList()`

# scePvfFindOptimumFont

Find optimum font

### Definition

```
#include <font/libpvf.h>
ScePvf_t_fontIndex fontIndex = scePvfFindOptimumFont(
        ScePvf_t_libId libID,
        ScePvf_t_fontStyleInfo *fontStyleInfo,
        ScePvf_t_error *errorCode
);
```

### Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

### Arguments

| | |
|---|---|
| *libID* | Library instance pointer |
| *fontStyleInfo* | Style information of font to be obtained |
| *errorCode* | Address for storing error code |

### Return Values

If the function completes normally, the index value of the optimum font is returned.

If an error occurs, 0 is returned in *fontIndex* and one of the following is stored in **errorCode*.

SCE_PVF_ERROR_NOMEMORY, SCE_PVF_ERROR_LIBID, SCE_PVF_ERROR_ARG,
SCE_PVF_ERROR_NOFILE, SCE_PVF_ERROR_FILEOPEN, SCE_PVF_ERROR_FILECLOSE,
SCE_PVF_ERROR_FILEREAD, SCE_PVF_ERROR_FILESEEK, SCE_PVF_ERROR_TOOMANYOPENED,
SCE_PVF_ERROR_ILLEGALVERSION, SCE_PVF_ERROR_DATAINCONSISTENT,
SCE_PVF_ERROR_EXPIRED, SCE_PVF_ERROR_NOSUPPORT, SCE_PVF_ERROR_UNKNOWN

### Description

This function finds the nearest vector font to the font style set in *fontStyleInfo* from the vector fonts that are installed on the PlayStation®Vita and returns a value for accessing that font in *fontIndex*.

### Examples

```
ScePvf_t_error errorCode;
ScePvf_t_fontStyleInfo targetStyle;
ScePvf_t_fontIndex targetFontIndex;

memset (&targetStyle, 0, sizeof(targetStyle));
targetStyle.languageCode = SCE_PVF_LANGUAGE_J;
targetStyle.familyCode = SCE_PVF_DEFAULT_FAMILY_CODE;
targetStyle.style = SCE_PVF_DEFAULT_STYLE_CODE;
targetFontIndex
        = scePvfFindOptimumFont (libID, &targetStyle, &errorCode);
if ( errorCode != SCE_OK ) {
        printf ("Error (scePvfFindOptimumFont): 0x%8.8x\n", (int)errorCode);
}
```

### Notes

This function finds the font determined to be closest based on an internal decision criterion that libpvf has.

### See Also

ScePvf_t_libId, ScePvf_t_error, scePvfFindFont()

# scePvfFindFont

Find font

### Definition

```
#include <font/libpvf.h>
ScePvf_t_fontIndex fontIndex = scePvfFindFont(
        ScePvf_t_libId libID,
        ScePvf_t_fontStyleInfo *fontStyleInfo,
        ScePvf_t_error *errorCode
);
```

### Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

### Arguments

| | |
|---|---|
| *libID* | Library instance pointer |
| *fontStyleInfo* | Style information about the font to be obtained |
| *errorCode* | Address for storing error code |

### Return Values

If the function completes normally, the index value of the optimum font is returned.

If an error occurs, 0 is returned in *fontIndex* and one of the following is stored in **errorCode*.

SCE_PVF_ERROR_NOMEMORY, SCE_PVF_ERROR_LIBID, SCE_PVF_ERROR_ARG,
SCE_PVF_ERROR_NOFILE, SCE_PVF_ERROR_FILEOPEN, SCE_PVF_ERROR_FILECLOSE,
SCE_PVF_ERROR_FILEREAD, SCE_PVF_ERROR_FILESEEK, SCE_PVF_ERROR_TOOMANYOPENED,
SCE_PVF_ERROR_ILLEGALVERSION, SCE_PVF_ERROR_DATAINCONSISTENT,
SCE_PVF_ERROR_EXPIRED, SCE_PVF_ERROR_NOSUPPORT, SCE_PVF_ERROR_UNKNOWN

### Description

This function finds the vector font that conforms to the font style that was set in *fontStyleInfo* from the vector fonts that are installed on the PlayStation®Vita and returns a value for accessing that font in *fontIndex*.

If no font conforms, -1 is returned in *fontIndex*.

### Examples

```
ScePvf_t_error errorCode;
ScePvf_t_fontStyleInfo targetStyle;
ScePvf_t_fontIndex targetFontIndex;

memset (&targetStyle, 0, sizeof(targetStyle));
targetStyle.languageCode = SCE_PVF_LANGUAGE_J;
targetStyle.familyCode = SCE_PVF_DEFAULT_FAMILY_CODE;
targetStyle.style = SCE_PVF_DEFAULT_STYLE_CODE;
targetFontIndex = scePvfFindFont (libID, &targetStyle, &errorCode);
if ( errorCode != SCE_OK ) {
        printf ("Error (scePvfFindFont): 0x%8.8x\n", (int)errorCode);
}
```

### Notes

This function finds the font for which all *fontStyleInfo* information matches.

### See Also

ScePvf_t_libId, ScePvf_t_error, scePvfFindOptimumFont()

# scePvfGetFontInfoByIndexNumber

Get font information (by specify font by number)

### Definition

```
#include <font/libpvf.h>
ScePvf_t_error errorCode = scePvfGetFontInfoByIndexNumber(
        ScePvf_t_libId libID,
        ScePvf_t_fontStyleInfo *fontStyleInfo,
        ScePvf_t_fontIndex fontIndex
);
```

### Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

### Arguments

| | |
|---|---|
| *libID* | Library instance pointer |
| *fontStyleInfo* | Pointer to area for storing font style information |
| *fontIndex* | Font index number |

### Return Values

If the function completes normally, SCE_OK is returned.

If an error occurs, one of the following is returned.

SCE_PVF_ERROR_NOMEMORY, SCE_PVF_ERROR_LIBID, SCE_PVF_ERROR_ARG,
SCE_PVF_ERROR_NOFILE, SCE_PVF_ERROR_FILEOPEN, SCE_PVF_ERROR_FILECLOSE,
SCE_PVF_ERROR_FILEREAD, SCE_PVF_ERROR_FILESEEK, SCE_PVF_ERROR_TOOMANYOPENED,
SCE_PVF_ERROR_ILLEGALVERSION, SCE_PVF_ERROR_DATAINCONSISTENT,
SCE_PVF_ERROR_EXPIRED, SCE_PVF_ERROR_NOSUPPORT, SCE_PVF_ERROR_UNKNOWN

### Description

This function gets the font style information of the vector font that corresponds to the font index number that was returned by scePvfFindOptimumFont() or scePvfFindFont().

### Examples

```
ScePvf_t_error errorCode;
ScePvf_t_fontStyleInfo fontStyleInfo;

errorCode = scePvfGetFontInfoByIndexNumber
        (libID, &fontStyleInfo, (ScePvf_t_fontIndex)0);
if ( errorCode != SCE_OK ) {
        printf ("Error (scePvfGetFontInfoByIndexNumber): 0x%8.8x\n",
                (int)errorCode);
}
```

### Notes

When 0 is specified for *fontIndex*, the font style information of the PlayStation®Vita system default font is stored in *fontStyleInfo*.

### See Also

`ScePvf_t_libId`, `ScePvf_t_error`, `scePvfFindOptimumFont()`, `scePvfFindFont()`

# scePvfOpen

Open font (that PlayStation®Vita system has internally)

### Definition

```
#include <font/libpvf.h>
ScePvf_t_fontId fontID = scePvfOpen(
        ScePvf_t_libId libID,
        ScePvf_t_fontIndex fontIndex,
        ScePvf_t_u32 mode,
        ScePvf_t_error *errorCode
);
```

### Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

### Arguments

| | |
|---|---|
| *libID* | Library instance pointer |
| *fontIndex* | Font index number |
| *mode* | Access mode<br>One of the following values is assigned.<br>SCE_PVF_FILEBASEDSTREAM, SCE_PVF_MEMORYBASEDSTREAM |
| *errorCode* | Address for storing the error code |

### Return Values

If the function completes normally, an ID for accessing the vector font is returned.

If an error occurs, NULL is returned in *fontID* and one of the following is stored in **errorCode*.

SCE_PVF_ERROR_NOMEMORY, SCE_PVF_ERROR_LIBID, SCE_PVF_ERROR_ARG,
SCE_PVF_ERROR_NOFILE, SCE_PVF_ERROR_FILEOPEN, SCE_PVF_ERROR_FILECLOSE,
SCE_PVF_ERROR_FILEREAD, SCE_PVF_ERROR_FILESEEK, SCE_PVF_ERROR_TOOMANYOPENED,
SCE_PVF_ERROR_ILLEGALVERSION, SCE_PVF_ERROR_DATAINCONSISTENT,
SCE_PVF_ERROR_EXPIRED, SCE_PVF_ERROR_NOSUPPORT, SCE_PVF_ERROR_UNKNOWN

### Description

This function opens the vector font corresponding to the font index number that was returned by scePvfFindOptimumFont() or scePvfFindFont().

### Examples

```
ScePvf_t_error errorCode;
ScePvf_t_fontId fontID;

fontID = scePvfOpen
        (libID, targetFontIndex, SCE_PVF_FILEBASEDSTREAM, &errorCode);
if ( errorCode != SCE_OK ) {
        printf ("Error (scePvfOpen)(%d): 0x%8.8x\n", i, (int)errorCode);
}
```

### Notes

If 0 is specified for *fontIndex*, the PlayStation®Vita system vector font is opened.

### See Also

`ScePvf_t_initRec`, `ScePvfTInitRec`, `ScePvf_t_libId`, `ScePvf_t_fontId`, `ScePvf_t_error`, `scePvfFindOptimumFont()`, `scePvfFindFont()`, `scePvfOpenUserFile()`, `scePvfOpenUserMemory()`, `scePvfOpenUserFileWithSubfontIndex()`, `scePvfOpenUserMemoryWithSubfontIndex()`, `scePvfClose()`

# scePvfOpenUserFile

Open font (by specifying filename)

### Definition

```
#include <font/libpvf.h>
ScePvf_t_fontId fontID = scePvfOpenUserFile(
        ScePvf_t_libId libID,
        ScePvf_t_pointer filename,
        ScePvf_t_u32 mode,
        ScePvf_t_error *errorCode
);
```

### Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

### Arguments

| | |
|---|---|
| *libID* | Library instance pointer |
| *filename* | Vector font file name |
| *mode* | Access mode<br>One of the following values is assigned.<br>SCE_PVF_FILEBASEDSTREAM, SCE_PVF_MEMORYBASEDSTREAM |
| *errorCode* | Address for storing the error code |

### Return Values

If the function completes normally, an ID for accessing the vector font is returned.

If an error occurs, NULL is returned in *fontID* and one of the following is stored in **errorCode*.

SCE_PVF_ERROR_NOMEMORY, SCE_PVF_ERROR_LIBID, SCE_PVF_ERROR_ARG,
SCE_PVF_ERROR_NOFILE, SCE_PVF_ERROR_FILEOPEN, SCE_PVF_ERROR_FILECLOSE,
SCE_PVF_ERROR_FILEREAD, SCE_PVF_ERROR_FILESEEK, SCE_PVF_ERROR_TOOMANYOPENED,
SCE_PVF_ERROR_ILLEGALVERSION, SCE_PVF_ERROR_DATAINCONSISTENT,
SCE_PVF_ERROR_EXPIRED, SCE_PVF_ERROR_NOSUPPORT, SCE_PVF_ERROR_UNKNOWN

### Description

This function opens a TrueType or OpenType vector font file specified by *filename*.

### Examples

```
ScePvf_t_error errorCode;
ScePvf_t_fontId fontID;

fontID = scePvfOpenUserFile
        (libID,
         "YOUR_TTF_FILE.TTF",
         SCE_PVF_FILEBASEDSTREAM, &errorCode);
if ( errorCode != SCE_OK ) {
        printf ("Error (scePvfOpen)(%d): 0x%8.8x\n", i, (int)errorCode);
}
```

### Notes

Although this function supports most TrueType and OpenType vector fonts, note that it may not be able to open, obtain correct values from, or correctly rasterize some fonts.

### See Also

`ScePvf_t_initRec`, `ScePvfTInitRec`, `ScePvf_t_libId`, `ScePvf_t_fontId`, `ScePvf_t_error`, `scePvfOpen()`, `scePvfOpenUserMemory()`, `scePvfOpenUserFileWithSubfontIndex()`, `scePvfOpenUserMemoryWithSubfontIndex()`, `scePvfClose()`

# scePvfOpenUserMemory

Open font (by specifying memory address)

### Definition

```
#include <font/libpvf.h>
ScePvf_t_fontId fontID = scePvfOpenUserMemory(
        ScePvf_t_libId libID,
        ScePvf_t_pointer addr,
        ScePvf_t_u32 size,
        ScePvf_t_error *errorCode
);
```

### Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

### Arguments

| | |
|---|---|
| *libID* | Library instance pointer |
| *addr* | Address of vector font data that is in memory |
| *size* | Size of vector font data that is in memory |
| *errorCode* | Address for storing the error code |

### Return Values

If the function completes normally, an ID for accessing the vector font is returned.

If an error occurs, NULL is returned in *fontID* and one of the following is stored in **errorCode*.

SCE_PVF_ERROR_NOMEMORY, SCE_PVF_ERROR_LIBID, SCE_PVF_ERROR_ARG,
SCE_PVF_ERROR_TOOMANYOPENED, SCE_PVF_ERROR_ILLEGALVERSION,
SCE_PVF_ERROR_DATAINCONSISTENT, SCE_PVF_ERROR_EXPIRED, SCE_PVF_ERROR_NOSUPPORT,
SCE_PVF_ERROR_UNKNOWN

### Description

This function opens a vector font that was read into memory specified by *addr*.

## Examples

```
ScePvf_t_error errorCode;
ScePvf_t_fontId fontID;
ScePvf_t_pointer pFontData;
ScePvf_t_u32 fontDataSize;

/* For example, function for allocating memory according to file size and reading
file */
fontDataSize = myReadAndAlloc ("YOUR_TTF_FILE.TTF", &pFontData);

if ( 0 < fontDataSize )
        fontID = scePvfOpenUserMemory
              (libID,
               pFontdata,
               fontDataSize, &errorCode);
        if ( errorCode != SCE_OK ) {
              printf ("Error (scePvfOpen)(%d): 0x%8.8x\n", i, (int)errorCode);
        }
}
```

## Notes

Although this function supports most TrueType and OpenType vector fonts, note that it may not be able to open, obtain correct values from, or correctly rasterize some fonts.

The common `scePvfClose()` function is used to close the font.

## See Also

`ScePvf_t_libId`, `ScePvf_t_fontId`, `ScePvf_t_error`, `scePvfOpen()`, `scePvfOpenUserFile()`, `scePvfOpenUserFileWithSubfontIndex()`, `scePvfOpenUserMemoryWithSubfontIndex()`, `scePvfClose()`

# scePvfOpenUserFileWithSubfontIndex

Open font (by specifying filename and sub font index)

### Definition

```
#include <font/libpvf.h>
ScePvf_t_fontId fontID = scePvfOpenUserFileWithSubfontIndex(
        ScePvf_t_libId libID,
        ScePvf_t_pointer filename,
        ScePvf_t_u32 mode,
        ScePvf_t_u32 subfontIndex,
        ScePvf_t_error *errorCode
);
```

### Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

### Arguments

| | |
|---|---|
| *libID* | Library instance pointer |
| *filename* | Vector font file name |
| *mode* | Access mode<br>One of the following values is assigned.<br>SCE_PVF_FILEBASEDSTREAM, SCE_PVF_MEMORYBASEDSTREAM |
| *subfontIndex* | Sub font index value |
| *errorCode* | Address for storing the error code |

### Return Values

If the function completes normally, an ID for accessing the vector font is returned.

If an error occurs, NULL is returned in *fontID* and one of the following is stored in **errorCode*.

SCE_PVF_ERROR_NOMEMORY, SCE_PVF_ERROR_LIBID, SCE_PVF_ERROR_ARG,
SCE_PVF_ERROR_NOFILE, SCE_PVF_ERROR_FILEOPEN, SCE_PVF_ERROR_FILECLOSE,
SCE_PVF_ERROR_FILEREAD, SCE_PVF_ERROR_FILESEEK, SCE_PVF_ERROR_TOOMANYOPENED,
SCE_PVF_ERROR_ILLEGALVERSION, SCE_PVF_ERROR_DATAINCONSISTENT,
SCE_PVF_ERROR_EXPIRED, SCE_PVF_ERROR_NOSUPPORT, SCE_PVF_ERROR_UNKNOWN

### Description

This function opens the vector font file in TrueTypeCollection format specified with *filename* by specifying its sub font index (faceIndex).

### Examples

```
ScePvf_t_error errorCode;
ScePvf_t_fontId fontID;

fontID = scePvfOpenUserFileWithSubfontIndex
         (libID,
          "YOUR_TTC_FILE.TTC",
          SCE_PVF_FILEBASEDSTREAM, 0, &errorCode);
if ( errorCode != SCE_OK ) {
         printf ("Error (scePvfOpen)(%d): 0x%8.8x¥n", i, (int)errorCode);
}
```

### Notes

Although this function supports most TrueTypeCollection vector fonts, note that it may not be able to open, obtain correct values from, or correctly rasterize some fonts.

### See Also

ScePvf_t_initRec, ScePvfTInitRec, ScePvf_t_libId, ScePvf_t_fontId, ScePvf_t_error, scePvfOpen(), scePvfOpenUserFile(), scePvfOpenUserMemory(), scePvfOpenUserMemoryWithSubfontIndex(), scePvfClose()

# scePvfOpenUserMemoryWithSubfontIndex

Open font (by specifying memory address and sub font index)

### Definition

```
#include <font/libpvf.h>
ScePvf_t_fontId fontID = scePvfOpenUserMemoryWithSubfontIndex(
        ScePvf_t_libId libID,
        ScePvf_t_pointer addr,
        ScePvf_t_u32 size,
        ScePvf_t_u32 subfontIndex,
        ScePvf_t_error *errorCode
);
```

### Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

### Arguments

| | |
|---|---|
| *libID* | Library instance pointer |
| *addr* | Address of vector font data that is in memory |
| *size* | Size of vector font data that is in memory |
| *subfontIndex* | Sub font index value |
| *errorCode* | Address for storing the error code |

### Return Values

If the function completes normally, an ID for accessing the vector font is returned.

If an error occurs, NULL is returned in *fontID* and one of the following is stored in **errorCode*.

SCE_PVF_ERROR_NOMEMORY, SCE_PVF_ERROR_LIBID, SCE_PVF_ERROR_ARG,
SCE_PVF_ERROR_TOOMANYOPENED, SCE_PVF_ERROR_ILLEGALVERSION,
SCE_PVF_ERROR_DATAINCONSISTENT, SCE_PVF_ERROR_EXPIRED, SCE_PVF_ERROR_NOSUPPORT,
SCE_PVF_ERROR_UNKNOWN

### Description

This function opens the vector font in TrueTypeCollection format loaded into the memory specified in *addr* by specifying its sub font index (faceIndex).

### Examples

```
ScePvf_t_error errorCode;
ScePvf_t_fontId fontID;
ScePvf_t_pointer pFontData;
ScePvf_t_u32 fontDataSize;

/* For example, function for allocating memory according to file size and reading
file */
fontDataSize = myReadAndAlloc ("YOUR_TTC_FILE.TTC", &pFontData);

if ( 0 < fontDataSize )
        fontID = scePvfOpenUserMemoryWithSubfontIndex
              (libID,
               pFontdata, fontDataSize, 0, &errorCode);
        if ( errorCode != SCE_OK ) {
              printf ("Error (scePvfOpen)(%d): 0x%8.8x¥n", i, (int)errorCode);
        }
}
```

### Notes

Although this function supports most TrueTypeCollection vector fonts, note that it may not be able to open, obtain correct values from, or correctly rasterize some fonts.

### See Also

ScePvf_t_initRec, ScePvfTInitRec, ScePvf_t_libId, ScePvf_t_fontId, ScePvf_t_error, scePvfOpen(), scePvfOpenUserMemory(), scePvfOpenUserFileWithSubfontIndex(), scePvfClose()

# scePvfClose

Close font

### Definition

```
#include <font/libpvf.h>
ScePvf_t_error errorCode = scePvfClose(
        ScePvf_t_fontId fontID
);
```

### Calling Conditions

Cannot be called from an interrupt handler
Can be called from a thread (must be called in an interrupt-enabled state)
Not multithread safe

### Arguments

*fontID* Font ID

### Return Values

If the function completes normally, SCE_OK is returned.
If an error occurs, one of the following is returned.

SCE_PVF_ERROR_NOMEMORY, SCE_PVF_ERROR_LIBID, SCE_PVF_ERROR_ARG,
SCE_PVF_ERROR_FILECLOSE, SCE_PVF_ERROR_DATAINCONSISTENT, SCE_PVF_ERROR_EXPIRED,
SCE_PVF_ERROR_NOSUPPORT, SCE_PVF_ERROR_UNKNOWN

### Description

This function closes a vector font.

### Examples

```
ScePvf_t_error errorCode;

errorCode = scePvfClose (fontID);
if ( errorCode != SCE_OK ) {
        printf ("Error (scePvfClose): 0x%8.8x\n", (int)errorCode);
}
```

### See Also

ScePvf_t_fontId, ScePvf_t_error, scePvfFindOptimumFont(), scePvfFindFont(),
scePvfOpen(), scePvfOpenUserFile(), scePvfOpenUserMemory(),
scePvfOpenUserFileWithSubfontIndex(), scePvfOpenUserMemoryWithSubfontIndex()

# scePvfFlush

Clear libpvf internal local cache

### Definition

```
#include <font/libpvf.h>
ScePvf_t_error errorCode = scePvfFlush(
        ScePvf_t_fontId fontID
);
```

### Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

### Arguments

*fontID* Font ID

### Return Values

When the function completes normally, SCE_OK is returned.

If an error occurs, SCE_PVF_ERROR_ARG is returned.

### Description

This function discards libpvf's internal local and short-term cache data and forcibly releases the memory that libpvf acquired for the cache according to the *allocFunc* member variable of ScePvf_t_initRec.

### Examples

```
ScePvf_t_error errorCode;

errorCode = scePvfFlush (fontID);
if ( errorCode != SCE_OK ) {
        printf ("Error (scePvfDoneLib): 0x%8.8x\n", (int)errorCode);
}
```

### See Also

ScePvf_t_initRec, ScePvfTInitRec

# scePvfSetCharSize

Set size of rasterized character

### Definition

```
#include <font/libpvf.h>
ScePvf_t_error errorCode = scePvfSetCharSize(
        ScePvf_t_fontId fontID,
        ScePvf_t_f32 hSize,
        ScePvf_t_f32 vSize
);
```

### Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

### Arguments

| | |
|---|---|
| *fontID* | Font ID |
| *hSize* | Horizontal size (point value) |
| *vSize* | Vertical size (point value) |

### Return Values

When the function completes normally, SCE_OK is returned.

If an error occurs, SCE_PVF_ERROR_ARG is returned.

### Description

This function sets the size of the rasterized character using point values. Vertical and horizontal sizes can be set independently.

### Examples

```
ScePvf_t_error errorCode;

errorCode = scePvfSetCharSize
        (fontID, (ScePvf_t_f32)10.0f, (ScePvf_t_f32)10.0f);
if ( errorCode != SCE_OK ) {
        printf ("Error (scePvfSetCharSize): 0x%8.8x\n", (int)errorCode);
}
```

### See Also

ScePvf_t_fontId, ScePvf_t_error, scePvfSetResolution()

# scePvfSetEmboldenRate

Set embolden (or unembolden) rate of rasterized character

### Definition

```
#include <font/libpvf.h>
ScePvf_t_error errorCode = scePvfSetEmboldenRate(
        ScePvf_t_fontId fontID,
        ScePvf_t_f32 emboldenRate
);
```

### Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

### Arguments

| | |
|---|---|
| *fontID* | Font ID |
| *emboldenRate* | Embolden rate value |

### Return Values

When the function completes normally, SCE_OK is returned.

If an error occurs, SCE_PVF_ERROR_ARG is returned.

### Description

This function sets the embolden size of the rasterized character. For *emboldenRate*, set a negative value to unembolden the rasterized character, and a positive value to embolden it. The specified value will be used as the percentage value with *hSize* (set with scePvfSetCharSize()) as the point of reference.

The value range is:

SCE_PVF_MIN_EMBOLDEN_RATE ≦ *emboldenRate* ≦ SCE_PVF_MAX_EMBOLDEN_RATE

### Examples

```
ScePvf_t_error errorCode;

errorCode = scePvfSetEmboldenRate
        (fontID, (ScePvf_t_f32)5.0f);
if ( errorCode != SCE_OK ) {
        printf ("Error (scePvfSetEmboldenRate): 0x%8.8x\n", (int)errorCode);
}
```

### Notes

Values provided by this function do not affect values used by ScePvf_t_iGlyphMetricsInfo.

Noise may result from rasterizing characters with complex glyph shapes, characters with glyphs designed using very thick lines, or characters with glyphs designed using very thin lines.

**See Also**

`ScePvf_t_fontId`, `ScePvf_t_error`, `scePvfSetResolution()`, `scePvfSetCharSize()`

# scePvfSetSkewValue

Set skew value of rasterized character

## Definition

```
#include <font/libpvf.h>
ScePvf_t_error errorCode = scePvfSetSkewValue(
        ScePvf_t_fontId fontID,
        ScePvf_t_f32 angleX,
        ScePvf_t_f32 angleY
);
```

## Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

## Arguments

| | |
|---|---|
| *fontID* | Font ID |
| *angleX* | Angle of x-axis defined by horizontal line (unit: degrees) |
| *angleY* | Angle of y-axis defined by vertical line (unit: degrees) |

## Return Values

When the function completes normally, SCE_OK is returned.

If an error occurs, SCE_PVF_ERROR_ARG is returned.

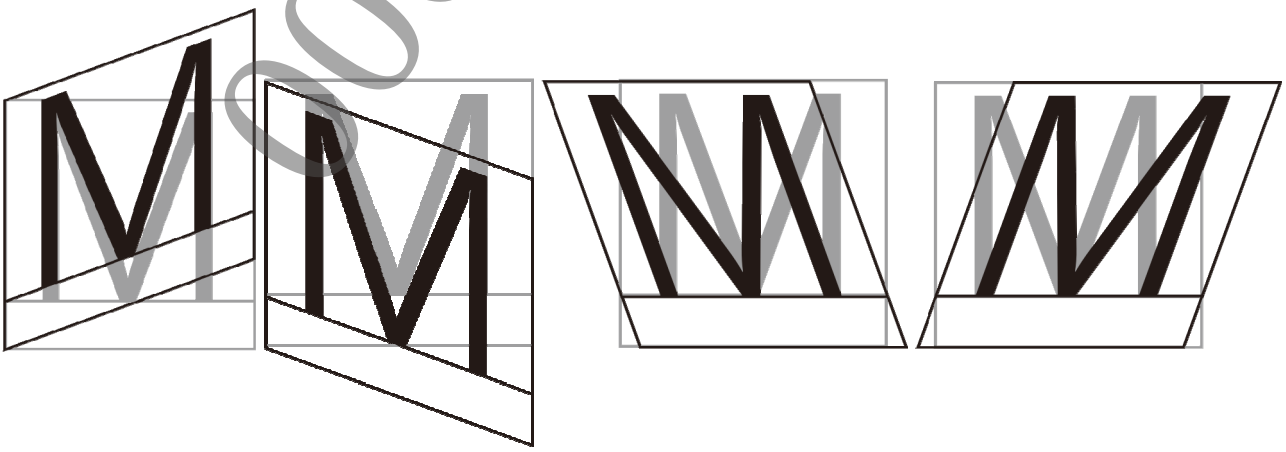
## Description

This function sets the character slant. A positive value indicates a clock-wise direction, and the value range is:

SCE_PVF_MIN_SKEW_VALUE ≦ *angleX* ≦ SCE_PVF_MAX_SKEW_VALUE

SCE_PVF_MIN_SKEW_VALUE ≦ *angleY* ≦ SCE_PVF_MAX_SKEW_VALUE

The following four diagrams show sample results for when *angleX* is a negative value, when *angleX* is a positive value, when *angleY* is a negative value, and when *angleY* is a positive value.



Values for both *angleX* and *angleY* can be set at the same time.

The following is an example of when *angleX* is a negative value and *angleY* is a positive value.

## Examples

```
ScePvf_t_error errorCode;

errorCode = scePvfSetSkewValue
          (fontID, (ScePvf_t_f32)-20.0f, (ScePvf_t_f32)20.0f);
if ( errorCode != SCE_OK ) {
         printf ("Error (scePvfSetSkewValue): 0x%8.8x\n", (int)errorCode);
}
```

## Notes

Values provided by this function do not affect values used by `ScePvf_t_iGlyphMetricsInfo`.

## See Also

`ScePvf_t_fontId`, `ScePvf_t_error`

# scePvfIsElement

Check whether or not glyph exists

### Definition

```
#include <font/libpvf.h>
ScePvf_t_bool result = scePvfIsElement(
        ScePvf_t_fontId fontID,
        ScePvf_t_charCode charCode
);
```

### Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

### Arguments

| | |
|---|---|
| *fontID* | Font ID |
| *charCode* | UCS2 character code |

### Return Values

If the character glyph specified in *charCode* exists in the vector font specified in *fontID*, SCE_PVF_TRUE is returned.

If the glyph does not exist, or if an error occurs, SCE_PVF_FALSE is returned.

### Description

This function checks whether or not the character glyph specified in *charCode* exists in the vector font specified in *fontID*, and returns the result.

### Examples

```
ScePvf_t_bool result;

result = scePvfIsElement
         (fontID, (ScePvf_t_charCode)0x0041);
printf ("Result (scePvfIsElement): %d\n", (int)result);
```

### Notes

Different glyphs are recorded for Japanese fonts, Latin fonts, Korean fonts, and Chinese fonts. Values of this function can be used to check for a font that includes the desired character code glyph at runtime.

### See Also

ScePvf_t_fontId, ScePvf_t_error, scePvfIsVertElement()

# scePvfIsVertElement

Check whether or not vertical layout glyph exists

### Definition

```
#include <font/libpvf.h>
ScePvf_t_bool result = scePvfIsVertElement(
        ScePvf_t_fontId fontID,
        ScePvf_t_charCode charCode
);
```

### Calling Conditions

Cannot be called from an interrupt handler.

Can be called from a thread (must be called in an interrupt-enabled state).

Not multithread safe.

### Arguments

| | |
|---|---|
| *fontID* | Font ID |
| *charCode* | UCS2 character code |

### Return Values

Returns SCE_PVF_TRUE when a vertical layout glyph of the characters specified with *charCode* exists in the vector font specified with *fontID*.

Returns SCE_PVF_FALSE when a vertical layout glyph does not exist or for errors.

### Description

This function checks if a vertical layout glyph for the characters specified with *charCode* is included in the vector font specified with *fontID* and returns the result.

### Examples

```
ScePvf_t_bool result;

result = scePvfIsVertElement
         (fontID, (ScePvf_t_charCode)0x3001);
printf ("Result (scePvfIsVertElement): %d¥n", (int)result);
```

### Notes

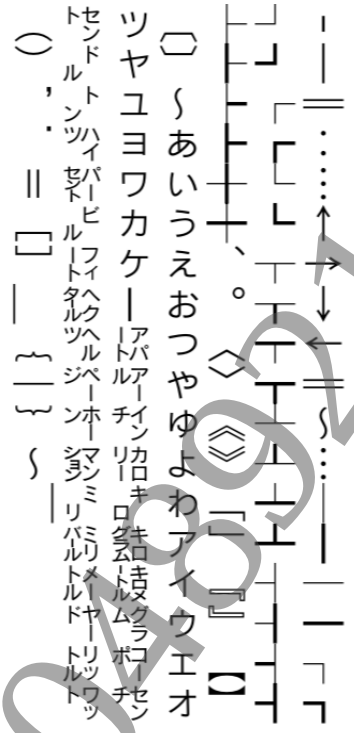
Among the fonts internal to the PlayStation®Vita system, only the Japanese fonts have vertical layout glyphs recorded. The Latin fonts, Korean fonts, and Chinese fonts do not have vertical layout glyphs recorded.

It is also possible to check if a vertical layout glyph exists or not for a user font that has been opened with each of the functions scePvfOpenUserFile(), scePvfOpenUserMemory(), scePvfOpenUserFileWithSubfontIndex(), and scePvfOpenUserMemoryWithSubfontIndex().

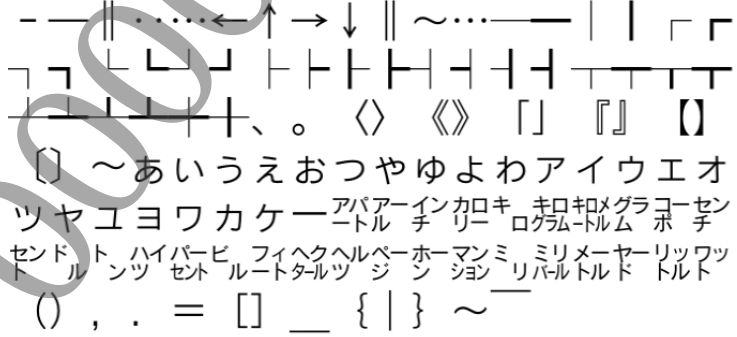
The UCS2 character codes for the vertical layout glyphs recorded in the Japanese fonts internal to the PlayStation®Vita system are as follows (121 glyphs).

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| U+2010 | U+2015 | U+2016 | U+2025 | U+2026 | U+2190 | U+2191 | U+2192 | U+2193 | U+2225 | U+223C |
| U+22EF | U+2500 | U+2501 | U+2502 | U+2503 | U+250C | U+250F | U+2510 | U+2513 | U+2514 | U+2517 |
| U+2518 | U+251B | U+251C | U+251D | U+2520 | U+2523 | U+2524 | U+2525 | U+2528 | U+252B | U+252C |
| U+252F | U+2530 | U+2533 | U+2534 | U+2537 | U+2538 | U+253B | U+253F | U+2542 | U+3001 | U+3002 |
| U+3008 | U+3009 | U+300A | U+300B | U+300C | U+300D | U+300E | U+300F | U+3010 | U+3011 | U+3014 |
| U+3015 | U+301C | U+3041 | U+3043 | U+3045 | U+3047 | U+3049 | U+3063 | U+3083 | U+3085 | U+3087 |
| U+308E | U+30A1 | U+30A3 | U+30A5 | U+30A7 | U+30A9 | U+30C3 | U+30E3 | U+30E5 | U+30E7 | U+30EE |
| U+30F5 | U+30F6 | U+30FC | U+3300 | U+3303 | U+3305 | U+330D | U+3314 | U+3315 | U+3316 | U+3318 |
| U+331E | U+3322 | U+3323 | U+3326 | U+3327 | U+332A | U+332B | U+3331 | U+3333 | U+3336 | U+3339 |
| U+333B | U+3342 | U+3347 | U+3349 | U+334A | U+334D | U+334E | U+3351 | U+3357 | U+FF08 | U+FF09 |
| U+FF0C | U+FF0E | U+FF1D | U+FF3B | U+FF3D | U+FF3F | U+FF5B | U+FF5C | U+FF5D | U+FF5E | U+FFE3 |

List of vertical layout glyphs (taken from the output of the sample program "`vertical`")



List of the horizontal layout glyphs for the same set of characters (standard glyphs) (taken from the output of the sample program "`vertical`")



## See Also

`ScePvf_t_fontId`, `ScePvf_t_error`, `scePvfIsElement()`

# scePvfGetFontInfo

Get font information

### Definition

```
#include <font/libpvf.h>
ScePvf_t_error errorCode = scePvfGetFontInfo(
        ScePvf_t_fontId fontID,
        ScePvf_t_fontInfo *fontInfo
);
```

### Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

### Arguments

| | |
|---|---|
| *fontID* | Font ID |
| *fontInfo* | Pointer to area for storing font information |

### Return Values

If the function completes normally, SCE_OK is returned.

If an error occurs, one of the following is returned.

SCE_PVF_ERROR_ARG, SCE_PVF_ERROR_DATAINCONSISTENT

### Description

This function stores information related to the open font indicated by *fontID* in *fontInfo*.

### Examples

```
ScePvf_t_error errorCode;
ScePvf_t_fontInfo fontInfo;

errorCode = scePvfGetFontInfo (fontID, &fontInfo);
if ( errorCode != SCE_OK ) {
        printf ("Error (scePvfGetFontInfo): 0x%8.8x\n", (int)errorCode);
}
```

### Notes

The following values related to a font that was opened by `scePvfOpenUserFile()`, `scePvfOpenUserMemory()`, `scePvfOpenUserFileWithSubfontIndex()`, or `scePvfOpenUserMemoryWithSubfontIndex()` cannot be obtained.

*fontInfo*->*fontStyleInfo.weight*
*fontInfo*->*fontStyleInfo.familyCode*
*fontInfo*->*fontStyleInfo.style*
*fontInfo*->*fontStyleInfo.subStyle*
*fontInfo*->*fontStyleInfo.languageCode*
*fontInfo*->*fontStyleInfo.regionCode*
*fontInfo*->*fontStyleInfo.countryCode*
*fontInfo*->*fontStyleInfo.fileName*
*fontInfo*->*fontStyleInfo.extraAttributes*
*fontInfo*->*fontStyleInfo.expireDate*

### See Also

`ScePvf_t_fontId`, `ScePvf_t_fontInfo`, `ScePvfTFontInfo`, `ScePvf_t_error`, `scePvfOpen()`

# scePvfGetCharInfo

Get character information

### Definition

```
#include <font/libpvf.h>
ScePvf_t_error errorCode = scePvfGetCharInfo(
        ScePvf_t_fontId fontID,
        ScePvf_t_charCode charCode,
        ScePvf_t_charInfo *charInfo
);
```

### Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

### Arguments

| | |
|---|---|
| *fontID* | Font ID |
| *charCode* | UCS2 character code |
| *charInfo* | Character information |

### Return Values

If the function completes normally, SCE_OK is returned.

If an error occurs, one of the following is returned.

SCE_PVF_ERROR_NOMEMORY, SCE_PVF_ERROR_ARG, SCE_PVF_ERROR_NOGLYPH

### Description

This function stores character information related to the character code indicated by *charCode* of the open font indicated by *fontID* in *charInfo*.

### Examples

```
ScePvf_t_error errorCode;
ScePvf_t_charInfo charInfo;
ScePvf_t_charCode charCode = 0x0041;

errorCode = scePvfGetCharInfo (fontID, charCode, &charInfo);
if ( errorCode != SCE_OK ) {
        printf ("Error (scePvfGetCharInfo): 0x%8.8x\n", (int)errorCode);
}
```

### See Also

ScePvf_t_fontId, ScePvf_t_charCode, ScePvf_t_charInfo, ScePvfTCharInfo, ScePvf_t_error, scePvfOpen(), scePvfGetVertCharInfo()

# scePvfGetVertCharInfo

Get vertical layout glyph character information

### Definition

```
#include <font/libpvf.h>
ScePvf_t_error errorCode = scePvfGetVertCharInfo(
        ScePvf_t_fontId fontID,
        ScePvf_t_charCode charCode,
        ScePvf_t_charInfo *charInfo
);
```

### Calling Conditions

Cannot be called from an interrupt handler.

Can be called from a thread (must be called in an interrupt-enabled state).

Not multithread safe.

### Arguments

| | |
|---|---|
| *fontID* | Font ID |
| *charCode* | UCS2 character code |
| *charInfo* | Character information |

### Return Values

If the function completes normally, SCE_OK is returned.

If an error occurs, one of the following is returned.

SCE_PVF_ERROR_NOMEMORY, SCE_PVF_ERROR_ARG, SCE_PVF_ERROR_NOGLYPH

### Description

This function stores the vertical layout glyph character information related to the character code indicated by *charCode* of the open font indicated by *fontID* in *charInfo*.

If a vertical layout glyph does not exist, a horizontal layout glyph value will be returned (equivalent value as the values returned by scePvfGetCharInfo()) without being handled as an error.

### Examples

```
ScePvf_t_error errorCode;
ScePvf_t_charInfo charInfo;
ScePvf_t_charCode charCode = 0x3001;

errorCode = scePvfGetVertCharInfo (fontID, charCode, &charInfo);
if ( errorCode != SCE_OK ) {
        printf ("Error (scePvfGetVertCharInfo): 0x%8.8x¥n", (int)errorCode);
}
```

### See Also

ScePvf_t_fontId, ScePvf_t_charCode, ScePvf_t_charInfo, ScePvfTCharInfo, ScePvf_t_error, scePvfOpen(), scePvfIsVertElement(), scePvfGetCharInfo()

# scePvfGetCharImageRect

Get size of character glyph image rectangle

### Definition

```
#include <font/libpvf.h>
ScePvf_t_error errorCode = scePvfGetCharImageRect(
        ScePvf_t_fontId fontID,
        ScePvf_t_charCode charCode,
        ScePvf_t_irect *rect
);
```

### Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

### Arguments

| | |
|---|---|
| *fontID* | Font ID |
| *charCode* | UCS2 character code |
| *rect* | Rectangle information |

### Return Values

If the function completes normally, SCE_OK is returned.

If an error occurs, one of the following is returned.

SCE_PVF_ERROR_NOMEMORY, SCE_PVF_ERROR_ARG, SCE_PVF_ERROR_NOGLYPH

### Description

This function stores the size of the glyph image rectangle of the character code indicated by *charCode* of the open font indicated by *fontID* in *rect*.

### Examples

```
ScePvf_t_error errorCode;
ScePvf_t_charCode charCode = 0x0041;
ScePvf_t_irect rect;

errorCode = scePvfGetCharImageRect (fontID, charCode, &rect);
if ( errorCode != SCE_OK ) {
        printf ("Error (scePvfGetCharImageRect): 0x%8.8x\n", (int)errorCode);
}
```

### See Also

ScePvf_t_fontId, ScePvf_t_charCode, ScePvf_t_charInfo, ScePvfTCharInfo, ScePvf_t_error, scePvfOpen(), scePvfGetVertCharImageRect()

# scePvfGetVertCharImageRect

Get vertical layout character glyph image rectangle size

### Definition

```
#include <font/libpvf.h>
ScePvf_t_error errorCode = scePvfGetVertCharImageRect(
        ScePvf_t_fontId fontID,
        ScePvf_t_charCode charCode,
        ScePvf_t_irect *rect
);
```

### Calling Conditions

Cannot be called from an interrupt handler.

Can be called from a thread (must be called in an interrupt-enabled state).

Not multithread safe.

### Arguments

| | |
|---|---|
| *fontID* | Font ID |
| *charCode* | UCS2 character code |
| *rect* | Rectangle information |

### Return Values

If the function completes normally, SCE_OK is returned.

If an error occurs, one of the following is returned.

SCE_PVF_ERROR_NOMEMORY, SCE_PVF_ERROR_ARG, SCE_PVF_ERROR_NOGLYPH

### Description

This function stores the size of the vertical layout glyph image rectangle of the character code indicated by *charCode* of the open font indicated by *fontID* in *rect*.

If a vertical layout glyph does not exist, a horizontal layout glyph image rectangle size will be returned (equivalent value as the values returned by scePvfGetCharImageRect()) without being handled as an error.

### Examples

```
ScePvf_t_error errorCode;
ScePvf_t_charCode charCode = 0x3001;
ScePvf_t_irect rect;

errorCode = scePvfGetVertCharImageRect (fontID, charCode, &rect);
if ( errorCode != SCE_OK ) {
        printf ("Error (scePvfGetCharImageRect): 0x%8.8x¥n", (int)errorCode);
}
```

### See Also

ScePvf_t_fontId, ScePvf_t_charCode, ScePvf_t_charInfo, ScePvfTCharInfo, ScePvf_t_error, scePvfOpen(), scePvfIsVertElement(), scePvfGetCharImageRect()

# scePvfGetCharGlyphImage

Get character glyph image

### Definition

```
#include <font/libpvf.h>
ScePvf_t_error errorCode = scePvfGetCharGlyphImage(
        ScePvf_t_fontId fontID,
        ScePvf_t_charCode charCode,
        ScePvf_t_userImageBufferRec *imageBuffer
);
```

### Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

### Arguments

| | |
|---|---|
| *fontID* | Font ID |
| *charCode* | UCS2 character code |
| *imageBuffer* | Pointer to area where information related to buffer for storing glyph image is stored |

### Return Values

If the function completes normally, SCE_OK is returned.

If an error occurs, one of the following is returned.

SCE_PVF_ERROR_NOMEMORY, SCE_PVF_ERROR_ARG, SCE_PVF_ERROR_NOGLYPH

### Description

This function stores the glyph image of the character code indicated by *charCode* of the open font indicated by *fontID* at the specified position of the buffer indicated by *imageBuffer*.

### Examples

```
#define IMAGE_WIDTH (256)
#define IMAGE_HEIGHT (128)
ScePvf_t_error errorCode;
ScePvf_t_charCode charCode = 0x0041;
ScePvf_t_userImageBufferRec imageBufferInfo;

imageBufferInfo.pixelFormat = SCE_PVF_USERIMAGE_DIRECT4_L;
imageBufferInfo.rect.width = IMAGE_WIDTH;
imageBufferInfo.rect.height = IMAGE_HEIGHT;
imageBufferInfo.bytesPerLine = IMAGE_WIDTH / 2;
imageBufferInfo.xPos64 = 0 << 6;
imageBufferInfo.yPos64 = 10 << 6;
imageBufferInfo.reserved = 0;
imageBufferInfo.buffer
        = (ScePvf_t_u8 *)memalign (16, IMAGE_WIDTH * IMAGE_HEIGHT);
errorCode = scePvfGetCharGlyphImage (fontID, charCode, &imageBufferInfo);
if ( errorCode != SCE_OK ) {
        printf ("Error (scePvfGetCharGlyphImage): 0x%8.8x\n",
```

```
                                (int)errorCode);
        }
```

**See Also**

ScePvf_t_fontId, ScePvf_t_charCode, ScePvf_t_userImageBufferRec, ScePvfTUserImageBufferRec, ScePvf_t_error, scePvfOpen(), scePvfGetVertCharGlyphImage()

# scePvfGetVertCharGlyphImage

Get vertical layout character glyph image

### Definition

```
#include <font/libpvf.h>
ScePvf_t_error errorCode = scePvfGetVertCharGlyphImage(
        ScePvf_t_fontId fontID,
        ScePvf_t_charCode charCode,
        ScePvf_t_userImageBufferRec *imageBuffer
);
```

### Calling Conditions

Cannot be called from an interrupt handler.

Can be called from a thread (must be called in an interrupt-enabled state).

Not multithread safe.

### Arguments

| | |
|---|---|
| *fontID* | Font ID |
| *charCode* | UCS2 character code |
| *imageBuffer* | Pointer to area where information related to buffer for storing glyph image is stored |

### Return Values

If the function completes normally, SCE_OK is returned.

If an error occurs, one of the following is returned.

SCE_PVF_ERROR_NOMEMORY, SCE_PVF_ERROR_ARG, SCE_PVF_ERROR_NOGLYPH

### Description

This function stores the vertical layout glyph image of the character code indicated by *charCode* of the open font indicated by *fontID* at the specified position of the buffer indicated by *imageBuffer*.

If a vertical layout glyph does not exist, the processing will obtain a horizontal layout glyph image (equivalent to the scePvfGetCharGlyphImage() processing) without being handled as an error.

## Examples

```
#define IMAGE_WIDTH (256)
#define IMAGE_HEIGHT (128)
ScePvf_t_error errorCode;
ScePvf_t_charCode charCode = 0x3001;
ScePvf_t_userImageBufferRec imageBufferInfo;

imageBufferInfo.pixelFormat = SCE_PVF_USERIMAGE_DIRECT4_L;
imageBufferInfo.rect.width = IMAGE_WIDTH;
imageBufferInfo.rect.height = IMAGE_HEIGHT;
imageBufferInfo.bytesPerLine = IMAGE_WIDTH / 2;
imageBufferInfo.xPos64 = 0 << 6;
imageBufferInfo.yPos64 = 10 << 6;
imageBufferInfo.reserved = 0;
imageBufferInfo.buffer
        = (ScePvf_t_u8 *)memalign (16, IMAGE_WIDTH * IMAGE_HEIGHT);
errorCode = scePvfGetCharGlyphImage (fontID, charCode, &imageBufferInfo);
if ( errorCode != SCE_OK ) {
        printf ("Error (scePvfGetVertCharGlyphImage): 0x%8.8x¥n",
                    (int)errorCode);
}
```

## See Also

ScePvf_t_fontId, ScePvf_t_charCode, ScePvf_t_userImageBufferRec, ScePvfTUserImageBufferRec, ScePvf_t_error, scePvfOpen(), scePvfIsVertElement(), scePvfGetCharGlyphImage()

# scePvfGetCharGlyphImage_Clip, scePvfGetCharGlyphImageClip

Get character glyph image with rectangle clipping function

## Definition

```
#include <font/libpvf.h>
ScePvf_t_error errorCode = scePvfGetCharGlyphImage_Clip(
        ScePvf_t_fontId fontID,
        ScePvf_t_charCode charCode,
        ScePvf_t_userImageBufferRec *imageBuffer,
        ScePvf_t_s32 clipX,
        ScePvf_t_s32 clipY,
        ScePvf_t_u32 clipWidth,
        ScePvf_t_u32 clipHeight
);
```

## Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

## Arguments

| | |
|---|---|
| *fontID* | Font ID |
| *charCode* | UCS2 character code |
| *imageBuffer* | Pointer to area where information related to buffer for storing glyph image is stored |
| *clipX* | X-position of clipping rectangle |
| *clipY* | Y-position of clipping rectangle |
| *clipWidth* | Clipping rectangle width |
| *clipHeight* | Clipping rectangle height |

## Return Values

If the function completes normally, `SCE_OK` is returned.

If an error occurs, one of the following is returned.

`SCE_PVF_ERROR_NOMEMORY`, `SCE_PVF_ERROR_ARG`, `SCE_PVF_ERROR_NOGLYPH`

## Description

This function stores the glyph image of the character code indicated by *charCode* of the open font indicated by *fontID* at the specified position of the buffer indicated by *imageBuffer* while clipping it according to *clipX*, *clipY*, *clipWidth*, and *clipHeight*. Both function names (`scePvfGetCharGlyphImage_Clip()` and `scePvfGetCharGlyphImageClip()`) can be used in the same way.

## Examples

```
#define IMAGE_WIDTH (256)
#define IMAGE_HEIGHT (128)
ScePvf_t_error errorCode;
ScePvf_t_charCode charCode = 0x90a3;
ScePvf_t_userImageBufferRec, ScePvfTUserImageBufferRec imageBufferInfo;

imageBufferInfo.pixelFormat = SCE_PVF_USERIMAGE_DIRECT4_L;
imageBufferInfo.rect.width = IMAGE_WIDTH;
imageBufferInfo.rect.height = IMAGE_HEIGHT;
imageBufferInfo.bytesPerLine = IMAGE_WIDTH / 2;
imageBufferInfo.xPos64 = 0 << 6;
imageBufferInfo.yPos64 = 10 << 6;
imageBufferInfo.reserved = 0;
imageBufferInfo.buffer
        = (ScePvf_t_u8 *)memalign (16, IMAGE_WIDTH * IMAGE_HEIGHT);
errorCode = scePvfGetCharGlyphImage_Clip
         (fontID, charCode, &imageBufferInfo, 3, 3, 10, 10);
if ( errorCode != SCE_OK ) {
        printf ("Error (scePvfGetCharGlyphImage_Clip): 0x%8.8x\n",
(int)errorCode);
}
```

## See Also

ScePvf_t_fontId, ScePvf_t_charCode, ScePvf_t_userImageBufferRec, ScePvfTUserImageBufferRec, ScePvf_t_error, scePvfOpen(), scePvfGetVertCharGlyphImage_Clip(), scePvfGetVertCharGlyphImageClip()

# scePvfGetVertCharGlyphImage_Clip, scePvfGetVertCharGlyphImageClip

Get vertical layout character glyph image with rectangle clipping function

## Definition

```
#include <font/libpvf.h>
ScePvf_t_error errorCode = scePvfGetVertCharGlyphImage_Clip(
        ScePvf_t_fontId fontID,
        ScePvf_t_charCode charCode,
        ScePvf_t_userImageBufferRec *imageBuffer,
        ScePvf_t_s32 clipX,
        ScePvf_t_s32 clipY,
        ScePvf_t_u32 clipWidth,
        ScePvf_t_u32 clipHeight
);
```

## Calling Conditions

Cannot be called from an interrupt handler.

Can be called from a thread (must be called in an interrupt-enabled state).

Not multithread safe.

## Arguments

| | |
|---|---|
| *fontID* | Font ID |
| *charCode* | UCS2 character code |
| *imageBuffer* | Pointer to area where information related to buffer for storing glyph image is stored |
| *clipX* | X-position of clipping rectangle |
| *clipY* | Y-position of clipping rectangle |
| *clipWidth* | Clipping rectangle width |
| *clipHeight* | Clipping rectangle height |

## Return Values

If the function completes normally, `SCE_OK` is returned.

If an error occurs, one of the following is returned.

`SCE_PVF_ERROR_NOMEMORY`, `SCE_PVF_ERROR_ARG`, `SCE_PVF_ERROR_NOGLYPH`

## Description

This function stores the vertical layout glyph image of the character code indicated by *charCode* of the open font indicated by *fontID* at the specified position of the buffer indicated by *imageBuffer* while clipping it according to *clipX*, *clipY*, *clipWidth*, and *clipHeight*. Both function names (`scePvfGetVertCharGlyphImage_Clip()` and `scePvfGetVertCharGlyphImageClip()`) can be used in the same way.

If a vertical layout glyph does not exist, the processing will obtain a horizontal layout glyph image (equivalent to the `scePvfGetCharGlyphImage_Clip()` processing) without being handled as an error.

## Examples

```
#define IMAGE_WIDTH (256)
#define IMAGE_HEIGHT (128)
ScePvf_t_error errorCode;
ScePvf_t_charCode charCode = 0x3001;
ScePvf_t_userImageBufferRec, ScePvfTUserImageBufferRec imageBufferInfo;

imageBufferInfo.pixelFormat = SCE_PVF_USERIMAGE_DIRECT4_L;
imageBufferInfo.rect.width = IMAGE_WIDTH;
imageBufferInfo.rect.height = IMAGE_HEIGHT;
imageBufferInfo.bytesPerLine = IMAGE_WIDTH / 2;
imageBufferInfo.xPos64 = 0 << 6;
imageBufferInfo.yPos64 = 10 << 6;
imageBufferInfo.reserved = 0;
imageBufferInfo.buffer
        = (ScePvf_t_u8 *)memalign (16, IMAGE_WIDTH * IMAGE_HEIGHT);
errorCode = scePvfGetVertCharGlyphImage_Clip
         (fontID, charCode, &imageBufferInfo, 3, 3, 10, 10);
if ( errorCode != SCE_OK ) {
        printf ("Error (scePvfGetCharGlyphImage_Clip): 0x%8.8x¥n",
(int)errorCode);
}
```

## See Also

ScePvf_t_fontId, ScePvf_t_charCode, ScePvf_t_userImageBufferRec, ScePvfTUserImageBufferRec, ScePvf_t_error, scePvfOpen(), scePvfIsVertElement(), scePvfGetCharGlyphImage_Clip(), scePvfGetCharGlyphImageClip()

# scePvfGetKerningInfo

Get kerning information

### Definition

```
#include <font/libpvf.h>
ScePvf_t_error errorCode = scePvfGetKerningInfo(
        ScePvf_t_fontId fontID,
        ScePvf_t_charCode leftCharCode,
        ScePvf_t_charCode rightCharCode,
        ScePvf_t_kerningInfo *kerningInfo
);
```

### Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

### Arguments

| | |
|---|---|
| *fontID* | Font ID |
| *leftCharCode* | UCS2 character code of characters positioned to the left |
| *rightCharCode* | UCS2 character code of characters positioned to the right |
| *kerningInfo* | Kerning information |

### Return Values

If the function completes normally, SCE_OK is returned.

If an error occurs, one of the following is returned.

SCE_PVF_ERROR_NOMEMORY, SCE_PVF_ERROR_ARG, SCE_PVF_ERROR_NOGLYPH

### Description

This function stores the kerning information defined by the combination of the open font *fontID*'s character codes *leftCharCode* and *rightCharCode* to *kerningInfo*.

Note that the vector fonts installed on the PlayStation®Vita do not have kerning information.

Kerning information can be retrieved if fonts with kerning information are opened using any of the following functions: scePvfOpenUserFile(), scePvfOpenUserMemory(), scePvfOpenUserFileWithSubfontIndex() or scePvfOpenUserMemoryWithSubfontIndex().

### Examples

```
ScePvf_t_error errorCode;
ScePvf_t_kerningInfo kerningInfo;
ScePvf_t_charCode leftCharCode = 0x0056;
ScePvf_t_charCode rightCharCode = 0x0041;

errorCode = scePvfGetKerningInfo
        (fontID, leftCharCode, rightCharCode, &kerningInfo);
if ( errorCode != SCE_OK ) {
        printf ("Error (scePvfGetCharInfo): 0x%8.8x¥n", (int)errorCode);
}
```

**See Also**

ScePvf_t_fontId, ScePvf_t_charCode, ScePvfTCharInfo, ScePvf_t_error, scePvfOpenUserFile(), scePvfOpenUserMemory(), scePvfOpenUserFileWithSubfontIndex(), scePvfOpenUserMemoryWithSubfontIndex(), scePvfSetCharSize()

# scePvfPixelToPointH

Convert from pixels to points (values related to horizontal direction)

### Definition

```
#include <font/libpvf.h>
ScePvf_t_f32 point = scePvfPixelToPointH(
        ScePvf_t_libId libID,
        ScePvf_t_f32 pixel,
        ScePvf_t_error *errorCode
);
```

### Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

### Arguments

| | |
|---|---|
| *libID* | Library instance pointer |
| *pixel* | Value in pixel units |
| *errorCode* | Address for storing the error code |

### Return Values

If the function completes normally, the result when *pixel* is unit converted to points is returned.

If an error occurs, 0 is returned in *point* and one of the following is stored in **errorCode*.

SCE_PVF_ERROR_LIBID, SCE_PVF_ERROR_NOSUPPORT

### Description

This function converts the value indicated by *pixel*, which is in dot units, to the point value according to the horizontal resolution (dpi) value set in the *libID* instance.

### Examples

```
ScePvf_t_error errorCode;
ScePvf_t_f32 point;

point = scePvfPixelToPointH (libID, (ScePvf_t_f32)18.0f, &errorCode);
if ( errorCode != SCE_OK ) {
        printf ("Error (scePvfPixelToPointH): 0x%8.8x\n", (int)errorCode);
}
```

### See Also

ScePvf_t_libId, ScePvf_t_error, scePvfPixelToPointV(), scePvfPointToPixelH(), scePvfPointToPixelV(), scePvfSetResolution(), ScePvf_t_fontStyleInfo, ScePvfTFontStyleInfo

# scePvfPixelToPointV

Convert from pixels to points (values related to vertical direction)

### Definition

```
#include <font/libpvf.h>
ScePvf_t_f32 point = scePvfPixelToPointV(
        ScePvf_t_libId libID,
        ScePvf_t_f32 pixel,
        ScePvf_t_error *errorCode
);
```

### Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

### Arguments

| | |
|---|---|
| *libID* | Library instance pointer |
| *pixel* | Value in pixel units |
| *errorCode* | Address for storing the error code |

### Return Values

If the function completes normally, the result when *pixel* is unit converted to points is returned.

If an error occurs, 0 is returned in *point* and one of the following is stored in **errorCode*.

SCE_PVF_ERROR_LIBID, SCE_PVF_ERROR_NOSUPPORT

### Description

This function converts the value indicated by *pixel*, which is in dot units, to the point value according to the vertical resolution (dpi) value set in the *libID* instance.

### Examples

```
ScePvf_t_error errorCode;
ScePvf_t_f32 point;

point = scePvfPixelToPointV (libID, (ScePvf_t_f32)18.0f, &errorCode);
if ( errorCode != SCE_OK ) {
        printf ("Error (scePvfPixelToPointV): 0x%8.8x\n", (int)errorCode);
}
```

### See Also

ScePvf_t_libId, ScePvf_t_error, scePvfPixelToPointH(), scePvfPointToPixelH(), scePvfPointToPixelV(), scePvfSetResolution(), ScePvf_t_fontStyleInfo, ScePvfTFontStyleInfo

# scePvfPointToPixelH

Convert from points to pixels (values related to horizontal direction)

### Definition

```
#include <font/libpvf.h>
ScePvf_t_f32 pixel = scePvfPointToPixelH(
        ScePvf_t_libId libID,
        ScePvf_t_f32 point,
        ScePvf_t_error *errorCode
);
```

### Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

### Arguments

| | |
|---|---|
| *libID* | Library instance pointer |
| *point* | Value in point units |
| *errorCode* | Address for storing the error code |

### Return Values

If the function completes normally, the result when *point* is unit converted to pixels is returned.

If an error occurs, 0 is returned in *pixel* and one of the following is stored in **errorCode*.

SCE_PVF_ERROR_LIBID, SCE_PVF_ERROR_ARG

### Description

This function converts the value indicated by *point*, which is in point units, to the pixel value according to the horizontal resolution (dpi) value set in the *libID* instance.

### Examples

```
ScePvf_t_error errorCode;
ScePvf_t_f32 pixel;

pixel = scePvfPointToPixelH (libID, (ScePvf_t_f32)10.0f, &errorCode);
if ( errorCode != SCE_OK ) {
        printf ("Error (scePvfPointToPixelH): 0x%8.8x\n", (int)errorCode);
}
```

### See Also

ScePvf_t_libId, ScePvf_t_error, scePvfPixelToPointH(), scePvfPixelToPointV(), scePvfPointToPixelV(), scePvfSetResolution(), ScePvf_t_fontStyleInfo, ScePvfTFontStyleInfo

# scePvfPointToPixelV

Convert from points to pixels (values related to vertical direction)

### Definition

```
#include <font/libpvf.h>
ScePvf_t_f32 pixel = scePvfPointToPixelV(
        ScePvf_t_libId libID,
        ScePvf_t_f32 point,
        ScePvf_t_error *errorCode
);
```

### Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

### Arguments

| | |
|---|---|
| *libID* | Library instance pointer |
| *point* | Value in point units |
| *errorCode* | Address for storing the error code |

### Return Values

If the function completes normally, the result when *point* is unit converted to pixels is returned.

If an error occurs, 0 is returned in *pixel* and one of the following is stored in **errorCode*.

SCE_PVF_ERROR_LIBID, SCE_PVF_ERROR_ARG

### Description

This function converts the value indicated by *point*, which is in point units, to the pixel value according to the vertical resolution (dpi) value set in the *libID* instance.

### Examples

```
ScePvf_t_error errorCode;
ScePvf_t_f32 pixel;

pixel = scePvfPointToPixelV (libID, (ScePvf_t_f32)10.0f, &errorCode);
if ( errorCode != SCE_OK ) {
        printf ("Error (scePvfPointToPixelV): 0x%8.8x\n", (int)errorCode);
}
```

### See Also

ScePvf_t_libId, ScePvf_t_error, scePvfPixelToPointH(), scePvfPixelToPointV(), scePvfPointToPixelH(), scePvfSetResolution(), ScePvf_t_fontStyleInfo, ScePvfTFontStyleInfo

# scePvfSetAltCharacterCode

Set alternate character code

### Definition

```
#include <font/libpvf.h>
ScePvf_t_error errorCode = scePvfSetAltCharacterCode(
        ScePvf_t_libId libID,
        ScePvf_t_charCode charCode
);
```

### Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

### Arguments

| | |
|---|---|
| *libID* | Library instance pointer |
| *charCode* | Character code of alternate characters |

### Return Values

If the function completes normally, SCE_OK is returned.

If an error occurs, one of the following is returned.

SCE_PVF_ERROR_LIBID, SCE_PVF_ERROR_ARG, SCE_PVF_ERROR_NOSUPPORT, SCE_PVF_ERROR_UNKNOWN

### Description

This function specifies the character code of the characters to be used as alternate glyphs when an attempt is made to get the glyph image by using a function such as scePvfGetCharGlyphImage() for a character code for which no glyphs exist.

Immediately after an instance of the libpvf library is created, U+005F is set as the default alternate character code.

When a function such as scePvfGetCharGlyphImage() is used to access a font and the font to be accessed does not contain the character with the character code that was passed to the function, libpvf automatically replaces the character with the character of the character code that was set by scePvfSetAltCharacterCode().

However, a precondition for this to work is that the character set of the font that is to be accessed by a function such as scePvfGetCharGlyphImage() contains the character of the character code *charCode* that was specified by scePvfSetAltCharacterCode().

If the font does not contain that alternate character code, replacement by an alternate character is not performed, and the result is handled as a character that has both its width and height set to zero.

The default alternate character code U+005F is included in both the built-in Latin fonts and built-in Japanese fonts, but it is not included in the built-in internal Korean fonts. This means that when a built-in Korean font is used with the default alternate character, alternate character replacement is not performed.

## Examples

```
ScePvf_t_error errorCode;
ScePvf_t_f32 pixel;

errorCode = scePvfSetAltCharacterCode (libID, (ScePvf_t_charCode)0x22a0);
if ( errorCode != SCE_OK ) {
        printf ("Error (scePvfSetAltCharacterCode): 0x%8.8x\n",
(int)errorCode);
}
```

## See Also

ScePvf_t_libId, ScePvf_t_error, ScePvf_t_charCode, scePvfGetCharGlyphImage()

# Constants

# List of Error Codes

## libpvf Error Codes

### Definition

| Macro | Value | Description |
|---|---|---|
| SCE_OK | 0 | No error (normal termination) |
| SCE_PVF_ERROR_NOMEMORY | 0x80460001 | Memory allocation failed |
| SCE_PVF_ERROR_LIBID | 0x80460002 | Invalid library instance |
| SCE_PVF_ERROR_ARG | 0x80460003 | Invalid argument |
| SCE_PVF_ERROR_NOFILE | 0x80460004 | No file |
| SCE_PVF_ERROR_FILEOPEN | 0x80460005 | Processing for opening file failed |
| SCE_PVF_ERROR_FILECLOSE | 0x80460006 | Processing for closing file failed |
| SCE_PVF_ERROR_FILEREAD | 0x80460007 | Processing for reading file failed |
| SCE_PVF_ERROR_FILESEEK | 0x80460008 | File seeking failed |
| SCE_PVF_ERROR_TOOMANYOPENED | 0x80460009 | Too many open fonts |
| SCE_PVF_ERROR_ILLEGALVERSION | 0x8046000a | Unsupported font version |
| SCE_PVF_ERROR_DATAINCONSISTENT | 0x8046000b | Inconsistency in font data |
| SCE_PVF_ERROR_EXPIRED | 0x8046000c | Usage period expired |
| SCE_PVF_ERROR_NOGLYPH | 0x8046000d | Glyph does not exist |
| SCE_PVF_ERROR_NOSUPPORT | 0x8046000e | Unsupported cause |
| SCE_PVF_ERROR_UNKNOWN | 0x8046ffff | Unknown error |