

CES Library Reference

© 2015 Sony Computer Entertainment Inc.
All Rights Reserved.
SCE Confidential

Table of Contents

Functions for Handling UCS Character Encoding Schemes.....	7
Functions for Checking the Code of One Character	8
sceCesUtf32CheckCode.....	8
sceCesUtf32beCheckCode, sceCesUtf32leCheckCode	9
sceCesUtf16CheckCode, sceCesUtf16beCheckCode, sceCesUtf16leCheckCode	10
sceCesUtf8CheckCode.....	12
sceCesUcs2CheckCode	14
CES Conversion for One Character.....	15
sceCesUtf32ToUtf16, sceCesUtf32ToUtf16be, sceCesUtf32ToUtf16le	15
sceCesUtf32ToUtf8	17
sceCesUtf32ToUcs2.....	19
sceCesUtf32beToUtf16, sceCesUtf32beToUtf16be, sceCesUtf32beToUtf16le	21
sceCesUtf32beToUtf8	23
sceCesUtf32beToUcs2.....	25
sceCesUtf32leToUtf16, sceCesUtf32leToUtf16be, sceCesUtf32leToUtf16le	27
sceCesUtf32leToUtf8	29
sceCesUtf32leToUcs2.....	31
sceCesUtf16ToUtf32, sceCesUtf16ToUtf32be, sceCesUtf16ToUtf32le	33
sceCesUtf16ToUtf8	35
sceCesUtf16ToUcs2.....	37
sceCesUtf16beToUtf32, sceCesUtf16beToUtf32be, sceCesUtf16beToUtf32le	39
sceCesUtf16beToUtf8	41
sceCesUtf16beToUcs2.....	43
sceCesUtf16leToUtf32, sceCesUtf16leToUtf32be, sceCesUtf16leToUtf32le	45
sceCesUtf16leToUtf8	47
sceCesUtf16leToUcs2.....	49
sceCesUtf8ToUtf32, sceCesUtf8ToUtf32be, sceCesUtf8ToUtf32le.....	51
sceCesUtf8ToUtf16, sceCesUtf8ToUtf16be, sceCesUtf8ToUtf16le.....	53
sceCesUtf8ToUcs2.....	56
sceCesUcs2ToUtf32, sceCesUcs2ToUtf32be, sceCesUcs2ToUtf32le.....	58
sceCesUcs2ToUtf16, sceCesUcs2ToUtf16be, sceCesUcs2ToUtf16le.....	60
sceCesUcs2ToUtf8.....	62
Context for Character String Conversion	64
SceCesUcsContext.....	64
sceCesUcsContextInit.....	65
sceCesUcsContextInitCopy	66
Settings for Character Sting Conversion.....	67
sceCesSetUcsPolicyDetectBom	67
sceCesSetUcsPolicyOutputBom.....	69
sceCesSetUtf16StrEndian, sceCesSetUtf32StrEndian.....	71
Encoding Scheme Conversion and Copy of Character Strings	73
sceCesUtf32StrToUtf16Str	73
sceCesUtf32StrToUtf8Str	75
sceCesUtf32StrToUcs2Str.....	77
sceCesUtf16StrToUtf32Str	79

sceCesUtf16StrToUtf8Str	81
sceCesUtf16StrToUcs2Str	83
sceCesUtf8StrToUtf32Str	85
sceCesUtf8StrToUtf16Str	87
sceCesUtf8StrToUcs2Str	89
sceCesUcs2StrToUtf32Str	91
sceCesUcs2StrToUtf16Str	93
sceCesUcs2StrToUtf8Str	95
sceCesUtf32StrToCopyStr	97
sceCesUtf16StrToCopyStr	99
sceCesUtf8StrToCopyStr	101
sceCesUcs2StrToCopyStr	103
Retrieving Character String Length by Encoding Scheme	105
sceCesUtf32StrGetUtf16Len	105
sceCesUtf32StrGetUtf8Len	107
sceCesUtf32StrGetUcs2Len	109
sceCesUtf16StrGetUtf32Len	111
sceCesUtf16StrGetUtf8Len	113
sceCesUtf16StrGetUcs2Len	115
sceCesUtf8StrGetUtf32Len	117
sceCesUtf8StrGetUtf16Len	119
sceCesUtf8StrGetUcs2Len	121
sceCesUcs2StrGetUtf32Len	123
sceCesUcs2StrGetUtf16Len	125
sceCesUcs2StrGetUtf8Len	127
sceCesUtf32StrGetCopyLen	129
sceCesUtf16StrGetCopyLen	131
sceCesUtf8StrGetCopyLen	133
sceCesUcs2StrGetCopyLen	135
Other Character Sets and UCS Character String Functions	137
Character String Conversion Context of MBCS (Including SBCS) and UCS	138
SceCesMbcsUcsContext	138
sceCesMbcsUcsContextInit	139
sceCesMbcsUcsContextInitCopy	141
Character String Conversion Functions for Handling MBCS (Including SBCS) and UCS	142
sceCesMbcsStrToUtf32Str	142
sceCesMbcsStrToUtf16Str	144
sceCesMbcsStrToUtf8Str	146
sceCesMbcsStrToUcs2Str	148
sceCesUtf32StrToMbcsStr	150
sceCesUtf16StrToMbcsStr	152
sceCesUtf8StrToMbcsStr	154
sceCesUcs2StrToMbcsStr	156
Character String Length Retrieval Functions for Handling MBCS (Including SBCS) and UCS	158
sceCesMbcsStrGetUtf32Len	158
sceCesMbcsStrGetUtf16Len	160
sceCesMbcsStrGetUtf8Len	162
sceCesMbcsStrGetUcs2Len	164

sceCesUtf32StrGetMbcLen	166
sceCesUtf16StrGetMbcLen	168
sceCesUtf8StrGetMbcLen	170
sceCesUcs2StrGetMbcLen	172
Character String Functions for Handling UCS Integrated Interconversion	174
sceCesUcsStrGetEncodingSize	175
sceCesUcsStrConvertEncoding	178
Setting Functions for Character String Control and Error Handling	181
Setting Functions for Character String Control	182
sceCesSetStrBegin, sceCesSetStrLast, sceCesSetStrEnd, sceCesSetStrContinue	182
sceCesSetStrMaxCharCount	184
Setting Functions for Error Handling	186
sceCesSetErrorOperation	186
sceCesUnsetErrorOperation	189
sceCesSetReplacementCharUCode	190
sceCesSetUcsReplacementCharCode, sceCesSetMbcReplacementCharCode, sceCesSetMbcReplacementCharUCode	191
UCS Conversion Profiles of Single-Byte Character Sets	193
7-Bit Character Set UCS Conversion Profiles	194
sceCesRefersUcsProfileAscii	194
sceCesRefersUcsProfileJisX0201Roman, sceCesRefersUcsProfileJisX0201RomanTilde0x7e	195
sceCesRefersUcsProfileGbT1988, sceCesRefersUcsProfileGbT1988Tilde0x7e	196
sceCesRefersUcsProfileKsX1003, sceCesRefersUcsProfileKsX1003Tilde0x7e	197
8-Bit Character Set UCS Conversion Profiles	198
sceCesRefersUcsProfileIso8859_1, sceCesRefersUcsProfileIso8859_2, sceCesRefersUcsProfileIso8859_3, sceCesRefersUcsProfileIso8859_4, sceCesRefersUcsProfileIso8859_5, sceCesRefersUcsProfileIso8859_6, sceCesRefersUcsProfileIso8859_7, sceCesRefersUcsProfileIso8859_8, sceCesRefersUcsProfileIso8859_9, sceCesRefersUcsProfileIso8859_10, sceCesRefersUcsProfileIso8859_11, sceCesRefersUcsProfileIso8859_13, sceCesRefersUcsProfileIso8859_14, sceCesRefersUcsProfileIso8859_15, sceCesRefersUcsProfileIso8859_16	198
sceCesRefersUcsProfileJisX0201, sceCesRefersUcsProfileJisX0201Tilde0x7e, sceCesRefersUcsProfileAsciiWithKatakana	200
sceCesRefersUcsProfileKoi8R, sceCesRefersUcsProfileKoi8U	201
sceCesRefersUcsProfileCp1250, sceCesRefersUcsProfileCp1251, sceCesRefersUcsProfileCp1252, sceCesRefersUcsProfileCp1253, sceCesRefersUcsProfileCp1254, sceCesRefersUcsProfileCp1255, sceCesRefersUcsProfileCp1256, sceCesRefersUcsProfileCp1257, sceCesRefersUcsProfileCp1258	202
sceCesRefersUcsProfileCp437, sceCesRefersUcsProfileCp737, sceCesRefersUcsProfileCp775, sceCesRefersUcsProfileCp850, sceCesRefersUcsProfileCp852, sceCesRefersUcsProfileCp855, sceCesRefersUcsProfileCp857, sceCesRefersUcsProfileCp858, sceCesRefersUcsProfileCp860, sceCesRefersUcsProfileCp861, sceCesRefersUcsProfileCp862, sceCesRefersUcsProfileCp863, sceCesRefersUcsProfileCp864, sceCesRefersUcsProfileCp865, sceCesRefersUcsProfileCp866, sceCesRefersUcsProfileCp869, sceCesRefersUcsProfileCp874	204
UCS Conversion Profiles of Multi-Byte Character Sets	206
UCS Conversion Profiles Shared by MBCS	207
SceCesUcsProfileSheet	207
SceCesMbcUcsProfile, sceCesGetMbcUcsProfile	208

SCEI CONFIDENTIAL

UCS Conversion Profile of Shift_JIS	209
SceCesSJisUcsProfile	209
sceCesUcsProfileInitSJis1997X0208	210
sceCesUcsProfileInitSJis1997Cp932, sceCesUcsProfileInitSJis	212
sceCesUcsProfileInitSJis2004X0213	214
sceCesSJisUcsProfileSetSbcs	215
UCS Conversion Profile of EUC-JP	217
SceCesEucJpUcsProfile	217
sceCesUcsProfileInitEucJp, sceCesUcsProfileInitEucJpCp51932	218
sceCesUcsProfileInitEucJpX0208, sceCesUcsProfileInitEucJpX0208Ss2, sceCesUcsProfileInitEucJpX0208Ss2Ss3	219
sceCesUcsProfileInitEucJis2004	220
UCS Conversion Profile of Big5	221
SceCesBig5UcsProfile	221
sceCesUcsProfileInitBig5, sceCesUcsProfileInitBig5Cp950	222
UCS Conversion Profile of GB	223
SceCesGbUcsProfile	223
sceCesUcsProfileInitGb18030, sceCesUcsProfileInitGb18030_2000	224
sceCesUcsProfileInitGb18030_2005	225
sceCesUcsProfileInitGbk, sceCesUcsProfileInitGbkCp936	226
sceCesGbUcsProfileSetSbcs	227
sceCesGbUcsProfileSetUdaMapping	228
sceCesGbUcsProfileSet4ByteCharRange	229
UCS Conversion Profile of EUC-CN	231
SceCesEucCnUcsProfile	231
sceCesUcsProfileInitEucCn, sceCesUcsProfileInitEucCnGb2312	232
UCS Conversion Profile of UHC(Unified Hangul Code)	233
SceCesUhcUcsProfile	233
sceCesUcsProfileInitUhc	234
sceCesUhcUcsProfileSetSbcs	235
UCS Conversion Profile of EUC-KR	236
SceCesEucKrUcsProfile	236
sceCesUcsProfileInitEucKr	237
One-Character Conversion Functions Using UCS Conversion Profiles	238
One-Character Conversion Functions Handling Single-Byte Character Codes and UCS	239
sceCesSbcToUtf32, sceCesSbcToUtf32be, sceCesSbcToUtf32le	239
sceCesSbcToUtf16, sceCesSbcToUtf16be, sceCesSbcToUtf16le	241
sceCesSbcToUtf8	243
sceCesSbcToUcs2	245
sceCesUtf32ToSbc	247
sceCesUtf32beToSbc, sceCesUtf32leToSbc	249
sceCesUtf16ToSbc, sceCesUtf16beToSbc, sceCesUtf16leToSbc	251
sceCesUtf8ToSbc	253
sceCesUcs2ToSbc	255
One-Character Conversion Functions Handling Multi-Byte Character Codes (Including SBC) and UCS	257
sceCesMbcToUtf32, sceCesMbcToUtf32be, sceCesMbcToUtf32le	257
sceCesMbcToUtf16, sceCesMbcToUtf16be, sceCesMbcToUtf16le	260

©SCEI

SCE CONFIDENTIAL

sceCesMbcToUtf8	263
sceCesMbcToUcs2	265
sceCesUtf32ToMbc	267
sceCesUtf32beToMbc, sceCesUtf32leToMbc	269
sceCesUtf16ToMbc, sceCesUtf16beToMbc, sceCesUtf16leToMbc	271
sceCesUtf8ToMbc	274
sceCesUcs2ToMbc	277
Functions for Handling Processing Specific to JIS Character Sets	279
Conversion Profiles of JIS Character Sets and UCS	280
SceCesJiscsUcsProfile, sceCesGetJiscsUcsProfile	280
Conversion Functions of JIS Character Sets and UCS	281
sceCesJiscsToUcs	281
sceCesUcsToJiscs	284
Functions for Handling JIS Characters	286
sceCesJisGetLevel	286
Functions for Handling Shift_JIS Codes	288
sceCesSjisGetCode	288
sceCesSjisPutCode	290
sceCesSjisCodeToJisX0208	292
sceCesSjisCodeToJisX0213	294
sceCesJisX0208ToSjisCode	296
sceCesJisX0213ToSjisCode	297
Constants	299
SceCesEndianParam	300
Return Codes	301

Functions for Handling UCS Character Encoding Schemes

Functions for Checking the Code of One Character

sceCesUtf32CheckCode

Check the code of one UTF-32 character

Definition

```
#include <ces.h>
int sceCesUtf32CheckCode (
    uint32_t utf32
)
```

Arguments

utf32 UTF-32 character code

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code

Description

This function receives a UTF-32 character code, checks whether the character code is normal and returns the result as the function's return value.

Specify the UTF-32 character code in *utf32*.

If SCE_CES_ERROR_INVALID_ENCODE returns, it means that encoding has been determined to be invalid because a value equal or greater than 0x00110000, which is outside the UTF-32 valid range, has been specified in *utf32*.

If SCE_CES_ERROR_ILLEGAL_CODE is returned, it means the code has been determined to be illegal because a value in the range between U+D800 and U+DFFF that is reserved as surrogate area has been passed in *utf32*.

Notes

This function is multi-thread safe.

See Also

sceCesUtf16CheckCode(), sceCesUtf8CheckCode(), sceCesUcs2CheckCode()

SCE CONFIDENTIAL

sceCesUtf32beCheckCode, sceCesUtf32leCheckCode

Check the code of one UTF-32 (BE/LE) character

Definition

```
#include <ces.h>
int sceCesUtf32beCheckCode (
    const uint32_t *utf32addr
)
int sceCesUtf32leCheckCode (
    const uint32_t *utf32addr
)
```

Arguments

utf32addr Address storing UTF-32 character code

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code

Description

The function described here checks whether the 32-bit word value that is stored at the specified address is appropriate as the UTF-32 character code of one character, then returns the result as the function's return value.

Use `sceCesUtf32beCheckCode()` if the value is stored with big-endian order, and use `sceCesUtf32leCheckCode()` if the value is stored with little-endian order.

Specify the address where the UTF-32 character code is stored in *utf32addr*.

If SCE_CES_ERROR_INVALID_ENCODE returns, it means that encoding has been determined to be invalid because a value equal or greater than U+0010FFFF, which is outside the UTF-32 valid range, has been specified in *utf32addr*.

If SCE_CES_ERROR_ILLEGAL_CODE is returned, it means the code has been determined to be illegal because a value in the range between U+D800 and U+DFFF that is reserved as surrogate area has been passed in *utf32be*.

Notes

This function is multi-thread safe.

See Also

`sceCesUtf32CheckCode()`

sceCesUtf16CheckCode, sceCesUtf16beCheckCode, sceCesUtf16leCheckCode

Check the code of one UTF-16 character and retrieve code length

Definition

```
#include <ces.h>
int sceCesUtf16CheckCode (
    const uint16_t *utf16addr,
    uint32_t utf16max,
    uint32_t *utf16Len
)
int sceCesUtf16beCheckCode (
    const uint16_t *utf16addr,
    uint32_t utf16max,
    uint32_t *utf16Len
)
int sceCesUtf16leCheckCode (
    const uint16_t *utf16addr,
    uint32_t utf16max,
    uint32_t *utf16Len
)
```

Arguments

utf16addr Address storing UTF-16 character code
utf16max Buffer length (16-bit word count) for which recognition of UTF-16 character code is allowed
utf16Len Address of the variable for receiving UTF-16 character code length (16-bit word count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code

Description

The function described here checks whether the 16-bit word value that is stored at the specified address is appropriate as the UTF-16 character code of one character, then returns the result as the function's return value. The function can also simultaneously retrieve the length of the code recognized successfully (16-bit word count).

If the calling function is `sceCesUtf16CheckCode()`, the code value will be read in 16-bit word units when checking. If you wish to expressly specify endianness, use `sceCesUtf16beCheckCode()` for big-endian and `sceCesUtf16leCheckCode()` for little-endian.

SCE CONFIDENTIAL

Specify the address where the UTF-16 character code is stored in *utf16addr*.

Specify the length of the buffer (16-bit word count) for which recognition of UTF-16 character code is allowed in *utf16max*.

Specify the address of the variable for receiving the length of the character code (16-bit word count) stored in UTF-16 in *utf16Len*. If a NULL pointer has been specified, this argument will be ignored.

In case of normal termination, the 16-bit word count stored in **utf16Len* will be 1 or 2.

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed in *utf16addr*, or that 0 has been passed in *utf16max*.

If `SCE_CES_ERROR_SRC_BUFFER_END` returns, it means that the character code is interrupted at the maximum limit specified in *utf16max*. In this case, the length of the character code (16-bit word count) whose recognition has been attempted will return to **utf16Len* as a value greater than *utf16max*.

If `SCE_CES_ERROR_ILLEGAL_CODE` is returned, it means the code has been determined to be illegal because codes in the range between U+D800 and U+DFFF that is reserved as surrogate area were not used in pairs. 1 will be stored in **utf16Len* as the length of the code (16-bit word count) that was found to be illegal.

Notes

This function is multi-thread safe.

See Also

`sceCesUtf32CheckCode()`, `sceCesUtf8CheckCode()`, `sceCesUcs2CheckCode()`

SCE CONFIDENTIAL

sceCesUtf8CheckCode

Check the code of one UTF-8 character and retrieve code length

Definition

```
#include <ces.h>
int sceCesUtf8CheckCode (
    const uint8_t *utf8addr,
    uint32_t utf8max,
    uint32_t *utf8Len
)
```

Arguments

<i>utf8addr</i>	Address storing UTF-8 character code
<i>utf8max</i>	Buffer length (byte count) for which recognition of UTF-8 character code is allowed
<i>utf8Len</i>	Address of the variable for receiving UTF-8 character code length (byte count)

Return Values

Returns `SCE_OK (0)` as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
<code>SCE_CES_ERROR_INVALID_SRC_BUFFER</code>	0x805C0010	Specified source buffer is invalid
<code>SCE_CES_ERROR_SRC_BUFFER_END</code>	0x805C0011	Specified source buffer is insufficient
<code>SCE_CES_ERROR_INVALID_ENCODE</code>	0x805C0014	Source encoding determined to be invalid
<code>SCE_CES_ERROR_ILLEGAL_CODE</code>	0x805C0015	Illegal code detected in source character code

Description

This function checks whether the byte string stored at the specified address is appropriate as the UTF-8 character code of one character, and returns the result as the function's return value. The function can also simultaneously retrieve the length of the code recognized successfully (byte count).

Specify the address where the UTF-8 character code is stored in *utf8addr*.

Specify the length of the buffer (byte count) for which recognition of UTF-8 character code is allowed in *utf8max*.

Specify the address of the variable for receiving the length of the character code (byte count) stored in UTF-8 in *utf8Len*. If a NULL pointer has been specified, this argument will be ignored.

In case of normal termination, the value stored in **utf8Len* will be 1 to 6.

In RFC 3629, UTF-8 is up to 4 bytes, but this function will not determine 5 or 6 bytes as an error.

Users should determine separately the number of bytes to allow. Enabling up to 3 bytes means enabling the UCS-2 range (U+FFFF).

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed in *utf8addr*, or that 0 has been passed in *utf8max*.

SCE CONFIDENTIAL

If `SCE_CES_ERROR_SRC_BUFFER_END` returns, it means that the character code is interrupted at the maximum limit specified in `utf8max`. In case of this error, the character code length determined from the first UTF-8 byte will return to `*utf8Len`. Note that a higher value than that specified in `utf8max` will return.

If `SCE_CES_ERROR_INVALID_ENCODE` returns, it means that the encoding is found to be invalid because a byte string that cannot be recognized as UTF-8 has been passed in `utf8addr`. The number of bytes that were recognized successfully as UTF-8 will return to `*utf8Len` as a value between 0 and 5.

If `SCE_CES_ERROR_ILLEGAL_CODE` returns, it means the code has been determined to be illegal because the byte string specified in `utf8addr` contains encoding of code in the U+D800 to U+DFFF range reserved as a surrogate area, or encoding with an unnecessarily large number of bytes (such as representing U+0000 as the 2-byte sequence C0,80). The code length (byte count) of the code that has been determined to be illegal is stored in `*utf8Len`.

Notes

This function is multi-thread safe.

See Also

`sceCesUcs2CheckCode()`, `sceCesUtf16CheckCode()`, `sceCesUtf32CheckCode()`

SCE CONFIDENTIAL

sceCesUcs2CheckCode

Check the code of one UCS-2 character

Definition

```
#include <ces.h>
int sceCesUcs2CheckCode (
    uint16_t ucs2
)
```

Arguments

ucs2 UCS-2 character code

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code

Description

This function checks whether the specified UCS-2 code (16-bit word value) is appropriate as a UCS-2 character code, and returns the result as the function's return value.

Specify the UCS-2 character code as a 16-bit value in *ucs2*.

Character codes equal or greater than U+00010000, which cannot be represented in UCS-2, cannot be passed.

If SCE_CES_ERROR_ILLEGAL_CODE is returned, it means the code has been determined to be illegal because a value in the range between U+D800 and U+DFFF that is reserved as surrogate area has been passed in *ucs2*.

Notes

This function is multi-thread safe.

See Also

sceCesUtf32CheckCode(), sceCesUtf16CheckCode(), sceCesUtf8CheckCode()

CES Conversion for One Character

sceCesUtf32ToUtf16, sceCesUtf32ToUtf16be, sceCesUtf32ToUtf16le

Conversion of one character from UTF-32 to UTF-16

Definition

```
#include <ces.h>
int sceCesUtf32ToUtf16(
    uint32_t utf32,
    uint16_t *utf16buf,
    uint32_t utf16max,
    uint32_t *utf16Len
)
int sceCesUtf32ToUtf16be(
    uint32_t utf32,
    uint16_t *utf16buf,
    uint32_t utf16max,
    uint32_t *utf16Len
)
int sceCesUtf32ToUtf16le(
    uint32_t utf32,
    uint16_t *utf16buf,
    uint32_t utf16max,
    uint32_t *utf16Len
)
```

Arguments

<i>utf32</i>	UTF-32 character code
<i>utf16buf</i>	Address of the buffer for receiving UTF-16 character code
<i>utf16max</i>	Maximum length (16-bit word count) of the buffer for receiving UTF-16 character code
<i>utf16Len</i>	Address of the variable for receiving UTF-16 character code length (16-bit word count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

SCE CONFIDENTIAL

Description

The function described here receives a UTF-32 character code and outputs the 16-bit code string representing that character code in UTF-16.

If the calling function is `sceCesUtf32ToUtf16()`, the output value will be written in 16-bit units.

If you wish to expressly specify the endianness of the output value, use `sceCesUtf32ToUtf16be()` for big-endian and `sceCesUtf32ToUtf16le()` for little-endian.

Specify the UTF-32 character code in `utf32`.

Specify the address for outputting the UTF-16 16-bit code in `utf16buf`.

Specify the size (16-bit word count) in which the UTF-16 16-bit code can be output in `utf16max`.

Specify the address of the variable for receiving the length of the UTF-16 character code (16-bit word count) in `utf16Len`. If a NULL pointer has been specified, this argument will be ignored.

In case of normal termination, an UTF-16 code of 2 or 4 bytes will be written in `utf16buf`, and the length of the UTF-16 code (16-bit word count) will return to `*utf16Len`.

The value stored in `*utf16Len` will coincide with the number of 16-bit words that has been written in case of normal function termination. However, it will not indicate the number of 16-bit words that has been written, but rather the length of the code (16-bit word count) represented in UTF-16. Code length will be stored also if nothing has been written due to an error.

If `SCE_CES_ERROR_INVALID_ENCODE` returns, it means that encoding has been determined to be invalid because a value equal or greater than 0x00110000, which is outside the UTF-32 valid range, has been specified in `utf32`. Nothing is written in `utf16buf`, and 0 is stored in `*utf16Len`.

If `SCE_CES_ERROR_ILLEGAL_CODE` is returned, it means the code has been determined to be illegal because a value in the range between U+D800 and U+DFFF that is reserved as surrogate area has been passed in `utf32`. In this case, the illegal code will be output to `utf16buf` and 1 will be stored in `*utf16Len` as its 16-bit word count.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been specified in `utf16buf`.

If `SCE_CES_ERROR_DST_BUFFER_END` returns, it means that the buffer length specified in `utf16max` was shorter than the length of the character code that was to be stored.

In case of an error caused by the output buffer, there will be no output to `utf16buf`, but the length of the code (16-bit word count) that was to be output will return to `*utf16Len`.

Notes

This function is multi-thread safe.

See Also

`sceCesUtf16ToUtf32()`, `sceCesUtf16beToUtf32()`, `sceCesUtf16leToUtf32()`

sceCesUtf32ToUtf8

Conversion of one character from UTF-32 to UTF-8

Definition

```
#include <ces.h>
int sceCesUtf32ToUtf8 (
    uint32_t utf32,
    uint8_t *utf8buf,
    uint32_t utf8max,
    uint32_t *utf8Len
)
```

Arguments

<i>utf32</i>	UTF-32 character code
<i>utf8buf</i>	Address of the buffer for receiving UTF-8 character code
<i>utf8max</i>	Maximum length (byte count) of the buffer for receiving UTF-8 character code
<i>utf8Len</i>	Address of the variable for receiving UTF-8 character code length (byte count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

The function receives a UTF-32 character code and outputs the 8-bit code string representing that character code in UTF-8.

Specify the UTF-32 character code in *utf32*.

Specify the address for outputting the UTF-8 8-bit code in *utf8buf*.

Specify the size (byte count) in which the UTF-8 8-bit code can be output in *utf8max*.

Specify the address of the variable for receiving the length of the UTF-8 character code (byte count) in *utf8Len*. If a NULL pointer has been specified, this argument will be ignored.

In case of normal termination, an UTF-8 code of 1 to 4 bytes will be written in *utf8buf*, and the length of the UTF-8 code (byte count) will return to **utf8Len*. The value stored in **utf8Len* will coincide with the number of bytes that has been written in case of normal function termination. However, it will not indicate the number of bytes that has been written, but rather the length of the code (byte count) represented in UTF-8. Code length will be stored also if nothing has been written due to an error.

SCE CONFIDENTIAL

If `SCE_CES_ERROR_INVALID_ENCODE` returns, it means that encoding has been determined to be invalid because a value equal or greater than 0x00110000, which is outside the UTF-32 valid range, has been specified in `utf32`. Nothing is written in `utf8buf`, and 0 is stored in `*utf8Len`.

If `SCE_CES_ERROR_ILLEGAL_CODE` is returned, it means the code has been determined to be illegal because a value in the range between U+D800 and U+DFFF that is reserved as surrogate area has been passed in `utf32`. In this case, the code will be output to `utf8buf`, and the byte count of the illegal code in UTF-8 will be stored in `*utf8Len`.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been specified in `utf8buf`.

If `SCE_CES_ERROR_DST_BUFFER_END` returns, it means that the buffer length specified in `utf8max` was shorter than the length of the character code represented in UTF-8.

In case of an error caused by the output buffer, there will be no output to `utf8buf`, but the length (byte count) of the UTF-8 code that was to be output will return to `*utf8Len`.

Notes

This function is multi-thread safe.

See Also

`sceCesUtf8ToUtf32()`

SCE CONFIDENTIAL

sceCesUtf32ToUcs2

Conversion of one character from UTF-32 to UCS-2

Definition

```
#include <ces.h>
int sceCesUtf32ToUcs2 (
    uint32_t utf32,
    uint16_t *ucs2
)
```

Arguments

utf32 UTF-32 character code
ucs2 Address for receiving UCS-2 character code

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_OUT_OF_CODE_RANGE	0x805C0024	Character outside the representable code range of the output destination encoding scheme is detected
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid

Description

The function receives a UTF-32 character code and returns a code value representing that character code in UCS-2.

Specify the UTF-32 character code in *utf32*.

Specify the address for receiving the UCS-2 character code in *ucs2*.

In case of normal termination, the same value will be written in *utf32* and **ucs2* since UTF-32 and UCS-2 both use Unicode code point scalar values.

If SCE_CES_ERROR_INVALID_ENCODE returns, it means that encoding has been determined to be invalid because a value equal or greater than 0x00110000, which is outside the UTF-32 valid range, has been specified in *utf32*. In this case, 0 will be stored in **ucs2*.

If SCE_CES_ERROR_ILLEGAL_CODE is returned, it means the code has been determined to be illegal because a value in the range between U+D800 and U+DFFF that is reserved as surrogate area has been passed in *utf32*. In this case, the illegal code value will be set in **ucs2*.

If SCE_CES_ERROR_OUT_OF_CODE_RANGE returns, it means that a valid value has been specified in *utf32*, but the range of the character code is such that it cannot be represented in the UCS-2 output destination. In other words, it means that a code value equal or greater than U+00010000 has been given to *utf32*. At this time, 0 will be stored in **ucs2*.

SCE CONFIDENTIAL

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been passed to `ucs2`. This error is only reported when no other errors are detected, and means that the function would have been successful if the output buffer had been set appropriately.

Notes

This function is multi-thread safe.

See Also

`sceCesUcs2ToUtf32()`

000004892117

sceCesUtf32beToUtf16, sceCesUtf32beToUtf16be, sceCesUtf32beToUtf16le

Conversion of one character from UTF-32BE to UTF-16(BE/LE)

Definition

```
#include <ces.h>
int sceCesUtf32beToUtf16(
    const uint32_t *utf32be,
    uint16_t *utf16buf,
    uint32_t utf16max,
    uint32_t *utf16Len
)
int sceCesUtf32beToUtf16be(
    const uint32_t *utf32be,
    uint16_t *utf16buf,
    uint32_t utf16max,
    uint32_t *utf16Len
)
int sceCesUtf32beToUtf16le(
    const uint32_t *utf32be,
    uint16_t *utf16buf,
    uint32_t utf16max,
    uint32_t *utf16Len
)
```

Arguments

<i>utf32be</i>	Address of UTF-32BE character code
<i>utf16buf</i>	Address of the buffer for receiving UTF-16 character code
<i>utf16max</i>	Maximum length (16-bit word count) of the buffer for receiving UTF-16 character code
<i>utf16Len</i>	Address of the variable for receiving UTF-16 character code length (16-bit word count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

The function described here receives a UTF-32 character code stored in big-endian format and returns a 16-bit code string representing that character code in UTF-16.

If the calling function is `sceCesUtf32beToUtf16()`, the output value will be written in 16-bit units.

If you wish to expressly specify the endianness of the output value, use

`sceCesUtf32beToUtf16be()` for big-endian and `sceCesUtf32beToUtf16le()` for little-endian.

Specify the address storing the UTF-32BE character code in `utf32be`.

Specify the address for outputting the UTF-16 16-bit code in `utf16buf`.

Specify the size (16-bit word count) in which the UTF-16 16-bit code can be output in `utf16max`.

Specify the address of the variable for receiving the length of the UTF-16 character code (16-bit word count) in `utf16Len`. If a NULL pointer has been specified, this argument will be ignored.

In case of normal termination, an UTF-16 code of 2 or 4 bytes will be written in `utf16buf`, and the length of the UTF-16 code (16-bit word count) will return to `*utf16Len`.

The value stored in `*utf16Len` will coincide with the number of 16-bit words that has been written in case of normal function termination. However, it will not indicate the number of 16-bit words that has been written, but rather the length of the code (16-bit word count) represented in UTF-16. Code length will be stored also if nothing has been written due to an error.

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed in `utf32be`. Nothing is written in `utf16buf`, and 0 is stored in `*utf16Len`.

If `SCE_CES_ERROR_INVALID_ENCODE` returns, it means that encoding has been determined to be invalid because a value equal or greater than 0x00110000, which is outside the UTF-32 valid range, has been specified in `utf32be`. Nothing is written in `utf16buf`, and 0 is stored in `*utf16Len`.

If `SCE_CES_ERROR_ILLEGAL_CODE` is returned, it means the code has been determined to be illegal because a value in the range between U+D800 and U+DFFF that is reserved as surrogate area has been passed in `utf32be`. In this case, the illegal code will be output to `utf16buf` and 1 will be stored in `*utf16Len` as its 16-bit word count.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been specified in `utf16buf`.

If `SCE_CES_ERROR_DST_BUFFER_END` returns, it means that the buffer length specified in `utf16max` was shorter than the length of the character code represented in UTF-16.

In case of an error caused by the output buffer, there will be no output to `utf16buf`, but the length of the code (16-bit word count) that was to be output will return to `*utf16Len`.

Notes

This function is multi-thread safe.

See Also

`sceCesUtf16ToUtf32be()`, `sceCesUtf16beToUtf32be()`, `sceCesUtf16leToUtf32be()`

sceCesUtf32beToUtf8

Conversion of one character from UTF-32BE to UTF-8

Definition

```
#include <ces.h>
int sceCesUtf32beToUtf8 (
    const uint32_t *utf32be,
    uint8_t *utf8buf,
    uint32_t utf8max,
    uint32_t *utf8Len
)
```

Arguments

utf32be Address of UTF-32BE character code
utf8buf Address of the buffer for receiving UTF-8 character code
utf8max Maximum length (byte count) of the buffer for receiving UTF-8 character code
utf8Len Address of the variable for receiving UTF-8 character code length (byte count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

The function receives a UTF-32 character code stored in big-endian format and returns an 8-bit code string representing that character code in UTF-8.

Specify the address storing the UTF-32BE character code in *utf32be*.

Specify the address for outputting the UTF-8 8-bit code in *utf8buf*.

Specify the size (byte count) in which the UTF-8 8-bit code can be output in *utf8max*.

Specify the address of the variable for receiving the length of the UTF-8 character code (byte count) in *utf8Len*. If a NULL pointer has been specified, this argument will be ignored.

In case of normal termination, an UTF-8 code of 1 to 4 bytes will be written in *utf8buf*, and the length of the UTF-8 code (byte count) will return to **utf8Len*. The value stored in **utf8Len* will coincide with the number of bytes that has been written in case of normal function termination. However, it will not indicate the number of bytes that has been written, but rather the length of the code (byte count) represented in UTF-8. Code length will be stored also if nothing has been written due to an error.

SCE CONFIDENTIAL

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed in `utf32be`. Nothing is written in `utf8buf`, and 0 is stored in `*utf8Len`.

If `SCE_CES_ERROR_INVALID_ENCODE` returns, it means that encoding has been determined to be invalid because a value equal or greater than 0x00110000, which is outside the UTF-32 valid range, has been specified in `utf32be`. Nothing is written in `utf8buf`, and 0 is stored in `*utf8Len`.

If `SCE_CES_ERROR_ILLEGAL_CODE` is returned, it means the code has been determined to be illegal because a value in the range between U+D800 and U+DFFF that is reserved as surrogate area has been passed in `utf32be`. In this case, the code will be output to `utf8buf`, and the byte count of the illegal code in UTF-8 will be stored in `*utf8Len`.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been specified in `utf8buf`.

If `SCE_CES_ERROR_DST_BUFFER_END` returns, it means that the buffer length specified in `utf8max` was shorter than the length of the character code represented in UTF-8.

In case of an error caused by the output buffer, there will be no output to `utf8buf`, but the length (byte count) of the UTF-8 code that was to be output will return to `*utf8Len`.

Notes

This function is multi-thread safe.

See Also

`sceCesUtf8ToUtf32be()`

SCE CONFIDENTIAL

sceCesUtf32beToUcs2

Conversion of one character from UTF-32BE to UCS-2

Definition

```
#include <ces.h>
int sceCesUtf32beToUcs2 (
    const uint32_t *utf32be,
    uint16_t *ucs2
)
```

Arguments

utf32be Address of UTF-32BE character code
ucs2 Address for receiving UCS-2 character code

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_OUT_OF_CODE_RANGE	0x805C0024	Character outside the representable code range of the output destination encoding scheme is detected
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid

Description

This function receives a UTF-32 character code stored in big-endian format and returns a code value representing that character code in UCS-2.

Specify the address storing the UTF-32BE character code in *utf32be*.

Specify the address for receiving the UCS-2 character code in *ucs2*.

If SCE_CES_ERROR_INVALID_ENCODE returns, it means that encoding has been determined to be invalid because a value equal or greater than 0x00110000, which is outside the UTF-32 valid range, has been specified in *utf32be*. In this case, 0 will be stored in **ucs2*.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed in *utf32be*. In this case, 0 will be stored in **ucs2*.

If SCE_CES_ERROR_ILLEGAL_CODE is returned, it means the code has been determined to be illegal because a value in the range between U+D800 and U+DFFF that is reserved as surrogate area has been passed in *utf32be*. In this case, the illegal code value will be set in **ucs2*.

If SCE_CES_ERROR_OUT_OF_CODE_RANGE returns, it means that a valid value has been specified in *utf32be*, but the range of the character code is such that it cannot be represented in the UCS-2 output destination. In other words, it means that a code value equal or greater than U+00010000 has been given to *utf32be*. At this time, 0 will be stored in **ucs2*.

SCE CONFIDENTIAL

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been passed to `ucs2`. This error is only reported when no other errors are detected, and means that the function would have been successful if the output buffer had been set appropriately.

Notes

This function is multi-thread safe.

See Also

`sceCesUcs2ToUtf32be()`

000004892117

SCE CONFIDENTIAL

sceCesUtf32leToUtf16, sceCesUtf32leToUtf16be, sceCesUtf32leToUtf16le

Conversion of one character from UTF-32LE to UTF-16(BE/LE)

Definition

```
#include <ces.h>
int sceCesUtf32leToUtf16(
    const uint32_t *utf32le,
    uint16_t *utf16buf,
    uint32_t utf16max,
    uint32_t *utf16Len
)
int sceCesUtf32leToUtf16be(
    const uint32_t *utf32le,
    uint16_t *utf16buf,
    uint32_t utf16max,
    uint32_t *utf16Len
)
int sceCesUtf32leToUtf16le(
    const uint32_t *utf32le,
    uint16_t *utf16buf,
    uint32_t utf16max,
    uint32_t *utf16Len
)
```

Arguments

<i>utf32le</i>	Address of UTF-32LE character code
<i>utf16buf</i>	Address of the buffer for receiving UTF-16 character code
<i>utf16max</i>	Maximum length (16-bit word count) of the buffer for receiving UTF-16 character code
<i>utf16Len</i>	Address of the variable for receiving UTF-16 character code length (16-bit word count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

The function described here receives a UTF-32 character code stored in little-endian format and returns a 16-bit code string representing that character code in UTF-16.

If the calling function is `sceCesUtf32leToUtf16()`, the output value will be written in 16-bit units.

If you wish to expressly specify the endianness of the output value, use

`sceCesUtf32leToUtf16be()` for big-endian and `sceCesUtf32leToUtf16le()` for little-endian.

Specify the address storing the UTF-32LE character code in `utf32le`.

Specify the address for outputting the UTF-16 16-bit code in `utf16buf`.

Specify the size (16-bit word count) in which the UTF-16 16-bit code can be output in `utf16max`.

Specify the address of the variable for receiving the length of the UTF-16 character code (16-bit word count) in `utf16Len`. If a NULL pointer has been specified, this argument will be ignored.

In case of normal termination, an UTF-16 code of 2 or 4 bytes will be written in `utf16buf`, and the length of the UTF-16 code (16-bit word count) will return to `*utf16Len`.

The value stored in `*utf16Len` will coincide with the number of 16-bit words that has been written in case of normal function termination. However, it will not indicate the number of 16-bit words that has been written, but rather the length of the code (16-bit word count) represented in UTF-16. Code length will be stored also if nothing has been written due to an error.

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed in `utf32le`. In this case, nothing will be written in `utf16buf`, and 0 will be stored in `*utf16Len`.

If `SCE_CES_ERROR_INVALID_ENCODE` returns, it means that encoding has been determined to be invalid because a value equal or greater than 0x00110000, which is outside the UTF-32 valid range, has been specified in `utf32le`. Nothing is written in `utf16buf`, and 0 is stored in `*utf16Len`.

If `SCE_CES_ERROR_ILLEGAL_CODE` is returned, it means the code has been determined to be illegal because a value in the range between U+D800 and U+DFFF that is reserved as surrogate area has been passed in `utf32le`. In this case, the illegal code will be output to `utf16buf` and 1 will be stored in `*utf16Len` as its 16-bit word count.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been specified in `utf16buf`.

If `SCE_CES_ERROR_DST_BUFFER_END` returns, it means that the buffer length specified in `utf16max` was shorter than the length of the character code represented in UTF-16.

In case of an error caused by the output buffer, there will be no output to `utf16buf`, but the length of the code (16-bit word count) that was to be output will return to `*utf16Len`.

Notes

This function is multi-thread safe.

See Also

`sceCesUtf16ToUtf32le()`, `sceCesUtf16beToUtf32le()`, `sceCesUtf16leToUtf32le()`

sceCesUtf32leToUtf8

Conversion of one character from UTF-32LE to UTF-8

Definition

```
#include <ces.h>
int sceCesUtf32leToUtf8 (
    const uint32_t *utf32le,
    uint8_t *utf8buf,
    uint32_t utf8max,
    uint32_t *utf8Len
)
```

Arguments

utf32le Address of UTF-32LE character code

utf8buf Address of the buffer for receiving UTF-8 character code

utf8max Maximum length (byte count) of the buffer for receiving UTF-8 character code

utf8Len Address of the variable for receiving UTF-8 character code length (byte count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

The function described here receives a UTF-32 character code stored in little-endian format and returns an 8-bit code string representing that character code in UTF-8.

Specify the address storing the UTF-32LE character code in *utf32le*.

Specify the address for outputting the UTF-8 8-bit code in *utf8buf*.

Specify the size (byte count) in which the UTF-8 8-bit code can be output in *utf8max*.

Specify the address of the variable for receiving the length of the UTF-8 character code (byte count) in *utf8Len*. If a NULL pointer has been specified, this argument will be ignored.

In case of normal termination, an UTF-8 code of 1 to 4 bytes will be written in *utf8buf*, and the length of the UTF-8 code (byte count) will return to **utf8Len*. The value stored in **utf8Len* will coincide with the number of bytes that has been written in case of normal function termination. However, it will not indicate the number of bytes that has been written, but rather the length of the code (byte count) represented in UTF-8. Code length will be stored also if nothing has been written due to an error.

SCE CONFIDENTIAL

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed in `utf32le`. In this case, nothing is written in `utf8buf`, and 0 is stored in `*utf8Len`.

If `SCE_CES_ERROR_INVALID_ENCODE` returns, it means that encoding has been determined to be invalid because a value equal or greater than 0x00110000, which is outside the UTF-32 valid range, has been specified in `utf32le`. Nothing is written in `utf8buf`, and 0 is stored in `*utf8Len`.

If `SCE_CES_ERROR_ILLEGAL_CODE` is returned, it means the code has been determined to be illegal because a value in the range between U+D800 and U+DFFF that is reserved as surrogate area has been passed in `utf32le`. In this case, the code will be output to `utf8buf`, and the byte count of the illegal code in UTF-8 will be stored in `*utf8Len`.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been specified in `utf8buf`.

If `SCE_CES_ERROR_DST_BUFFER_END` returns, it means that the buffer length specified in `utf8max` was shorter than the length of the character code represented in UTF-8.

In case of an error caused by the output buffer, there will be no output to `utf8buf`, but the length (byte count) of the UTF-8 code that was to be output will return to `*utf8Len`.

Notes

This function is multi-thread safe.

See Also

`sceCesUtf8ToUtf32le()`

SCE CONFIDENTIAL

sceCesUtf32leToUcs2

Conversion of one character from UTF-32LE to UCS-2

Definition

```
#include <ces.h>
int sceCesUtf32leToUcs2(
    const uint32_t *utf32le,
    uint16_t *ucs2
)
```

Arguments

utf32le Address of UTF-32LE character code
ucs2 Address for receiving UCS-2 character code

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_OUT_OF_CODE_RANGE	0x805C0024	Character outside the representable code range of the output destination encoding scheme is detected
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid

Description

This function receives a UTF-32 character code stored in little-endian format and returns a code value representing that character code in UCS-2.

Specify the address storing the UTF-32LE character code in *utf32le*.

Specify the address for receiving the UCS-2 character code in *ucs2*.

If SCE_CES_ERROR_INVALID_ENCODE returns, it means that encoding has been determined to be invalid because a value equal or greater than 0x00110000, which is outside the UTF-32 valid range, has been specified in *utf32le*. In this case, 0 will be stored in **ucs2*.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed in *utf32le*. In this case, 0 will be stored in **ucs2*.

If SCE_CES_ERROR_ILLEGAL_CODE is returned, it means the code has been determined to be illegal because a value in the range between U+D800 and U+DFFF that is reserved as surrogate area has been passed in *utf32le*. In this case, the illegal code value will be set in **ucs2*.

If SCE_CES_ERROR_OUT_OF_CODE_RANGE returns, it means that a valid value has been specified in *utf32le*, but the range of the character code is such that it cannot be represented in the UCS-2 output destination. In other words, it means that a code value equal or greater than U+00010000 has been given to *utf32le*. At this time, 0 will be stored in **ucs2*.

SCE CONFIDENTIAL

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been passed to `ucs2`. This error is only reported when no other errors are detected, and means that the function would have been successful if the output buffer had been set appropriately.

Notes

This function is multi-thread safe.

See Also

`sceCesUcs2ToUtf32le()`

000004892117

SCE CONFIDENTIAL

sceCesUtf16ToUtf32, sceCesUtf16ToUtf32be, sceCesUtf16ToUtf32le

Conversion of one character from UTF-16 to UTF-32 (BE/LE)

Definition

```
#include <ces.h>
int sceCesUtf16ToUtf32(
    const uint16_t *utf16addr,
    uint32_t utf16max,
    uint32_t *utf16Len,
    uint32_t *utf32
)
int sceCesUtf16ToUtf32be(
    const uint16_t *utf16addr,
    uint32_t utf16max,
    uint32_t *utf16Len,
    uint32_t *utf32
)
int sceCesUtf16ToUtf32le(
    const uint16_t *utf16addr,
    uint32_t utf16max,
    uint32_t *utf16Len,
    uint32_t *utf32
)
```

Arguments

<i>utf16addr</i>	Address storing UTF-16 character code
<i>utf16max</i>	Buffer length (16-bit word count) for which recognition of UTF-16 character code is allowed
<i>utf16Len</i>	Address of the variable for receiving successfully recognized UTF-16 character code length (16-bit word count)
<i>utf32</i>	Address for receiving UTF-32 character code

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid

Description

The function described here receives a UTF-16 character code and returns a code value representing that character code in UTF-32.

If the calling function is `sceCesUtf16ToUtf32()`, the output value will be written in 32-bit units.

If you wish to expressly specify the endianness of the output value, use `sceCesUtf16ToUtf32be()` for big-endian and `sceCesUtf16ToUtf32le()` for little-endian.

Specify the address where the UTF-16 character code is stored in `utf16addr`.

Specify the length of the buffer (16-bit word count) for which recognition of UTF-16 character code is allowed in `utf16max`.

Specify the address of the variable for receiving the length of the character code (16-bit word count) stored in UTF-16 in `utf16Len`. If a NULL pointer has been specified, this argument will be ignored.

Specify the address for receiving the UTF-32 character code in `utf32`.

In case of normal termination, the value stored in `*utf16Len` will be 1 or 2.

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed in `utf16addr`, or that 0 has been passed in `utf16max`. At this time, 0 will be stored in `*utf32`.

If `SCE_CES_ERROR_SRC_BUFFER_END` returns, it means that the character code is interrupted at the maximum limit specified in `utf16max`. In this case, the length of the character code (16-bit word count) whose recognition has been attempted will return to `*utf16Len` as a value greater than `utf16max`.

If `SCE_CES_ERROR_ILLEGAL_CODE` is returned, it means the code has been determined to be illegal because codes in the range between U+D800 and U+DFFF that is reserved as surrogate area were not used in pairs. In this case, the code value of the surrogate area will be stored as it is in `*utf32`, and 1 will return to `*utf16Len`.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been passed to `utf32`. This error is only reported when no other errors are detected, and means that the function would have been successful if the output buffer had been set appropriately.

Notes

This function is multi-thread safe.

See Also

`sceCesUtf32ToUtf16()`, `sceCesUtf32beToUtf16()`, `sceCesUtf32leToUtf16()`

SCE CONFIDENTIAL

sceCesUtf16ToUtf8

Conversion of one character from UTF-16 to UTF-8

Definition

```
#include <ces.h>
int sceCesUtf16ToUtf8(
    const uint16_t *utf16addr,
    uint32_t utf16max,
    uint32_t *utf16Len,
    uint8_t *utf8buf,
    uint32_t utf8max,
    uint32_t *utf8Len
)
```

Arguments

<i>utf16addr</i>	Address storing UTF-16 character code
<i>utf16max</i>	Buffer length (16-bit word count) for which recognition of UTF-16 character code is allowed
<i>utf16Len</i>	Address of the variable for receiving successfully recognized UTF-16 character code length (16-bit word count)
<i>utf8buf</i>	Address of the buffer for receiving UTF-8 character code
<i>utf8max</i>	Maximum length (byte count) of the buffer for receiving UTF-8 character code
<i>utf8Len</i>	Address of the variable for receiving UTF-8 character code length (byte count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

The function receives a UTF-16 character code and returns the 8-bit code string representing that character code in UTF-8.

Specify the address where the UTF-16 character code is stored in *utf16addr*.

Specify the length of the buffer (16-bit word count) for which recognition of UTF-16 character code is allowed in *utf16max*.

Specify the address of the variable for receiving the length of the character code (16-bit word count) stored in UTF-16 in *utf16Len*. If a NULL pointer has been specified, this argument will be ignored.

Specify the address for outputting the UTF-8 8-bit code in *utf8buf*.

Specify the size (byte count) in which the UTF-8 8-bit code can be output in *utf8max*.

SCE CONFIDENTIAL

Specify the address of the variable for receiving the length of the UTF-8 character code (byte count) in *utf8Len*. If a NULL pointer has been specified, this argument will be ignored.

In case of normal termination, the value stored in **utf16Len* will be 1 or 2 and the value in **utf8Len* will be from 1 to 4.

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed in *utf16addr*, or that 0 has been passed in *utf16max*. At this time, nothing will be written in *utf8addr*, and 0 will be stored in both **utf16Len* and **utf8Len*.

If `SCE_CES_ERROR_SRC_BUFFER_END` returns, it means that the character code is interrupted at the maximum limit specified in *utf16max*. In this case, the length of the character code (16-bit word count) whose recognition has been attempted will return to **utf16Len* as a value greater than *utf16max*. Nothing is written in *utf8addr*, and 0 is stored in **utf8Len*.

If `SCE_CES_ERROR_ILLEGAL_CODE` is returned, it means the code has been determined to be illegal because codes in the range between U+D800 and U+DFFF that is reserved as surrogate area were not used in pairs. In the case of this error, 1 and the result of encoding an illegal code will return to **utf16Len* and *utf8buf* respectively, and 3 will return to **utf8Len* as the byte number of the result.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been specified in *utf8buf*.

If `SCE_CES_ERROR_DST_BUFFER_END` returns, it means that the buffer length specified in *utf8max* was shorter than the length of the character code represented in UTF-8.

In case of an error caused by the output buffer, there will be no output to *utf8buf*, but the length (byte count) of the UTF-8 code that was to be output will return to **utf8Len*.

Notes

This function is multi-thread safe.

See Also

`sceCesUtf8ToUtf16()`

SCE CONFIDENTIAL

sceCesUtf16ToUcs2

Conversion of one character from UTF-16 to UCS-2

Definition

```
#include <ces.h>
int sceCesUtf16ToUcs2 (
    const uint16_t *utf16addr,
    uint32_t utf16max,
    uint32_t *utf16Len,
    uint16_t *ucs2
)
```

Arguments

<i>utf16addr</i>	Address storing UTF-16 character code
<i>utf16max</i>	Buffer length (16-bit word count) for which recognition of UTF-16 character code is allowed
<i>utf16Len</i>	Address of the variable for receiving successfully recognized UTF-16 character code length (16-bit word count)
<i>ucs2</i>	Address for receiving UCS-2 character code

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_OUT_OF_CODE_RANGE	0x805C0024	Character outside the representable code range of the output destination encoding scheme is detected
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid

Description

The function receives a UTF-16 character code and returns a code value representing that character code in UCS-2.

Specify the address where the UTF-16 character code is stored in *utf16addr*.

Specify the length of the buffer (16-bit word count) for which recognition of UTF-16 character code is allowed in *utf16max*.

Specify the address of the variable for receiving the length of the character code (16-bit word count) stored in UTF-16 in *utf16Len*. If a NULL pointer has been specified, this argument will be ignored.

Specify the address for receiving the UCS-2 character code in *ucs2*.

In case of normal termination, the value stored in **utf16Len* will be 1 or 2.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed in *utf16addr*, or that 0 has been passed in *utf16max*. At this time, 0 will be stored in **ucs2*.

©SCEI

SCE CONFIDENTIAL

If `SCE_CES_ERROR_SRC_BUFFER_END` returns, it means that the character code is interrupted at the maximum limit specified in `utf16max`. In this case, the length of the character code (16-bit word count) whose recognition has been attempted will return to `*utf16Len` as a value greater than `utf16max`.

If `SCE_CES_ERROR_ILLEGAL_CODE` is returned, it means the code has been determined to be illegal because codes in the range between U+D800 and U+DFFF that is reserved as surrogate area were not used in pairs. In this case, the code value of the surrogate area will be stored as it is in `*ucs2`, and 1 will return to `*utf16Len`.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been passed to `ucs2`. This error is only reported when no other errors are detected, and means that the function would have been successful if the output buffer had been set appropriately.

Notes

This function is multi-thread safe.

See Also

`sceCesUcs2ToUtf16()`

sceCesUtf16beToUtf32, sceCesUtf16beToUtf32be, sceCesUtf16beToUtf32le

Conversion of one character from UTF-16BE to UTF-32(BE/LE)

Definition

```
#include <ces.h>
int sceCesUtf16beToUtf32 (
    const uint16_t *utf16addr,
    uint32_t utf16max,
    uint32_t *utf16Len,
    uint32_t *utf32
)
int sceCesUtf16beToUtf32be (
    const uint16_t *utf16addr,
    uint32_t utf16max,
    uint32_t *utf16Len,
    uint32_t *utf32
)
int sceCesUtf16beToUtf32le (
    const uint16_t *utf16addr,
    uint32_t utf16max,
    uint32_t *utf16Len,
    uint32_t *utf32
)
```

Arguments

utf16addr Address storing UTF-16BE character code
utf16max Buffer length (16-bit word count) for which recognition of UTF-16BE character code is allowed
utf16Len Address of the variable for receiving successfully recognized UTF-16BE character code length (16-bit word count)
utf32 Address for receiving UTF-32 (BE/LE) character code

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid

Description

The function described here receives a UTF-16 character code stored in big-endian format and returns a code value representing that character code in UTF-32.

If the calling function is `sceCesUtf16beToUtf32()`, the output value will be written in 32-bit units.

If you wish to expressly specify the endianness of the output value, use

`sceCesUtf16beToUtf32be()` for big-endian and `sceCesUtf16beToUtf32le()` for little-endian.

Specify the address where the UTF-16BE character code is stored in `utf16addr`.

Specify the length of the buffer (16-bit word count) for which recognition of UTF-16BE character code is allowed in `utf16max`.

Specify the address of the variable for receiving the length of the character code (16-bit word count) stored in UTF-16BE in `utf16Len`. If a NULL pointer has been specified, this argument will be ignored.

Specify the address for receiving the UTF-32 character code in `utf32`.

In case of normal termination, the value stored in `*utf16Len` will be 1 or 2.

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed in `utf16addr`, or that 0 has been passed in `utf16max`. At this time, 0 will be stored in `*utf32`.

If `SCE_CES_ERROR_SRC_BUFFER_END` returns, it means that the character code is interrupted at the maximum limit specified in `utf16max`. In this case, the length of the character code (16-bit word count) whose recognition has been attempted will return to `*utf16Len` as a value greater than `utf16max`.

If `SCE_CES_ERROR_ILLEGAL_CODE` is returned, it means the code has been determined to be illegal because codes in the range between U+D800 and U+DFFF that is reserved as surrogate area were not used in pairs. In this case, the code value of the surrogate area will be stored as it is in `*utf32`, and 1 will return to `*utf16Len`.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been passed to `utf32`. This error is only reported when no other errors are detected, and means that the function would have been successful if the output buffer had been set appropriately.

Notes

This function is multi-thread safe.

See Also

`sceCesUtf32ToUtf16be()`, `sceCesUtf32beToUtf16be()`, `sceCesUtf32leToUtf16be()`

sceCesUtf16beToUtf8

Conversion of one character from UTF-16 to UTF-8

Definition

```
#include <ces.h>
int sceCesUtf16beToUtf8(
    const uint16_t *utf16addr,
    uint32_t utf16max,
    uint32_t *utf16Len,
    uint8_t *utf8buf,
    uint32_t utf8max,
    uint32_t *utf8Len
)
```

Arguments

utf16addr Address storing UTF-16BE character code

utf16max Buffer length (16-bit word count) for which recognition of UTF-16BE character code is allowed

utf16Len Address of the variable for receiving successfully recognized UTF-16BE character code length (16-bit word count)

utf8buf Address of the buffer for receiving UTF-8 character code

utf8max Maximum length (byte count) of the buffer for receiving UTF-8 character code

utf8Len Address of the variable for receiving UTF-8 character code length (byte count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

The function described here receives a UTF-16 character code stored in big-endian format and returns an 8-bit code string representing that character code in UTF-8.

Specify the address where the UTF-16BE character code is stored in *utf16addr*.

Specify the length of the buffer (16-bit word count) for which recognition of UTF-16BE character code is allowed in *utf16max*.

Specify the address of the variable for receiving the length of the character code (16-bit word count) stored in UTF-16BE in *utf16Len*. If a NULL pointer has been specified, this argument will be ignored.

Specify the address for outputting the UTF-8 8-bit code in *utf8buf*.

Specify the size (byte count) in which the UTF-8 8-bit code can be output in *utf8max*.

SCE CONFIDENTIAL

Specify the address of the variable for receiving the length of the UTF-8 character code (byte count) in *utf8Len*. If a NULL pointer has been specified, this argument will be ignored.

In case of normal termination, the value stored in **utf16Len* will be 1 or 2 and the value in **utf8Len* will be from 1 to 4.

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed in *utf16addr*, or that 0 has been passed in *utf16max*. At this time, nothing will be written in *utf8addr*, and 0 will be stored in both **utf16Len* and **utf8Len*.

If `SCE_CES_ERROR_SRC_BUFFER_END` returns, it means that the character code is interrupted at the maximum limit specified in *utf16max*. In this case, the length of the character code (16-bit word count) whose recognition has been attempted will return to **utf16Len* as a value greater than *utf16max*. Nothing is written in *utf8addr*, and 0 is stored in **utf8Len*.

If `SCE_CES_ERROR_ILLEGAL_CODE` is returned, it means the code has been determined to be illegal because codes in the range between U+D800 and U+DFFF that is reserved as surrogate area were not used in pairs. In the case of this error, 1 and the result of encoding an illegal code will return to **utf16Len* and *utf8buf* respectively, and 3 will return to **utf8Len* as the byte number of the result.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been specified in *utf8buf*.

If `SCE_CES_ERROR_DST_BUFFER_END` returns, it means that the buffer length specified in *utf8max* was shorter than the length of the character code represented in UTF-8.

In case of an error caused by the output buffer, there will be no output to *utf8buf*, but the length (byte count) of the UTF-8 code that was to be output will return to **utf8Len*.

Notes

This function is multi-thread safe.

See Also

`sceCesUtf8ToUtf16be()`

sceCesUtf16beToUcs2

Conversion of one character from UTF-16 to UCS-2

Definition

```
#include <ces.h>
int sceCesUtf16beToUcs2 (
    const uint16_t *utf16addr,
    uint32_t utf16max,
    uint32_t *utf16Len,
    uint16_t *ucs2
)
```

Arguments

utf16addr Address storing UTF-16BE character code

utf16max Buffer length (16-bit word count) for which recognition of UTF-16BE character code is allowed

utf16Len Address of the variable for receiving successfully recognized UTF-16BE character code length (16-bit word count)

ucs2 Address for receiving UCS-2 character code

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_OUT_OF_CODE_RANGE	0x805C0024	Character outside the representable code range of the output destination encoding scheme is detected
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid

Description

This function receives a UTF-16 character code stored in big-endian format and returns a code value representing that character code in UCS-2.

Specify the address where the UTF-16BE character code is stored in *utf16addr*.

Specify the length of the buffer (16-bit word count) for which recognition of UTF-16BE character code is allowed in *utf16max*.

Specify the address of the variable for receiving the length of the character code (16-bit word count) stored in UTF-16BE in *utf16Len*. If a NULL pointer has been specified, this argument will be ignored.

Specify the address for receiving the UCS-2 character code in *ucs2*.

In case of normal termination, the value stored in **utf16Len* will be 1 or 2.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed in *utf16addr*, or that 0 has been passed in *utf16max*. At this time, 0 will be stored in **ucs2*.

SCE CONFIDENTIAL

If `SCE_CES_ERROR_SRC_BUFFER_END` returns, it means that the character code is interrupted at the maximum limit specified in `utf16max`. In this case, the length of the character code (16-bit word count) whose recognition has been attempted will return to `*utf16Len` as a value greater than `utf16max`.

If `SCE_CES_ERROR_ILLEGAL_CODE` is returned, it means the code has been determined to be illegal because codes in the range between U+D800 and U+DFFF that is reserved as surrogate area were not used in pairs. In this case, the code value of the surrogate area will be stored as it is in `*ucs2`, and 1 will return to `*utf16Len`.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been passed to `ucs2`. This error is only reported when no other errors are detected, and means that the function would have been successful if the output buffer had been set appropriately.

Notes

This function is multi-thread safe.

See Also

`sceCesUcs2ToUtf16be()`

SCE CONFIDENTIAL

sceCesUtf16leToUtf32, sceCesUtf16leToUtf32be, sceCesUtf16leToUtf32le

Conversion of one character from UTF-16LE to UTF-32 (BE/LE)

Definition

```
#include <ces.h>
int sceCesUtf16leToUtf32 (
    const uint16_t *utf16addr,
    uint32_t utf16max,
    uint32_t *utf16Len,
    uint32_t *utf32
)
int sceCesUtf16leToUtf32be (
    const uint16_t *utf16addr,
    uint32_t utf16max,
    uint32_t *utf16Len,
    uint32_t *utf32
)
int sceCesUtf16leToUtf32le (
    const uint16_t *utf16addr,
    uint32_t utf16max,
    uint32_t *utf16Len,
    uint32_t *utf32
)
```

Arguments

utf16addr Address storing UTF-16LE character code
utf16max Buffer length (16-bit word count) for which recognition of UTF-16LE character code is allowed
utf16Len Address of the variable for receiving successfully recognized UTF-16LE character code length (16-bit word count)
utf32 Address for receiving UTF-32 (BE/LE) character code

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid

Description

The function described here receives a UTF-16 character code stored in little-endian format and returns a code value representing that character code in UTF-32.

If the calling function is `sceCesUtf16leToUtf32()`, the output value will be written in 32-bit units.

If you wish to expressly specify the endianness of the output value, use

`sceCesUtf16leToUtf32be()` for big-endian and `sceCesUtf16leToUtf32le()` for little-endian.

©SCEI

SCE CONFIDENTIAL

Specify the address where the UTF-16LE character code is stored in *utf16addr*.

Specify the length of the buffer (16-bit word count) for which recognition of UTF-16LE character code is allowed in *utf16max*.

Specify the address of the variable for receiving the length of the character code (16-bit word count) stored in UTF-16LE in *utf16Len*. If a NULL pointer has been specified, this argument will be ignored.

Specify the address for receiving the UTF-32 character code in *utf32*.

In case of normal termination, the value stored in **utf16Len* will be 1 or 2.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed in *utf16addr*, or that 0 has been passed in *utf16max*. At this time, 0 will be stored in **utf32*.

If SCE_CES_ERROR_SRC_BUFFER_END returns, it means that the character code is interrupted at the maximum limit specified in *utf16max*. In this case, the length of the character code (16-bit word count) whose recognition has been attempted will return to **utf16Len* as a value greater than *utf16max*.

If SCE_CES_ERROR_ILLEGAL_CODE is returned, it means the code has been determined to be illegal because codes in the range between U+D800 and U+DFFF that is reserved as surrogate area were not used in pairs. In this case, the code value of the surrogate area will be stored as it is in **utf32*, and 1 will return to **utf16Len*.

If SCE_CES_ERROR_INVALID_DST_BUFFER returns, it means that a NULL pointer has been passed to *utf32*. This error is only reported when no other errors are detected, and means that the function would have been successful if the output buffer had been set appropriately.

Notes

This function is multi-thread safe.

See Also

`sceCesUtf32ToUtf16le()`, `sceCesUtf32beToUtf16le()`, `sceCesUtf32leToUtf16le()`

SCE CONFIDENTIAL

sceCesUtf16leToUtf8

Conversion of one character from UTF-16LE to UTF-8

Definition

```
#include <ces.h>
int sceCesUtf16leToUtf8(
    const uint16_t *utf16addr,
    uint32_t utf16max,
    uint32_t *utf16Len,
    uint8_t *utf8buf,
    uint32_t utf8max,
    uint32_t *utf8Len
)
```

Arguments

<i>utf16addr</i>	Address storing UTF-16LE character code
<i>utf16max</i>	Buffer length (16-bit word count) for which recognition of UTF-16LE character code is allowed
<i>utf16Len</i>	Address of the variable for receiving successfully recognized UTF-16LE character code length (16-bit word count)
<i>utf8buf</i>	Address of the buffer for receiving UTF-8 character code
<i>utf8max</i>	Maximum length (byte count) of the buffer for receiving UTF-8 character code
<i>utf8Len</i>	Address of the variable for receiving UTF-8 character code length (byte count)

Return Values

Returns `SCE_OK (0)` as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
<code>SCE_CES_ERROR_INVALID_SRC_BUFFER</code>	0x805C0010	Specified source buffer is invalid
<code>SCE_CES_ERROR_SRC_BUFFER_END</code>	0x805C0011	Specified source buffer is insufficient
<code>SCE_CES_ERROR_ILLEGAL_CODE</code>	0x805C0015	Illegal code detected in source character code
<code>SCE_CES_ERROR_INVALID_DST_BUFFER</code>	0x805C0030	Output destination buffer is invalid
<code>SCE_CES_ERROR_DST_BUFFER_END</code>	0x805C0031	Output destination buffer is insufficient

Description

This function receives a UTF-16 character code stored in little-endian format and returns an 8-bit code string representing that character code in UTF-8.

Specify the address where the UTF-16LE character code is stored in *utf16addr*.

Specify the length of the buffer (16-bit word count) for which recognition of UTF-16LE character code is allowed in *utf16max*.

Specify the address of the variable for receiving the length of the character code (16-bit word count) stored in UTF-16LE in *utf16Len*. If a NULL pointer has been specified, this argument will be ignored.

Specify the address for outputting the UTF-8 8-bit code in *utf8buf*.

Specify the size (byte count) in which the UTF-8 8-bit code can be output in *utf8max*.

©SCEI

SCE CONFIDENTIAL

Specify the address of the variable for receiving the length of the UTF-8 character code (byte count) in *utf8Len*. If a NULL pointer has been specified, this argument will be ignored.

In case of normal termination, the value stored in **utf16Len* will be 1 or 2 and the value in **utf8Len* will be from 1 to 4.

If *SCE_CES_ERROR_INVALID_SRC_BUFFER* returns, it means that a NULL pointer has been passed in *utf16addr*, or that 0 has been passed in *utf16max*. At this time, nothing will be written in *utf8addr*, and 0 will be stored in both **utf16Len* and **utf8Len*.

If *SCE_CES_ERROR_SRC_BUFFER_END* returns, it means that the character code is interrupted at the maximum limit specified in *utf16max*. In this case, the length of the character code (16-bit word count) whose recognition has been attempted will return to **utf16Len* as a value greater than *utf16max*. Nothing is written in *utf8addr*, and 0 is stored in **utf8Len*.

If *SCE_CES_ERROR_ILLEGAL_CODE* is returned, it means the code has been determined to be illegal because codes in the range between U+D800 and U+DFFF that is reserved as surrogate area were not used in pairs. In the case of this error, 1 and the result of encoding an illegal code will return to **utf16Len* and *utf8buf* respectively, and 3 will return to **utf8Len* as the byte number of the result.

If *SCE_CES_ERROR_INVALID_DST_BUFFER* returns, it means that a NULL pointer has been specified in *utf8buf*.

If *SCE_CES_ERROR_DST_BUFFER_END* returns, it means that the buffer length specified in *utf8max* was shorter than the length of the character code represented in UTF-8.

In case of an error caused by the output buffer, there will be no output to *utf8buf*, but the length (byte count) of the UTF-8 code that was to be output will return to **utf8Len*.

Notes

This function is multi-thread safe.

See Also

`sceCesUtf8ToUtf16le()`

sceCesUtf16leToUcs2

Conversion of one character from UTF-16LE to UCS-2

Definition

```
#include <ces.h>
int sceCesUtf16leToUcs2 (
    const uint16_t *utf16addr,
    uint32_t utf16max,
    uint32_t *utf16Len,
    uint16_t *ucs2
)
```

Arguments

utf16addr Address storing UTF-16LE character code

utf16max Buffer length (16-bit word count) for which recognition of UTF-16LE character code is allowed

utf16Len Address of the variable for receiving successfully recognized UTF-16LE character code length (16-bit word count)

ucs2 Address for receiving UCS-2 character code

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_OUT_OF_CODE_RANGE	0x805C0024	Character outside the representable code range of the output destination encoding scheme is detected
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid

Description

This function receives a UTF-16 character code stored in little-endian format and returns a code value representing that character code in UCS-2.

Specify the address where the UTF-16LE character code is stored in *utf16addr*.

Specify the length of the buffer (16-bit word count) for which recognition of UTF-16LE character code is allowed in *utf16max*.

Specify the address of the variable for receiving the length of the character code (16-bit word count) stored in UTF-16LE in *utf16Len*. If a NULL pointer has been specified, this argument will be ignored.

Specify the address for receiving the UCS-2 character code in *ucs2*.

In case of normal termination, the value stored in **utf16Len* will be 1 or 2.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed in *utf16addr*, or that 0 has been passed in *utf16max*. At this time, 0 will be stored in **ucs2*.

SCE CONFIDENTIAL

If `SCE_CES_ERROR_SRC_BUFFER_END` returns, it means that the character code is interrupted at the maximum limit specified in `utf16max`. In this case, the length of the character code (16-bit word count) whose recognition has been attempted will return to `*utf16Len` as a value greater than `utf16max`.

If `SCE_CES_ERROR_ILLEGAL_CODE` is returned, it means the code has been determined to be illegal because codes in the range between U+D800 and U+DFFF that is reserved as surrogate area were not used in pairs. In this case, the code value of the surrogate area will be stored as it is in `*ucs2`, and 1 will return to `*utf16Len`.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been passed to `ucs2`. This error is only reported when no other errors are detected, and means that the function would have been successful if the output buffer had been set appropriately.

Notes

This function is multi-thread safe.

See Also

`sceCesUcs2ToUtf16le()`

SCE CONFIDENTIAL

sceCesUtf8ToUtf32, sceCesUtf8ToUtf32be, sceCesUtf8ToUtf32le

Conversion of one character from UTF-8 to UTF-32

Definition

```
#include <ces.h>
int sceCesUtf8ToUtf32(
    const uint8_t *utf8addr,
    uint32_t utf8max,
    uint32_t *utf8Len,
    uint32_t *utf32
)
int sceCesUtf8ToUtf32be(
    const uint8_t *utf8addr,
    uint32_t utf8max,
    uint32_t *utf8Len,
    uint32_t *utf32
)
int sceCesUtf8ToUtf32le(
    const uint8_t *utf8addr,
    uint32_t utf8max,
    uint32_t *utf8Len,
    uint32_t *utf32
)
```

Arguments

<i>utf8addr</i>	Address storing UTF-8 character code
<i>utf8max</i>	Buffer length (byte count) for which recognition of UTF-8 character code is allowed
<i>utf8Len</i>	Address of the variable for receiving successfully recognized UTF-8 character code length (byte count)
<i>utf32</i>	Address for receiving UTF-32 character code

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_OUT_OF_CODE_RANGE	0x805C0024	Character outside the representable code range of the output destination encoding scheme is detected
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid

SCE CONFIDENTIAL

Description

The function described here receives a UTF-8 character code and returns a code value representing that character code in UTF-32.

If the calling function is `sceCesUtf8ToUtf32()`, the output value will be written in 32-bit units.

If you wish to expressly specify the endianness of the output value, use `sceCesUtf8ToUtf32be()` for big-endian and `sceCesUtf8ToUtf32le()` for little-endian.

Specify the address where the UTF-8 character code is stored in `utf8addr`.

Specify the length of the buffer (byte count) for which recognition of UTF-8 character code is allowed in `utf8max`.

Specify the address of the variable for receiving the length (byte count) of the stored UTF-8 character code in `utf8Len`. Specify the address of the variable for receiving the length (byte count) of the stored UTF-8 character code in `utf8Len`. If a NULL pointer has been specified, this argument will be ignored.

Specify the address for receiving the UTF-32 character code in `utf32`.

In case of normal termination, the value stored in `*utf8Len` will be from 1 to 4.

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed in `utf8addr`, or that 0 has been passed in `utf8max`. At this time, 0 will be stored in `*utf32`.

If `SCE_CES_ERROR_SRC_BUFFER_END` returns, it means that the character code is interrupted at the maximum limit specified in `utf8max`. In case of this error, the character code length determined from the first UTF-8 byte will return to `*utf8Len`. Note that a higher value than that specified in `utf8max` will return.

If `SCE_CES_ERROR_INVALID_ENCODE` returns, it means that the encoding is found to be invalid because a byte string that cannot be recognized as UTF-8 has been passed in `utf8addr`. The number of bytes that were recognized successfully as UTF-8 will return to `*utf8Len` as a value between 0 and 5. 0 will be stored in `*utf32`.

If `SCE_CES_ERROR_ILLEGAL_CODE` returns, it means the code has been determined to be illegal because the byte string specified in `utf8addr` contains encoding of code in the U+D800 to U+DFFF range reserved as a surrogate area, or encoding with an unnecessarily large number of bytes (such as representing U+0000 as the 2-byte sequence C0,80). The code value obtained from decoding an invalid encode will be stored in `*utf32`.

If `SCE_CES_ERROR_OUT_OF_CODE_RANGE` returns, it means that output has been determined to be impossible because a character code equal or greater than U+00110000, which is outside the range representable in UTF-32, was encoded in `utf8addr`. At this time, a value equal or greater than 0x00110000 will be stored in `*utf32`. The value returning to `*utf8Len` will be from 4 to 6.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been passed to `utf32`. This error is only reported when no other errors are detected, and means that the function would have been successful if the output buffer had been set appropriately.

Notes

This function is multi-thread safe.

See Also

`sceCesUtf32ToUtf8()`, `sceCesUtf32beToUtf8()`, `sceCesUtf32leToUtf8()`

SCE CONFIDENTIAL

sceCesUtf8ToUtf16, sceCesUtf8ToUtf16be, sceCesUtf8ToUtf16le

Conversion of one character from UTF-8 to UTF-16

Definition

```
#include <ces.h>
int sceCesUtf8ToUtf16(
    const uint8_t *utf8addr,
    uint32_t utf8max,
    uint32_t *utf8Len,
    uint16_t *utf16buf,
    uint32_t utf16max,
    uint32_t *utf16Len
)
int sceCesUtf8ToUtf16be(
    const uint8_t *utf8addr,
    uint32_t utf8max,
    uint32_t *utf8Len,
    uint16_t *utf16buf,
    uint32_t utf16max,
    uint32_t *utf16Len
)
int sceCesUtf8ToUtf16le(
    const uint8_t *utf8addr,
    uint32_t utf8max,
    uint32_t *utf8Len,
    uint16_t *utf16buf,
    uint32_t utf16max,
    uint32_t *utf16Len
)
```

Arguments

<i>utf8addr</i>	Address storing UTF-8 character code
<i>utf8max</i>	Buffer length (byte count) for which recognition of UTF-8 character code is allowed
<i>utf8Len</i>	Address of the variable for receiving successfully recognized UTF-8 character code length (byte count)
<i>utf16buf</i>	Address of the buffer for receiving UTF-16 character code
<i>utf16max</i>	Maximum length (16-bit word count) of the buffer for receiving UTF-16 character code
<i>utf16Len</i>	Address of the variable for receiving UTF-16 character code length (16-bit word count)

Return Values

Returns `SCE_OK (0)` as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
<code>SCE_CES_ERROR_INVALID_SRC_BUFFER</code>	<code>0x805C0010</code>	Specified source buffer is invalid
<code>SCE_CES_ERROR_SRC_BUFFER_END</code>	<code>0x805C0011</code>	Specified source buffer is insufficient
<code>SCE_CES_ERROR_INVALID_ENCODE</code>	<code>0x805C0014</code>	Source encoding determined to be invalid
<code>SCE_CES_ERROR_ILLEGAL_CODE</code>	<code>0x805C0015</code>	Illegal code detected in source character code
<code>SCE_CES_ERROR_OUT_OF_CODE_RANGE</code>	<code>0x805C0024</code>	Character outside the representable code range of the output destination encoding scheme is detected
<code>SCE_CES_ERROR_INVALID_DST_BUFFER</code>	<code>0x805C0030</code>	Output destination buffer is invalid
<code>SCE_CES_ERROR_DST_BUFFER_END</code>	<code>0x805C0031</code>	Output destination buffer is insufficient

Description

The function described here receives a UTF-8 character code and returns a 16-bit code string representing that character code in UTF-16.

If the calling function is `sceCesUtf8ToUtf16()`, the output value will be written in 16-bit units.

If you wish to expressly specify the endianness of the output value, use `sceCesUtf8ToUtf16be()` for big-endian and `sceCesUtf8ToUtf16le()` for little-endian.

Specify the address where the UTF-8 character code is stored in `utf8addr`.

Specify the length of the buffer (byte count) for which recognition of UTF-8 character code is allowed in `utf8max`.

Specify the address of the variable for receiving the length (byte count) of the stored UTF-8 character code in `utf8Len`. If a NULL pointer has been specified, this argument will be ignored.

Specify the address for outputting the UTF-16 16-bit code in `utf16buf`.

Specify the size (16-bit word count) in which the UTF-16 16-bit code can be output in `utf16max`.

Specify the address of the variable for receiving the length of the UTF-16 character code (16-bit word count) in `utf16Len`. If a NULL pointer has been specified, this argument will be ignored.

In case of normal termination, the value stored in `*utf8Len` will be from 1 to 4 and the value stored in `*utf16Len` will be 1 or 2.

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed in `utf8addr`, or that 0 has been passed in `utf8max`. At this time, nothing will be written in `utf16addr`, and 0 will be stored in `*utf16Len`.

If `SCE_CES_ERROR_SRC_BUFFER_END` returns, it means that the character code is interrupted at the maximum limit specified in `utf8max`. In case of this error, the character code length determined from the first UTF-8 byte will return to `*utf8Len`. Note that a higher value than that specified in `utf8max` will return.

SCE CONFIDENTIAL

If `SCE_CES_ERROR_INVALID_ENCODE` returns, it means that the encoding is found to be invalid because a byte string that cannot be recognized as UTF-8 has been passed in `utf8addr`. The number of bytes that were recognized successfully as UTF-8 will return to `*utf8Len` as a value between 0 and 5. Nothing is written in `utf16buf`, and 0 is stored in `*utf16Len`.

If `SCE_CES_ERROR_ILLEGAL_CODE` returns, it means the code has been determined to be illegal because the byte string specified in `utf8addr` contains encoding of code in the U+D800 to U+DFFF range reserved as a surrogate area, or encoding with an unnecessarily large number of bytes (such as representing U+0000 as the 2-byte sequence C0,80). In this case, the results of encoding of the code determined to be illegal will be output to `utf16addr` and `*utf16Len`.

If `SCE_CES_ERROR_OUT_OF_CODE_RANGE` returns, it means that output has been determined to be impossible because a character code equal or greater than U+00110000, which is outside the range representable in UTF-16, was encoded in `utf8addr`. In this case, there will be no output to `utf16buf` and 0 will return to `*utf16Len`.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been specified in `utf16buf`.

If `SCE_CES_ERROR_DST_BUFFER_END` returns, it means that the buffer length specified in `utf16max` was shorter than the length of the character code represented in UTF-16.

In case of an error caused by the output buffer, there will be no output to `utf16buf`, but the length of the code (16-bit word count) that was to be output will return to `*utf16Len`.

Notes

This function is multi-thread safe.

See Also

`sceCesUtf16ToUtf8()`, `sceCesUtf16beToUtf8()`, `sceCesUtf16leToUtf8()`

sceCesUtf8ToUcs2

Conversion of one character from UTF-8 to UCS-2

Definition

```
#include <ces.h>
int sceCesUtf8ToUcs2 (
    const uint8_t *utf8addr,
    uint32_t utf8max,
    uint32_t *utf8Len,
    uint16_t *ucs2
)
```

Arguments

<i>utf8addr</i>	Address storing UTF-8 character code
<i>utf8max</i>	Buffer length (byte count) for which recognition of UTF-8 character code is allowed
<i>utf8Len</i>	Address of the variable for receiving successfully recognized UTF-8 character code length (byte count)
<i>ucs2</i>	Address for receiving UCS-2 character code

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_OUT_OF_CODE_RANGE	0x805C0024	Character outside the representable code range of the output destination encoding scheme is detected
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid

Description

The function described here receives a UTF-8 character code and returns a code value representing that character code in UCS-2.

Specify the address where the UTF-8 character code is stored in *utf8addr*.

Specify the length of the buffer (byte count) for which recognition of UTF-8 character code is allowed in *utf8max*.

Specify the address of the variable for receiving the length (byte count) of the stored UTF-8 character code in *utf8Len*. If a NULL pointer has been specified, this argument will be ignored.

Specify the address for receiving the UCS-2 character code in *ucs2*.

In case of normal termination, the value stored in **utf8Len* will be 1 to 3.

SCE CONFIDENTIAL

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed in `utf8addr`, or that 0 has been passed in `utf8max`. At this time, 0 will be stored in `*ucs2`.

If `SCE_CES_ERROR_SRC_BUFFER_END` returns, it means that the character code is interrupted at the maximum limit specified in `utf8max`. In case of this error, the character code length determined from the first UTF-8 byte will return to `*utf8Len`. Note that a higher value than that specified in `utf8max` will return.

If `SCE_CES_ERROR_INVALID_ENCODE` returns, it means that the encoding is found to be invalid because a byte string that cannot be recognized as UTF-8 has been passed in `utf8addr`. The number of bytes that were recognized successfully as UTF-8 will return to `*utf8Len` as a value between 0 and 5. 0 will be stored in `*ucs2`.

If `SCE_CES_ERROR_ILLEGAL_CODE` returns, it means the code has been determined to be illegal because the byte string specified in `utf8addr` contains encoding of code in the U+D800 to U+DFFF range reserved as a surrogate area, or encoding with an unnecessarily large number of bytes (such as representing U+0000 as the 2-byte sequence C0,80). The code value obtained from decoding an invalid encode will be stored in `*ucs2`.

If `SCE_CES_ERROR_OUT_OF_CODE_RANGE` returns, it means that output has been determined to be impossible because a character code equal or greater than U+00010000, which is outside the range representable in UCS-2, was encoded in `utf8addr`. At this time, 0 will be stored in `*ucs2`.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been passed to `ucs2`. This error is only reported when no other errors are detected, and means that the function would have been successful if the output buffer had been set appropriately.

Notes

This function is multi-thread safe.

See Also

`sceCesUcs2ToUtf8()`

sceCesUcs2ToUtf32, sceCesUcs2ToUtf32be, sceCesUcs2ToUtf32le

Conversion of one character from UCS-2 to UTF-32

Definition

```
#include <ces.h>
int sceCesUcs2ToUtf32 (
    uint16_t ucs2,
    uint32_t *utf32
)
int sceCesUcs2ToUtf32be (
    uint16_t ucs2,
    uint32_t *utf32
)
int sceCesUcs2ToUtf32le (
    uint16_t ucs2,
    uint32_t *utf32
)
```

Arguments

<i>ucs2</i>	UCS-2 character code
<i>utf32</i>	Address for receiving UTF-32 character code

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid

Description

The function described here receives a UCS-2 character code and returns a code value representing that character code in UTF-32.

If the calling function is `sceCesUcs2ToUtf32()`, the output value will be written in 32-bit units.

If you wish to expressly specify the endianness of the output value, use `sceCesUcs2ToUtf32be()` for big-endian and `sceCesUcs2ToUtf32le()` for little-endian.

Specify the UCS-2 character code in *ucs2*.

Specify the address for receiving the UTF-32 character code in *utf32*.

In case of normal termination, *ucs2* and *utf32* will have the same value because UTF-32 represents Unicode code points in 32-bit fixed width.

If SCE_CES_ERROR_ILLEGAL_CODE is returned, it means the code has been determined to be illegal because a value in the range between U+D800 and U+DFFF that is reserved as surrogate area has been passed in *ucs2*. In this case, the illegal code value will be set in **ucs2*.

SCE CONFIDENTIAL

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been passed to `utf32`. This error is only reported when no other errors are detected, and means that the function would have been successful if the output buffer had been set appropriately.

Notes

This function is multi-thread safe.

See Also

`sceCesUtf32ToUcs2()`, `sceCesUtf32beToUcs2()`, `sceCesUtf32leToUcs2()`

000004892117

sceCesUcs2ToUtf16, sceCesUcs2ToUtf16be, sceCesUcs2ToUtf16le

Conversion of one character from UCS-2 to UTF-16 (BE/LE)

Definition

```
#include <ces.h>
int sceCesUcs2ToUtf16(
    uint16_t ucs2,
    uint16_t *utf16buf,
    uint32_t utf16max,
    uint32_t *utf16Len
)
int sceCesUcs2ToUtf16be(
    uint16_t ucs2,
    uint16_t *utf16buf,
    uint32_t utf16max,
    uint32_t *utf16Len
)
int sceCesUcs2ToUtf16le(
    uint16_t ucs2,
    uint16_t *utf16buf,
    uint32_t utf16max,
    uint32_t *utf16Len
)
```

Arguments

<i>ucs2</i>	UCS-2 character code
<i>utf16buf</i>	Address of the buffer for receiving UTF-16 character code
<i>utf16max</i>	Maximum length (16-bit word count) of the buffer for receiving UTF-16 character code
<i>utf16Len</i>	Address of the variable for receiving UTF-16 character code length (16-bit word count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

The function described here receives a UCS-2 character code and returns a 16-bit code string representing that character code in UTF-16.

If the calling function is `sceCesUcs2ToUtf16()`, the output value will be written in 16-bit units.

If you wish to expressly specify the endianness of the output value, use `sceCesUcs2ToUtf16be()` for big-endian and `sceCesUcs2ToUtf16le()` for little-endian.

SCE CONFIDENTIAL

Specify the UCS-2 character code in *ucs2*.

Specify the address for outputting the UTF-16 16-bit code in *utf16buf*.

Specify the size (16-bit word count) in which the UTF-16 16-bit code can be output in *utf16max*.

Specify the address of the variable for receiving the length of the UTF-16 character code (16-bit word count) in *utf16Len*. If a NULL pointer has been specified, this argument will be ignored.

In case of normal termination, *ucs2* and **utf16buf* will have the same value and 1 will always return to **utf16Len* because UTF-16 has backward compatibility with UCS-2.

If SCE_CES_ERROR_ILLEGAL_CODE is returned, it means the code has been determined to be illegal because a value in the range between U+D800 and U+DFFF that is reserved as surrogate area has been passed in *ucs2*. In this case, the illegal code will be output to *utf16buf* and 1 will be stored in **utf16Len* as its 16-bit word count.

If SCE_CES_ERROR_INVALID_DST_BUFFER returns, it means that a NULL pointer has been specified in *utf16buf*.

If SCE_CES_ERROR_DST_BUFFER_END returns, it means that the buffer length specified in *utf16max* was shorter than the length of the character code represented in UTF-16.

In case of an error caused by the output buffer, there will be no output to *utf16buf*, but the length of the code (16-bit word count) that was to be output will return to **utf16Len*.

Notes

This function is multi-thread safe.

See Also

`sceCesUtf16ToUcs2()`, `sceCesUtf16beToUcs2()`, `sceCesUtf16leToUcs2()`

sceCesUcs2ToUtf8

Conversion of one character from UCS-2 to UTF-8

Definition

```
#include <ces.h>
int sceCesUcs2ToUtf8 (
    uint16_t ucs2,
    uint8_t *utf8buf,
    uint32_t utf8max,
    uint32_t *utf8Len
)
```

Arguments

<i>ucs2</i>	UCS-2 character code
<i>utf8buf</i>	Address of the buffer for receiving UTF-8 character code
<i>utf8max</i>	Maximum length (byte count) of the buffer for receiving UTF-8 character code
<i>utf8Len</i>	Address of the variable for receiving UTF-8 character code length (byte count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

The function described here receives a UCS-2 character code and returns an 8-bit code string representing that character code in UTF-8.

Specify the UCS-2 character code in *ucs2*.

Specify the address for outputting the UTF-8 8-bit code in *utf8buf*.

Specify the size (byte count) in which the UTF-8 8-bit code can be output in *utf8max*.

Specify the address of the variable for receiving the length of the UTF-8 character code (byte count) in *utf8Len*. If a NULL pointer has been specified, this argument will be ignored.

In case of normal termination, an UTF-8 code of 1 to 3 bytes will be written in *utf8buf*, and the length (byte count) of the UTF-8 code will return to **utf8Len*. The value stored in **utf8Len* will coincide with the number of bytes that has been written in case of normal function termination. However, it will not indicate the number of bytes that has been written, but rather the length of the code (byte count) represented in UTF-8. Code length will be stored also if nothing has been written due to an error.

If SCE_CES_ERROR_ILLEGAL_CODE is returned, it means the code has been determined to be illegal because a value in the range between U+D800 and U+DFFF that is reserved as surrogate area has been passed in *ucs2*. In this case, the code will be output to *utf8buf*, and the byte count of the illegal code in UTF-8 will be stored in **utf8Len*.

SCE CONFIDENTIAL

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been specified in `utf8buf`.

If `SCE_CES_ERROR_DST_BUFFER_END` returns, it means that the buffer length specified in `utf8max` was shorter than the length of the character code represented in UTF-8.

In case of an error caused by the output buffer, there will be no output to `utf8buf`, but the length (byte count) of the UTF-8 code that was to be output will return to `*utf8Len`.

Notes

This function is multi-thread safe.

See Also

`sceCesUtf8ToUcs2()`

000004892117

Context for Character String Conversion

SceCesUcsContext

Context for UCS character string conversion

Definition

```
#include <ces.h>
typedef struct SceCesUcsContext{
    /* omitted */
} SceCesUcsContext;
```

Description

This is a context type for UCS character string conversion.

In this library, it is used when switching UCS CES for a character string.

Before use, it is necessary to perform initialization with `sceCesUcsContextInit()`.

See Also

```
sceCesUcsContextInit(), sceCesSetUtf16StrEndian(), sceCesSetUtf32StrEndian(),
sceCesUtf32StrGetCopyLen(), sceCesUtf32StrToCopyStr(),
sceCesUtf16StrGetCopyLen(), sceCesUtf16StrToCopyStr(),
sceCesUtf8StrGetCopyLen(), sceCesUtf8StrToCopyStr(), sceCesUcs2StrGetCopyLen(),
sceCesUcs2StrToCopyStr(), sceCesUtf32StrGetUtf16Len(),
sceCesUtf32StrToUtf16Str(), sceCesUtf32StrGetUtf8Len(),
sceCesUtf32StrToUtf8Str(), sceCesUtf32StrGetUcs2Len(),
sceCesUtf32StrToUcs2Str(), sceCesUtf16StrGetUtf32Len(),
sceCesUtf16StrToUtf32Str(), sceCesUtf16StrGetUtf8Len(),
sceCesUtf16StrToUtf8Str(), sceCesUtf16StrGetUcs2Len(),
sceCesUtf16StrToUcs2Str(), sceCesUtf8StrGetUtf32Len(),
sceCesUtf8StrToUtf32Str(), sceCesUtf8StrGetUtf16Len(),
sceCesUtf8StrToUtf16Str(), sceCesUtf8StrGetUcs2Len(), sceCesUtf8StrToUcs2Str(),
sceCesUcs2StrGetUtf32Len(), sceCesUcs2StrToUtf32Str(),
sceCesUcs2StrGetUtf16Len(), sceCesUcs2StrToUtf16Str(),
sceCesUcs2StrGetUtf8Len(), sceCesUcs2StrToUtf8Str(),
sceCesUcsStrGetEncodingSize(), sceCesUcsStrConvertEncoding()
```


sceCesUcsContextInit

Initialize the context for UCS character string conversion

Definition

```
#include <ces.h>
int sceCesUcsContextInit(
    SceCesUcsContext* context,
)
```

Arguments

context Address of the context for UCS character string conversion

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns the following error code (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid

Description

Initializes the context to be used for UCS character string conversion.

The caller must prepare the context entity.

Allocate and provide the memory for the `SceCesUcsContext` type.

Also, the memory must not be freed while using `SceCesUcsContext`.

Examples

```
SceCesUcsContext Context;

sceCesUcsContextInit( &Context );
```

Notes

This function is not multi-thread safe.

Context entity must be divided by thread for multithreading.

See Also

`SceCesUcsContext`, `SceCesMbcUcsContext`

SCE CONFIDENTIAL

sceCesUcsContextInitCopy

Copy Initialization of the context for UCS character string conversion

Definition

```
#include <ces.h>
int sceCesUcsContextInitCopy (
    SceCesUcsContext* context,
    const SceCesUcsContext* context0
)
```

Arguments

context Address of the context for UCS character string conversion to be initialized
context0 Address of the context for UCS character string conversion of the copy source

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid

Description

This function initializes the context used for converting UCS character strings with the same setting status of other initialized contexts.

The caller must prepare the `SceCesUcsContext` type memory for *context*. The memory must not be freed while being used as `SceCesUcsContext`.

Specify the address of the context for UCS character string conversion that has already been initialized with `sceCesUcsContextInit()` to *context0*. The settings and status of the context specified in *context0* will be copied to the context specified in *context*.

If `SCE_CES_ERROR_INVALID_PARAMETER` returns, it means that a NULL pointer has been passed to *context* or *context0*.

Notes

This function is not multi-thread safe.

Context entity must be divided by thread for multithreading.

See Also

`SceCesUcsContext`, `sceCesUcsContextInit()`

Settings for Character Sting Conversion

sceCesSetUcsPolicyDetectBom

Set the policy for detecting Byte Order Mark

Definition

```
#include <ces.h>
int sceCesSetUcsPolicyDetectBom(
    SceCesUcsContext* context,
    int enable
)
```

Arguments

<i>context</i>	Address of the context for UCS character string conversion
<i>enable</i>	Setting value of BOM detection policy

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns the following error code (negative value) in case of error.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid

Description

If the reversal value of U+FEFF(ZERO WIDTH NON-BLAKING SPACE/ Byte Order Mark) is detected in the input character string of a function having UTF-16 and UTF-32 character strings as input arguments, the present function will enable/disable detection as BOM (Byte Order Mark) in the context.

If BOM (Byte Order Mark) has been detected, character string recognition will be performed by switching endiannes after the byte order has been determined as reversed.

Specify the address of the context for UCS character string conversion initialized with `sceCesUcsContextInit()` in *context*.

Being a macro function, this function allows you to specify the address of UCS and multi-byte character set conversion context in addition to the address of the `SceCesUcsContext` type.

Specify one of the following values in *enable*.

Value	Description
SCE_CES_DETECT_ENABLE	Enables detection as BOM (default value)
SCE_CES_DETECT_DISABLE	Disables detection as BOM

The setting value of the context immediately after performing initialization with `sceCesUcsContextInit()` will be `SCE_CES_DETECT_ENABLE`.

If `SCE_CES_ERROR_INVALID_PARAMETER` returns, it means that a NULL pointer has been passed to *context*, or that the value specified in *endian* has been determined to be invalid because outside the specifiable range.

SCE CONFIDENTIAL

Examples

Example 1

```
SceCesUcsContext ctx;  
sceCesUcsContextInit( &ctx );  
sceCesSetUcsPolicyDetectBom( &ctx, SCE_CES_DETECT_ENABLE );  
:
```

Example 2

```
SceCesMbcsUcsContext mctx;  
sceCesMbcsUcsContextInit( &mctx );  
sceCesSetUcsPolicyDetectBom( &mctx, SCE_CES_DETECT_ENABLE );  
:
```

Notes

This function is not multi-thread safe.

Context entity must be divided by thread for multithreading.

See Also

SceCesUcsContext, SceCesMbcsUcsContext, sceCesUcsContextInit(),
sceCesUtf32StrGetCopyLen(), sceCesUtf32StrToCopyStr(),
sceCesUtf16StrGetCopyLen(), sceCesUtf16StrToCopyStr(),
sceCesUtf8StrGetCopyLen(), sceCesUtf8StrToCopyStr(), sceCesUcs2StrGetCopyLen(),
sceCesUcs2StrToCopyStr(), sceCesUtf32StrGetUtf16Len(),
sceCesUtf32StrToUtf16Str(), sceCesUtf32StrGetUtf8Len(),
sceCesUtf32StrToUtf8Str(), sceCesUtf32StrGetUcs2Len(),
sceCesUtf32StrToUcs2Str(), sceCesUtf16StrGetUtf32Len(),
sceCesUtf16StrToUtf32Str(), sceCesUtf16StrGetUtf8Len(),
sceCesUtf16StrToUtf8Str(), sceCesUtf16StrGetUcs2Len(),
sceCesUtf16StrToUcs2Str()

SCE CONFIDENTIAL

sceCesSetUcsPolicyOutputBom

Set Byte Order Mark output policy

Definition

```
#include <ces.h>
int sceCesSetUcsPolicyOutputBom(
    SceCesUcsContext* context,
    int enable
)
```

Arguments

<i>context</i>	Address of the context for UCS character string conversion
<i>enable</i>	BOM output policy setting value

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns the following error code (negative value) in case of error.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid

Description

This function sets in the context the character output policy of U+FEFF(ZERO WIDTH NON-BLANKING SPACE/Byte Order Mark) for functions that output Unicode character strings.

Specify the address of the context for UCS character string conversion initialized with `sceCesUcsContextInit()` in *context*.

Being a macro function, this function allows you to specify the address of UCS and multi-byte character set conversion context in addition to the address of the `SceCesUcsContext` type.

Specify one of the following values in *enable*.

Value	Description
SCE_CES_OUTPUT_DISABLE	Disables output. All U+FEFF code points in input character strings are excluded from output.
SCE_CES_OUTPUT_ENABLE	Enables output. Output follows input character strings (default value)
SCE_CES_OUTPUT_TOP_CUT	Outputs without BOM. U+FEFF code points at the start of character strings are excluded.
SCE_CES_OUTPUT_TOP_SET	Outputs with BOM U+FEFF code points will be added to the start of character strings if not present.

While U+FEFF can be used as BOM, as a character it is also a “zero width non-breaking space”, and it is unclear whether it is used as the former or the latter. Be careful not to exclude it in inappropriate cases.

The setting value immediately after initialization is performed with `sceCesUcsContextInit()` will be `SCE_CES_OUTPUT_ENABLE`.

SCE CONFIDENTIAL

If `SCE_CES_ERROR_INVALID_PARAMETER` returns, it means that a NULL pointer has been passed to *context*, or that the value specified in *endian* has been determined to be invalid because outside the specifiable range.

Examples

```
SceCesUcsContext Context;

sceCesUcsContextInit( &Context );
sceCesSetUcsPolicyOutputBom( &Context, SCE_CES_OUTPUT_DISABLE );
//Convert the UTF-16 character string to a UTF-8 character string that does not
include BOM.
sceCesUtf16StrToUtf8Str( &Context,
                        utf16str, utf16max, &utf16Len,
                        utf8buf , utf8max, &utf8Len );
```

Notes

This function is not multi-thread safe.

Context entity must be divided by thread for multithreading.

See Also

```
SceCesUcsContext, SceCesMbcUcsContext, sceCesUcsContextInit(),
sceCesUtf32StrGetCopyLen(), sceCesUtf32StrToCopyStr(),
sceCesUtf16StrGetCopyLen(), sceCesUtf16StrToCopyStr(),
sceCesUtf8StrGetCopyLen(), sceCesUtf8StrToCopyStr(), sceCesUcs2StrGetCopyLen(),
sceCesUcs2StrToCopyStr(), sceCesUtf32StrGetUtf16Len(),
sceCesUtf32StrToUtf16Str(), sceCesUtf32StrGetUtf8Len(),
sceCesUtf32StrToUtf8Str(), sceCesUtf32StrGetUcs2Len(),
sceCesUtf32StrToUcs2Str(), sceCesUtf16StrGetUtf32Len(),
sceCesUtf16StrToUtf32Str(), sceCesUtf16StrGetUtf8Len(),
sceCesUtf16StrToUtf8Str(), sceCesUtf16StrGetUcs2Len(),
sceCesUtf16StrToUcs2Str(), sceCesUtf8StrGetUtf32Len(),
sceCesUtf8StrToUtf32Str(), sceCesUtf8StrGetUtf16Len(),
sceCesUtf8StrToUtf16Str(), sceCesUtf8StrGetUcs2Len(), sceCesUtf8StrToUcs2Str(),
sceCesUcs2StrGetUtf32Len(), sceCesUcs2StrToUtf32Str(),
sceCesUcs2StrGetUtf16Len(), sceCesUcs2StrToUtf16Str(),
sceCesUcs2StrGetUtf8Len(), sceCesUcs2StrToUtf8Str()
```

sceCesSetUtf16StrEndian, sceCesSetUtf32StrEndian

Set UTF character string endianness

Definition

```
#include <ces.h>
int sceCesSetUtf16StrEndian (
    SceCesUcsContext* context,
    SceCesEndianParam srcStrEndian,
    SceCesEndianParam dstStrEndian
)
int sceCesSetUtf32StrEndian (
    SceCesUcsContext* context,
    SceCesEndianParam srcStrEndian,
    SceCesEndianParam dstStrEndian
)
```

Arguments

<i>context</i>	Address of the context for UCS character string conversion
<i>srcStrEndian</i>	Endianness of the input-side UTF character string handled by character string function
<i>dstStrEndian</i>	Endianness of the output-side UTF character string handled by character string function

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns the following error code (negative value) in case of error.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid

Description

The function described here sets in the specified context how UTF character string endianness is treated.

`sceCesSetUtf16StrEndian()` sets UTF-16 character string endianness, while `sceCesSetUtf32StrEndian()` sets UTF-32 character string endianness.

Specify the address of a context that can handle UCS encoding schemes in *context*.

Being a macro function, this function allows you to specify the address of UCS and multi-byte character set conversion context in addition to the address of the `SceCesUcsContext` type.

Specify one of the following values in *srcStrEndian* and *dstStrEndian*.

Value	Description
SCE_CES_ENDIAN_BE	Big-endian
SCE_CES_ENDIAN_LE	Little-endian
SCE_CES_ENDIAN_SYS	Simplified specification for selecting the same endianness as the system. (set so that the byte order will be the same as when performing memory write of 16-bit and 32-bit values.)

SCE CONFIDENTIAL

The setting specified with `srcStrEndian` will be referenced by character string processing functions containing UTF-16 or UTF-32 input, and will be used for the recognition of the endianness of input character strings. However, if BOM detection is enabled, the endianness indicated by the BOM will be prioritized. In other words, endianness will follow this setting if BOM is not detected.

If you wish to fix input character string endianness as that specified with `srcStrEndian`, disable BOM detection with `sceCesSetUcsPolicyDetectBom()`.

The setting specified with `dstStrEndian` will be referenced by character string processing functions containing UTF-16 or UTF-32 output, and character string output will always be performed in accordance with the value specified in `dstStrEndian`.

If `SCE_CES_ERROR_INVALID_PARAMETER` returns, it means that an error has been found because NULL pointer has been passed in `context`, or that an appropriate value has not been passed in `srcStrEndian` and `dstStrEndian`.

Examples

```
// Convert from UTF-16BE to an endianness that can easily be handled by the system
SceCesUcsContext Context;
sceCesUcsContextInit( &Context );
sceCesSetUcsPolicyDetectBom( &Context, SCE_CES_DETECT_DISABLE );
sceCesSetUtf16StrEndian( &Context, SCE_CES_ENDIAN_BE, SCE_CES_ENDIAN_SYS );
sceCesUtf16StrGetCopyLen( &Context, utf16beAddr, utf16beMax, &utf16beLen,
                                                                    &utf16Len );

uint32_t  utf16max = utf16Len + 1;
uint16_t* utf16str = malloc( utf16max * sizeof(uint16_t) );
if ( utf16str ) {
    // Copy of a character string with code check and endianness conversion
    sceCesUtf16StrToCopyStr( &Context, utf16be, utf16beMax, &utf16beLen,
                            utf16str, utf16max, &utf16Len );
}
```

Notes

This function is not multi-thread safe.

Context entity must be divided by thread for multithreading.

See Also

```
SceCesUcsContext, SceCesEndianParam, sceCesUcsContextInit(),
sceCesUtf32StrGetCopyLen(), sceCesUtf32StrToCopyStr(),
sceCesUtf16StrGetCopyLen(), sceCesUtf16StrToCopyStr(),
sceCesUtf32StrGetUtf16Len(), sceCesUtf32StrToUtf16Str(),
sceCesUtf32StrGetUtf8Len(), sceCesUtf32StrToUtf8Str(),
sceCesUtf32StrGetUcs2Len(), sceCesUtf32StrToUcs2Str(),
sceCesUtf16StrGetUtf32Len(), sceCesUtf16StrToUtf32Str(),
sceCesUtf16StrGetUtf8Len(), sceCesUtf16StrToUtf8Str(),
sceCesUtf16StrGetUcs2Len(), sceCesUtf16StrToUcs2Str(),
sceCesUtf8StrGetUtf32Len(), sceCesUtf8StrToUtf32Str(),
sceCesUtf8StrGetUtf16Len(), sceCesUtf8StrToUtf16Str(),
sceCesUcs2StrGetUtf32Len(), sceCesUcs2StrToUtf32Str(),
sceCesUcs2StrGetUtf16Len(), sceCesUcs2StrToUtf16Str()
```


Encoding Scheme Conversion and Copy of Character Strings

sceCesUtf32StrToUtf16Str

Convert from UTF-32 character string to UTF-16 character string

Definition

```
#include <ces.h>
int sceCesUtf32StrToUtf16Str(
    SceCesUcsContext* context,
    const uint32_t *utf32str,
    uint32_t utf32max,
    uint32_t *utf32Len,
    uint16_t *utf16buf,
    uint32_t utf16max,
    uint32_t *utf16Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>utf32str</i>	Address of the buffer storing UTF-32 character string
<i>utf32max</i>	Size (32-bit word count) of the buffer storing UTF-32 character string
<i>utf32Len</i>	Address of the variable for receiving UTF-32 character string length (32-bit word count)
<i>utf16buf</i>	Address of the buffer for receiving UTF-16 character string
<i>utf16max</i>	Size (16-bit word count) of the buffer for receiving UTF-16 character string
<i>utf16Len</i>	Address of the variable for receiving UTF-16 character string length (16-bit word count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

This function converts UTF-32 character strings to UTF-16 character strings.

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

Specify the address of the buffer storing the UTF-32 character string in *utf32str*.

SCE CONFIDENTIAL

Specify the size (32-bit word count) of the buffer storing the UTF-32 character string in *utf32max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (32-bit word count) of the UTF-32 character string used for conversion to UTF-16 will return to *utf32Len*. The NULL termination character will not be included.

In *utf16buf*, specify the address of the buffer storing the character string specified in *utf32str* and represented in UTF-16. NULL termination is always guaranteed for the buffer passed to this argument.

Specify the size (16-bit word count) allowed for writing to *utf16buf* in *utf16max*. Specify a size taking into account the part for writing the NULL termination character.

The length (16-bit word count) of the character string successfully converted to UTF-16 will return to *utf16Len*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *utf32max*, and that all processing has been completed normally.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed to *utf32str*. 0 will be in both *utf32Len* and *utf16Len*.

If SCE_CES_ERROR_INVALID_ENCODE returns, it means that encoding has been determined to be invalid because a value equal or greater than 0x00110000, which is outside the UTF-32 valid range, has been found in the character string specified in *utf32str*.

If SCE_CES_ERROR_ILLEGAL_CODE returns, it means that the code has been determined to be illegal because code in the U+D800 to U+DFFF range reserved as a surrogate area was found in the character string specified in *utf32str*.

If SCE_CES_ERROR_INVALID_DST_BUFFER returns, it means that a NULL pointer has been passed to *utf16buf*.

If SCE_CES_ERROR_DST_BUFFER_END returns, it means that the buffer specified as output destination has run out, and that the character string could not be output completely.

The value that returns to *utf32Len* and *utf16Len* in case of an error is the value up to the character processed successfully.

Use `sceCesUtf32StrGetUtf16Len()` if you only want to look up output character string length.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`sceCesUcsContextInit()`, `sceCesUtf32StrGetUtf16Len()`,
`sceCesUtf16StrToUtf32Str()`

SCE CONFIDENTIAL

sceCesUtf32StrToUtf8Str

Convert from UTF-32 character string to UTF-8 character string

Definition

```
#include <ces.h>
int sceCesUtf32StrToUtf8Str(
    SceCesUcsContext* context,
    const uint32_t *utf32str,
    uint32_t utf32max,
    uint32_t *utf32Len,
    uint8_t *utf8buf,
    uint32_t utf8max,
    uint32_t *utf8Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>utf32str</i>	Address of the buffer storing UTF-32 character string
<i>utf32max</i>	Size (32-bit word count) of the buffer storing UTF-32 character string
<i>utf32Len</i>	Address of the variable for receiving UTF-32 character string length (32-bit word count)
<i>utf8buf</i>	Address of the buffer for receiving UTF-8 character string
<i>utf8max</i>	Size (byte count) of the buffer for receiving UTF-8 character string
<i>utf8Len</i>	Address of the variable for receiving UTF-8 character string length (byte count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

This function converts UTF-32 character strings to UTF-8 character strings.

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

Specify the address of the buffer storing the UTF-32 character string in *utf32str*.

Specify the size (32-bit word count) of the buffer storing the UTF-32 character string in *utf32max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (32-bit word count) of the UTF-32 character string used for conversion to UTF-8 will return to *utf32Len*. The NULL termination character will not be included.

SCE CONFIDENTIAL

In *utf8buf*, specify the address of the buffer storing the character string specified in *utf32str* and represented in UTF-8. NULL termination is always guaranteed for the buffer passed to this argument. Specify the size (byte count) allowed for writing to *utf8buf* in *utf8max*. Specify a size taking into account the part for writing the NULL termination character.

The length (byte count) of the character string successfully converted to UTF-8 will return to *utf8Len*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *utf32max*, and that all processing has been completed normally.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed to *utf32str*. 0 is stored in both *utf32Len* and *utf8Len*.

If SCE_CES_ERROR_INVALID_ENCODE returns, it means that encoding has been determined to be invalid because a value equal or greater than 0x00110000, which is outside the UTF-32 valid range, has been found in the character string specified in *utf32str*.

If SCE_CES_ERROR_ILLEGAL_CODE returns, it means that the code has been determined to be illegal because code in the U+D800 to U+DFFF range reserved as a surrogate area was found in the character string specified in *utf32str*.

If SCE_CES_ERROR_INVALID_DST_BUFFER returns, it means that a NULL pointer has been passed to *utf8buf*.

If SCE_CES_ERROR_DST_BUFFER_END returns, it means that the buffer specified as output destination has run out, and that the character string could not be output completely.

The value that returns to *utf32Len* and *utf8Len* in case of an error is the value up to the character processed successfully.

Use `sceCesUtf32StrGetUtf8Len()` if you only want to look up output character string length.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`sceCesUcsContextInit()`, `sceCesUtf32StrGetUtf8Len()`, `sceCesUtf8StrToUtf32Str()`

SCE CONFIDENTIAL

sceCesUtf32StrToUcs2Str

Convert from UTF-32 character string to UCS-2 character string

Definition

```
#include <ces.h>
int sceCesUtf32StrToUcs2Str (
    SceCesUcsContext* context,
    const uint32_t *utf32str,
    uint32_t utf32max,
    uint32_t *utf32Len,
    uint16_t *ucs2buf,
    uint32_t ucs2max,
    uint32_t *ucs2Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>utf32str</i>	Address of the buffer storing UTF-32 character string
<i>utf32max</i>	Size (32-bit word count) of the buffer storing UTF-32 character string
<i>utf32Len</i>	Address of the variable for receiving UTF-32 character string length (32-bit word count)
<i>ucs2buf</i>	Address of the buffer for receiving UCS-2 character string
<i>ucs2max</i>	Size (16-bit word count) of the buffer for receiving UCS-2 character string
<i>ucs2Len</i>	Address of the variable for receiving UCS-2 character string length (16-bit word count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_OUT_OF_CODE_RANGE	0x805C0024	Character outside the representable code range of the output destination encoding scheme is detected
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

This function converts UTF-32 character strings to UCS-2 character strings.

(This function treats UCS-2 arrays as character strings terminating with U+0000.)

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

Specify the address of the buffer storing the UTF-32 character string in *utf32str*.

SCE CONFIDENTIAL

Specify the size (32-bit word count) of the buffer storing the UTF-32 character string in *utf32max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (32-bit word count) of the UTF-32 character string used for conversion to UCS-2 will return to *utf32Len*. The NULL termination character will not be included.

In *ucs2buf*, specify the address of the buffer storing the character string specified in *utf32str* and represented in UCS-2. NULL termination is always guaranteed for the buffer passed to this argument.

Specify the size (16-bit word count) allowed for writing to *ucs2buf* in *ucs2max*. Specify a size taking into account the part for writing the NULL termination character.

The length (16-bit word count) of the character string successfully converted to UCS-2 will return to *ucs2Len*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *utf32max*, and that all processing has been completed normally.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed to *utf32str*. 0 is stored in both *utf32Len* and *ucs2Len*.

If SCE_CES_ERROR_INVALID_ENCODE returns, it means that encoding has been determined to be invalid because a value equal or greater than 0x00110000, which is outside the UTF-32 valid range, has been found in the character string specified in *utf32str*.

If SCE_CES_ERROR_ILLEGAL_CODE returns, it means that the code has been determined to be illegal because code in the U+D800 to U+DFFF range reserved as a surrogate area was found in the character string specified in *utf32str*.

If SCE_CES_ERROR_OUT_OF_CODE_RANGE returns, it means that a character code equal or greater than U+00010000, which is outside the range that can be represented with UCS-2, was contained in the character string specified in *utf32str*.

If SCE_CES_ERROR_INVALID_DST_BUFFER returns, it means that a NULL pointer has been passed to *ucs2buf*.

If SCE_CES_ERROR_DST_BUFFER_END returns, it means that the buffer specified as output destination has run out, and that the character string could not be output completely.

The value that returns to *utf32Len* and *ucs2Len* in case of an error is the value up to the character processed successfully.

Use `sceCesUtf32StrGetUcs2Len()` if you only want to look up output character string length.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`sceCesUcsContextInit()`, `sceCesUtf32StrGetUcs2Len()`, `sceCesUcs2StrToUtf32Str()`

SCE CONFIDENTIAL

sceCesUtf16StrToUtf32Str

Convert from UTF-16 character string to UTF-32 character string

Definition

```
#include <ces.h>
int sceCesUtf16StrToUtf32Str(
    SceCesUcsContext* context,
    const uint16_t *utf16str,
    uint32_t utf16max,
    uint32_t *utf16Len,
    uint32_t *utf32buf,
    uint32_t utf32max,
    uint32_t *utf32Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>utf16str</i>	Address of the buffer storing UTF-16 character string
<i>utf16max</i>	Size (16-bit word count) of the buffer storing UTF-16 character string
<i>utf16Len</i>	Address of the variable for receiving UTF-16 character string length (16-bit word count)
<i>utf32buf</i>	Address of the buffer for receiving UTF-32 character string
<i>utf32max</i>	Size (32-bit word count) of the buffer for receiving UTF-32 character string
<i>utf32Len</i>	Address of the variable for receiving UTF-32 character string length (32-bit word count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

This function converts UTF-16 character strings to UTF-32 character strings.

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

Specify the address of the buffer storing the UTF-16 character string in *utf16str*.

Specify the size (16-bit word count) of the buffer storing the UTF-16 character string in *utf16max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (16-bit word count) of the UTF-16 character string used for conversion to UTF-32 will return to *utf16Len*. The NULL termination character will not be included.

SCE CONFIDENTIAL

In *utf32buf*, specify the address of the buffer storing the character string specified in *utf16str* and represented in UTF-32. NULL termination is always guaranteed for the buffer passed to this argument. Specify the size (32-bit word count) allowed for writing to *utf32buf* in *utf32max*. Specify a size taking into account the part for writing the NULL termination character.

The length (32-bit word count) of the character string successfully converted to UTF-32 will return to *utf32Len*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *utf16max*, and that all processing has been completed normally.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed to *utf16str*. 0 is stored in both *utf16Len* and *utf32Len*.

If SCE_CES_ERROR_SRC_BUFFER_END returns, it means that the character string specified in *utf16str* has been interrupted in the midst of a code representing one character due to limitation by *utf16max*.

If SCE_CES_ERROR_ILLEGAL_CODE returns, it means the code has been determined to be illegal because codes in the range between U+D800 and U+DFFF that is reserved as surrogate area were not used in pairs in the character string specified in *utf16str*.

If SCE_CES_ERROR_INVALID_DST_BUFFER returns, it means that a NULL pointer has been passed to *utf32buf*.

If SCE_CES_ERROR_DST_BUFFER_END returns, it means that the buffer specified as output destination has run out, and that the character string could not be output completely.

The value that returns to *utf16Len* and *utf32Len* in case of an error is the value up to the character processed successfully.

Use `sceCesUtf16StrGetUtf32Len()` if you only want to look up output character string length.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

```
sceCesUcsContextInit(), sceCesUtf16StrGetUtf32Len(),
sceCesUtf32StrToUtf16Str()
```


sceCesUtf16StrToUtf8Str

Convert from UTF-16 character string to UTF-8 character string

Definition

```
#include <ces.h>
int sceCesUtf16StrToUtf8Str(
    SceCesUcsContext* context,
    const uint16_t *utf16str,
    uint32_t utf16max,
    uint32_t *utf16Len,
    uint8_t *utf8buf,
    uint32_t utf8max,
    uint32_t *utf8Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>utf16str</i>	Address of the buffer storing UTF-16 character string
<i>utf16max</i>	Size (16-bit word count) of the buffer storing UTF-16 character string
<i>utf16Len</i>	Address of the variable for receiving UTF-16 character string length (16-bit word count)
<i>utf8buf</i>	Address of the buffer for receiving UTF-8 character string
<i>utf8max</i>	Size (byte count) of the buffer for receiving UTF-8 character string
<i>utf8Len</i>	Address of the variable for receiving UTF-8 character string length (byte count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

This function converts UTF-16 character strings to UTF-8 character strings.

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

Specify the address of the buffer storing the UTF-16 character string in *utf16str*.

Specify the size (16-bit word count) of the buffer storing the UTF-16 character string in *utf16max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (16-bit word count) of the UTF-16 character string used for conversion to UTF-8 will return to *utf16Len*. The NULL termination character will not be included.

SCE CONFIDENTIAL

In *utf8buf*, specify the address of the buffer storing the character string specified in *utf8str* and represented in UTF-8. NULL termination is always guaranteed for the buffer passed to this argument. Specify the size (byte count) allowed for writing to *utf8buf* in *utf8max*. Specify a size taking into account the part for writing the NULL termination character.

The length (byte count) of the character string successfully converted to UTF-8 will return to *utf8Len*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *utf16max*, and that all processing has been completed normally.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed to *utf16str*. 0 is stored in both *utf16Len* and *utf8Len*.

If SCE_CES_ERROR_SRC_BUFFER_END returns, it means that the character string specified in *utf16str* has been interrupted in the midst of a code representing one character due to limitation by *utf16max*.

If SCE_CES_ERROR_ILLEGAL_CODE returns, it means the code has been determined to be illegal because codes in the range between U+D800 and U+DFFF that is reserved as surrogate area were not used in pairs in the character string specified in *utf16str*.

If SCE_CES_ERROR_INVALID_DST_BUFFER returns, it means that a NULL pointer has been passed to *utf8buf*.

If SCE_CES_ERROR_DST_BUFFER_END returns, it means that the buffer specified as output destination has run out, and that the character string could not be output completely.

The value that returns to *utf16Len* and *utf8Len* in case of an error is the value up to the character processed successfully.

Use `sceCesUtf16StrGetUtf8Len()` if you only want to look up output character string length.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`sceCesUcsContextInit()`, `sceCesUtf16StrGetUtf8Len()`, `sceCesUtf8StrToUtf16Str()`

sceCesUtf16StrToUcs2Str

Convert from UTF-16 character string to UCS-2 character string

Definition

```
#include <ces.h>
int sceCesUtf16StrToUcs2Str (
    SceCesUcsContext* context,
    const uint16_t *utf16str,
    uint32_t utf16max,
    uint32_t *utf16Len,
    uint16_t *ucs2buf,
    uint32_t ucs2max,
    uint32_t *ucs2Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>utf16str</i>	Address of the buffer storing UTF-16 character string
<i>utf16max</i>	Size (16-bit word count) of the buffer storing UTF-16 character string
<i>utf16Len</i>	Address of the variable for receiving UTF-16 character string length (16-bit word count)
<i>ucs2buf</i>	Address of the buffer for receiving UCS-2 character string
<i>ucs2max</i>	Size (16-bit word count) of the buffer for receiving UCS-2 character string
<i>ucs2Len</i>	Address of the variable for receiving UCS-2 character string length (16-bit word count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_OUT_OF_CODE_RANGE	0x805C0024	Character outside the representable code range of the output destination encoding scheme is detected
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

This function converts UTF-16 character strings to UCS-2 character strings.

(This function treats UCS-2 arrays as character strings terminating with U+0000.)

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

Specify the address of the buffer storing the UTF-16 character string in *utf16str*.

SCE CONFIDENTIAL

Specify the size (16-bit word count) of the buffer storing the UTF-16 character string in *utf16max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (16-bit word count) of the UTF-16 character string used for conversion to UCS-2 will return to *utf16Len*. The NULL termination character will not be included.

In *ucs2buf*, specify the address of the buffer storing the character string specified in *utf16str* and represented in UCS-2. NULL termination is always guaranteed for the buffer passed to this argument.

Specify the size (16-bit word count) allowed for writing to *ucs2buf* in *ucs2max*. Specify a size taking into account the part for writing the NULL termination character.

The length (16-bit word count) of the character string successfully converted to UCS-2 will return to *ucs2Len*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *utf16max*, and that all processing has been completed normally.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed to *utf16str*. 0 is stored in both *utf16Len* and *ucs2Len*.

If SCE_CES_ERROR_SRC_BUFFER_END returns, it means that the character string specified in *utf16str* has been interrupted in the midst of a code representing one character due to limitation by *utf16max*.

If SCE_CES_ERROR_ILLEGAL_CODE is returned, it means the code has been determined to be illegal because codes in the range between U+D800 and U+DFFF that is reserved as surrogate area were not used in pairs.

If SCE_CES_ERROR_OUT_OF_CODE_RANGE returns, it means that a character code equal or greater than U+00010000, which is outside the range that can be represented with UCS-2, was contained in the character string specified in *utf16str*.

If SCE_CES_ERROR_INVALID_DST_BUFFER returns, it means that a NULL pointer has been passed to *ucs2buf*.

If SCE_CES_ERROR_DST_BUFFER_END returns, it means that the buffer specified as output destination has run out, and that the character string could not be output completely.

The value that returns to *utf16Len* and *ucs2Len* in case of an error is the value up to the character processed successfully.

Use `sceCesUtf16StrGetUcs2Len()` if you only want to look up output character string length.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`sceCesUcsContextInit()`, `sceCesUtf16StrGetUcs2Len()`, `sceCesUcs2StrToUtf16Str()`

SCE CONFIDENTIAL

sceCesUtf8StrToUtf32Str

Convert from UTF-8 character string to UTF-32 character string

Definition

```
#include <ces.h>
int sceCesUtf8StrToUtf32Str(
    SceCesUcsContext* context,
    const uint8_t *utf8str,
    uint32_t utf8max,
    uint32_t *utf8Len,
    uint32_t *utf32buf,
    uint32_t utf32max,
    uint32_t *utf32Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>utf8str</i>	Address of the buffer storing UTF-8 character string
<i>utf8max</i>	Size (byte count) of the buffer storing UTF-8 character string
<i>utf8Len</i>	Address of the variable for receiving UTF-8 character string length (byte count)
<i>utf32buf</i>	Address of the buffer for receiving UTF-32 character string
<i>utf32max</i>	Size (32-bit word count) of the buffer for receiving UTF-32 character string
<i>utf32Len</i>	Address of the variable for receiving UTF-32 character string length (32-bit word count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_OUT_OF_CODE_RANGE	0x805C0024	Character outside the representable code range of the output destination encoding scheme is detected
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

This function converts UTF-8 character strings to UTF-32 character strings.

UTF-32 endianness can be specified with `sceCesSetUtf32StrEndian()`.

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

©SCEI

Specify the address of the buffer storing the UTF-8 character string in *utf8str*.

Specify the size (byte count) of the buffer storing the UTF-8 character string in *utf8max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (byte count) of the UTF-8 character string used for conversion to UTF-32 will return to *utf8Len*. The NULL termination character will not be included.

In *utf32buf*, specify the address of the buffer storing the character string specified in *utf8str* and represented in UTF-32. NULL termination is always guaranteed for the buffer passed to this argument.

Specify the size (32-bit word count) allowed for writing to *utf32buf* in *utf32max*. Specify a size taking into account the part for writing the NULL termination character.

The length (32-bit word count) of the character string successfully converted to UTF-32 will return to *utf32Len*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *utf8max*, and that all processing has been completed normally.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed to *utf8str*. 0 is stored in both *utf8Len* and *utf32Len*.

If SCE_CES_ERROR_SRC_BUFFER_END returns, it means that the character string specified in *utf8str* has been interrupted in the midst of a code representing one character due to limitation by *utf8max*.

If SCE_CES_ERROR_INVALID_ENCODE returns, it means encoding has been determined to be invalid because the character string specified in *utf8str* contains byte strings that cannot be recognized as UTF-8.

If SCE_CES_ERROR_ILLEGAL_CODE returns, it means that the code has been found invalid because the character string specified in *utf8str* contains codes in the U+D800 to U+DFFF range reserved as a surrogate area, or encoding with an unnecessarily large number of bytes (such as representing U+0000 as the 2-byte sequence C0,80).

If SCE_CES_ERROR_OUT_OF_CODE_RANGE returns, it means that the character string specified in *utf8str* contained character code equal or greater than U+00110000 that cannot be handled by UTF-32.

If SCE_CES_ERROR_INVALID_DST_BUFFER returns, it means that a NULL pointer has been passed to *utf32buf*.

If SCE_CES_ERROR_DST_BUFFER_END returns, it means that the buffer specified as output destination has run out, and that the character string could not be output completely.

The value that returns to *utf8Len* and *utf32Len* in case of an error is the value up to the character processed successfully.

Use `sceCesUtf8StrGetUtf32Len()` if you only want to look up output character string length.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`sceCesUcsContextInit()`, `sceCesUtf8StrGetUtf32Len()`, `sceCesUtf32StrToUtf8Str()`

SCE CONFIDENTIAL

sceCesUtf8StrToUtf16Str

Convert from UTF-8 character string to UTF-16 character string

Definition

```
#include <ces.h>
int sceCesUtf8StrToUtf16Str(
    SceCesUcsContext* context,
    const uint8_t *utf8str,
    uint32_t utf8max,
    uint32_t *utf8Len,
    uint16_t *utf16buf,
    uint32_t utf16max,
    uint32_t *utf16Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>utf8str</i>	Address of the buffer storing UTF-8 character string
<i>utf8max</i>	Size (byte count) of the buffer storing UTF-8 character string
<i>utf8Len</i>	Address of the variable for receiving UTF-8 character string length (byte count)
<i>utf16buf</i>	Address of the buffer for receiving UTF-16 character string
<i>utf16max</i>	Size (16-bit word count) of the buffer for receiving UTF-16 character string
<i>utf16Len</i>	Address of the variable for receiving UTF-16 character string length (16-bit word count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_OUT_OF_CODE_RANGE	0x805C0024	Character outside the representable code range of the output destination encoding scheme is detected
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

This function converts UTF-8 character strings to UTF-16 character strings.

UTF-16 endianness can be specified with `sceCesSetUtf16StrEndian()`.

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

©SCEI

Specify the address of the buffer storing the UTF-8 character string in *utf8str*.

Specify the size (byte count) of the buffer storing the UTF-8 character string in *utf8max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (byte count) of the UTF-8 character string used for conversion to UTF-16 will return to *utf8Len*. The NULL termination character will not be included.

In *utf16buf*, specify the address of the buffer storing the character string specified in *utf8str* and represented in UTF-16. NULL termination is always guaranteed for the buffer passed to this argument.

Specify the size (16-bit word count) allowed for writing to *utf16buf* in *utf16max*. Specify a size taking into account the part for writing the NULL termination character.

The length (16-bit word count) of the character string successfully converted to UTF-16 will return to *utf16Len*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *utf8max*, and that all processing has been completed normally.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed to *utf8str*. 0 is stored in both *utf8Len* and *utf16Len*.

If SCE_CES_ERROR_SRC_BUFFER_END returns, it means that the character string specified in *utf8str* has been interrupted in the midst of a code representing one character due to limitation by *utf8max*.

If SCE_CES_ERROR_INVALID_ENCODE returns, it means encoding has been determined to be invalid because the character string specified in *utf8str* contains byte strings that cannot be recognized as UTF-8.

If SCE_CES_ERROR_ILLEGAL_CODE returns, it means that the code has been found invalid because the character string specified in *utf8str* contains codes in the U+D800 to U+DFFF range reserved as a surrogate area, or encoding with an unnecessarily large number of bytes (such as representing U+0000 as the 2-byte sequence C0,80).

If SCE_CES_ERROR_OUT_OF_CODE_RANGE returns, it means that a character code equal or greater than U+00110000, which is outside the range that can be represented with UTF-16, was contained in the character string specified in *utf8str*.

If SCE_CES_ERROR_INVALID_DST_BUFFER returns, it means that a NULL pointer has been passed to *utf16buf*.

If SCE_CES_ERROR_DST_BUFFER_END returns, it means that the buffer specified as output destination has run out, and that the character string could not be output completely.

The value that returns to *utf8Len* and *utf16Len* in case of an error is the value up to the character processed successfully.

Use `sceCesUtf8StrGetUtf16Len()` if you only want to look up output character string length.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`sceCesUcsContextInit()`, `sceCesUtf8StrGetUtf16Len()`, `sceCesUtf16StrToUtf8Str()`

SCE CONFIDENTIAL

sceCesUtf8StrToUcs2Str

Convert from UTF-8 character string to UCS-2 character string

Definition

```
#include <ces.h>
int sceCesUtf8StrToUcs2Str(
    SceCesUcsContext* context,
    const uint8_t *utf8str,
    uint32_t utf8max,
    uint32_t *utf8Len,
    uint16_t *ucs2buf,
    uint32_t ucs2max,
    uint32_t *ucs2Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>utf8str</i>	Address of the buffer storing UTF-8 character string
<i>utf8max</i>	Size (byte count) of the buffer storing UTF-8 character string
<i>utf8Len</i>	Address of the variable for receiving UTF-8 character string length (byte count)
<i>ucs2buf</i>	Address of the buffer for receiving UCS-2 character string
<i>ucs2max</i>	Size (16-bit word count) of the buffer for receiving UCS-2 character string
<i>ucs2Len</i>	Address of the variable for receiving UCS-2 character string length (16-bit word count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_OUT_OF_CODE_RANGE	0x805C0024	Character outside the representable code range of the output destination encoding scheme is detected
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

This function converts UTF-8 character strings to UCS-2 character strings.
(This function treats UCS-2 arrays as character strings terminating with U+0000.)

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

Specify the address of the buffer storing the UTF-8 character string in *utf8str*.

Specify the size (byte count) of the buffer storing the UTF-8 character string in *utf8max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (byte count) of the UTF-8 character string used for conversion to UCS-2 will return to *utf8Len*. The NULL termination character will not be included.

In *ucs2buf*, specify the address of the buffer storing the character string specified in *utf8str* and represented in UCS-2. NULL termination is always guaranteed for the buffer passed to this argument.

Specify the size (16-bit word count) allowed for writing to *ucs2buf* in *ucs2max*. Specify a size taking into account the part for writing the NULL termination character.

The length (16-bit word count) of the character string successfully converted to UCS-2 will return to *ucs2Len*. The NULL termination character will not be included.

If `SCE_OK` returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *utf8max*, and that all processing has been completed normally.

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed to *utf8str*. 0 is stored in both *utf8Len* and *ucs2Len*.

If `SCE_CES_ERROR_SRC_BUFFER_END` returns, it means that the character string specified in *utf8str* has been interrupted in the midst of a code representing one character due to limitation by *utf8max*.

If `SCE_CES_ERROR_INVALID_ENCODE` returns, it means encoding has been determined to be invalid because the character string specified in *utf8str* contains byte strings that cannot be recognized as UTF-8.

If `SCE_CES_ERROR_ILLEGAL_CODE` returns, it means that the code has been found invalid because the character string specified in *utf8str* contains codes in the U+D800 to U+DFFF range reserved as a surrogate area, or encoding with an unnecessarily large number of bytes (such as representing U+0000 as the 2-byte sequence C0,80).

If `SCE_CES_ERROR_OUT_OF_CODE_RANGE` returns, it means that a character code equal or greater than U+00010000, which is outside the range that can be represented with UCS-2, was contained in the character string specified in *utf8str*.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been passed to *ucs2buf*.

If `SCE_CES_ERROR_DST_BUFFER_END` returns, it means that the buffer specified as output destination has run out, and that the character string could not be output completely.

The value that returns to *utf8Len* and *ucs2Len* in case of an error is the value up to the character processed successfully.

Use `sceCesUtf8StrGetUcs2Len()` if you only want to look up output character string length.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`sceCesUcsContextInit()`, `sceCesUtf8StrGetUcs2Len()`, `sceCesUcs2StrToUtf8Str()`

SCE CONFIDENTIAL

sceCesUcs2StrToUtf32Str

Convert from UCS-2 character string to UTF-32 character string

Definition

```
#include <ces.h>
int sceCesUcs2StrToUtf32Str(
    SceCesUcsContext* context,
    const uint16_t *ucs2str,
    uint32_t ucs2max,
    uint32_t *ucs2Len,
    uint32_t *utf32buf,
    uint32_t utf32max,
    uint32_t *utf32Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>ucs2str</i>	Address of the buffer storing UCS-2 character string
<i>ucs2max</i>	Size (16-bit word count) of the buffer storing UCS-2 character string
<i>ucs2Len</i>	Address of the variable for receiving UCS-2 character string length (16-bit word count)
<i>utf32buf</i>	Address of the buffer for receiving UTF-32 character string
<i>utf32max</i>	Size (32-bit word count) of the buffer for receiving UTF-32 character string
<i>utf32Len</i>	Address of the variable for receiving UTF-32 character string length (32-bit word count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

This function converts UCS-2 character strings to UTF-32 character strings.

(This function treats UCS-2 code arrays as character strings terminating with U+0000.)

UTF-32 endianness can be specified with `sceCesSetUtf32StrEndian()`.

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

Specify the address of the buffer storing the UCS-2 character string in *ucs2str*.

Specify the size (16-bit word count) of the buffer storing the UCS-2 character string in *ucs2max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified. In this case, *ucs2str*'s termination must always be U+0000.

SCE CONFIDENTIAL

The length (16-bit word count) of the UCS-2 character string used for conversion to UTF-32 will return to *ucs2Len*. The NULL termination character will not be included.

In *utf32buf*, specify the address of the buffer storing the character string specified in *ucs2str* and represented in UTF-32. NULL termination is always guaranteed for the buffer passed to this argument.

Specify the size (32-bit word count) allowed for writing to *utf32buf* in *utf32max*. Specify a size taking into account the part for writing the NULL termination character.

The length (32-bit word count) of the character string successfully converted to UTF-32 will return to *utf32Len*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *ucs2max*, and that all processing has been completed normally.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed to *ucs2str*. 0 is stored in both *ucs2Len* and *utf32Len*.

If SCE_CES_ERROR_ILLEGAL_CODE returns, it means that the code has been determined to be illegal because code in the U+D800 to U+DFFF range reserved as a surrogate area was found in the character string specified in *ucs2str*.

If SCE_CES_ERROR_INVALID_DST_BUFFER returns, it means that a NULL pointer has been passed to *utf32buf*.

If SCE_CES_ERROR_DST_BUFFER_END returns, it means that the buffer specified as output destination has run out, and that the character string could not be output completely.

The value that returns to *ucs2Len* and *utf32Len* in case of an error is the value up to the character processed successfully.

Use `sceCesUcs2StrGetUtf32Len()` if you only want to look up output character string length.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`sceCesUcsContextInit()`, `sceCesUcs2StrGetUtf32Len()`, `sceCesUtf32StrToUcs2Str()`

sceCesUcs2StrToUtf16Str

Convert from UCS-2 character string to UTF-16 character string

Definition

```
#include <ces.h>
int sceCesUcs2StrToUtf16Str(
    SceCesUcsContext* context,
    const uint16_t *ucs2str,
    uint32_t ucs2max,
    uint32_t *ucs2Len,
    uint16_t *utf16buf,
    uint32_t utf16max,
    uint32_t *utf16Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>ucs2str</i>	Address of the buffer storing UCS-2 character string
<i>ucs2max</i>	Size (16-bit word count) of the buffer storing UCS-2 character string
<i>ucs2Len</i>	Address of the variable for receiving UCS-2 character string length (16-bit word count)
<i>utf16buf</i>	Address of the buffer for receiving UTF-16 character string
<i>utf16max</i>	Size (16-bit word count) of the buffer for receiving UTF-16 character string
<i>utf16Len</i>	Address of the variable for receiving UTF-16 character string length (16-bit word count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

This function converts UCS-2 character strings to UTF-16 character strings.

(This function treats UCS-2 code arrays as character strings terminating with U+0000.)

UTF-16 endianness can be specified with `sceCesSetUtf16StrEndian()`.

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

Specify the address of the buffer storing the UCS-2 character string in *ucs2str*.

Specify the size (16-bit word count) of the buffer storing the UCS-2 character string in *ucs2max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified. In this case, *ucs2str*'s termination must always be U+0000.

The length (16-bit word count) of the UCS-2 character string used for conversion to UTF-16 will return to *ucs2Len*. The NULL termination character will not be included.

SCE CONFIDENTIAL

In *utf16buf*, specify the address of the buffer storing the character string specified in *ucs2str* and represented in UTF-16. NULL termination is always guaranteed for the buffer passed to this argument. Specify the size (16-bit word count) allowed for writing to *utf16buf* in *utf16max*. Specify a size taking into account the part for writing the NULL termination character.

The length (16-bit word count) of the character string successfully converted to UTF-16 will return to *utf16Len*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *ucs2max*, and that all processing has been completed normally.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed to *ucs2str*. 0 is stored in both *ucs2Len* and *utf8Len*.

If SCE_CES_ERROR_ILLEGAL_CODE returns, it means that the code has been determined to be illegal because code in the U+D800 to U+DFFF range reserved as a surrogate area was found in the character string specified in *ucs2str*.

If SCE_CES_ERROR_INVALID_DST_BUFFER returns, it means that a NULL pointer has been passed to *utf16buf*.

If SCE_CES_ERROR_DST_BUFFER_END returns, it means that the buffer specified as output destination has run out, and that the character string could not be output completely.

The value that returns to *ucs2Len* and *utf16Len* in case of an error is the value up to the character processed successfully.

Use `sceCesUcs2StrGetUtf16Len()` if you only want to look up output character string length.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`sceCesUcsContextInit()`, `sceCesUcs2StrGetUtf16Len()`, `sceCesUtf16StrToUcs2Str()`

sceCesUcs2StrToUtf8Str

Convert from UCS-2 character string to UTF-8 character string

Definition

```
#include <ces.h>
int sceCesUcs2StrToUtf8Str(
    SceCesUcsContext* context,
    const uint16_t *ucs2str,
    uint32_t ucs2max,
    uint32_t *ucs2Len,
    uint8_t *utf8buf,
    uint32_t utf8max,
    uint32_t *utf8Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>ucs2str</i>	Address of the buffer storing UCS-2 character string
<i>ucs2max</i>	Size (16-bit word count) of the buffer storing UCS-2 character string
<i>ucs2Len</i>	Address of the variable for receiving UCS-2 character string length (16-bit word count)
<i>utf8buf</i>	Address of the buffer for receiving UTF-8 character string
<i>utf8max</i>	Size (byte count) of the buffer for receiving UTF-8 character string
<i>utf8Len</i>	Address of the variable for receiving UTF-8 character string length (byte count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

This function converts UCS-2 character strings to UTF-8 character strings.

(This function treats UCS-2 code arrays as character strings terminating with U+0000.)

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

Specify the address of the buffer storing the UCS-2 character string in *ucs2str*.

Specify the size (16-bit word count) of the buffer storing the UCS-2 character string in *ucs2max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified. In this case, *ucs2str*'s termination must always be U+0000.

The length (16-bit word count) of the UCS-2 character string used for conversion to UTF-8 will return to *ucs2Len*. The NULL termination character will not be included.

SCE CONFIDENTIAL

In *utf8buf*, specify the address of the buffer storing the character string specified in *ucs2str* and represented in UTF-8. NULL termination is always guaranteed for the buffer passed to this argument. Specify the size (byte count) allowed for writing to *utf8buf* in *utf8max*. Specify a size taking into account the part for writing the NULL termination character.

The length (byte count) of the character string successfully converted to UTF-8 will return to *utf8Len*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *ucs2max*, and that all processing has been completed normally.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed to *ucs2str*. 0 is stored in both *ucs2Len* and *utf8Len*.

If SCE_CES_ERROR_ILLEGAL_CODE returns, it means that the code has been determined to be illegal because code in the U+D800 to U+DFFF range reserved as a surrogate area was found in the character string specified in *ucs2str*.

If SCE_CES_ERROR_INVALID_DST_BUFFER returns, it means that a NULL pointer has been passed to *utf8buf*.

If SCE_CES_ERROR_DST_BUFFER_END returns, it means that the buffer specified as output destination has run out, and that the character string could not be output completely.

The value that returns to *ucs2Len* and *utf8Len* in case of an error is the value up to the character processed successfully.

Use `sceCesUcs2StrGetUtf8Len()` if you only want to look up output character string length.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`sceCesUcsContextInit()`, `sceCesUcs2StrGetUtf8Len()`, `sceCesUtf8StrToUcs2Str()`

SCE CONFIDENTIAL

sceCesUtf32StrToCopyStr

Copy UTF-32 character strings

Definition

```
#include <ces.h>
int sceCesUtf32StrToCopyStr (
    SceCesUcsContext* context,
    const uint32_t *utf32str,
    uint32_t utf32max,
    uint32_t *utf32Len,
    uint32_t *copyBuf,
    uint32_t copyMax,
    uint32_t *copyLen
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>utf32str</i>	Address of the buffer storing UTF-32 character string
<i>utf32max</i>	Size (32-bit word count) of the buffer storing UTF-32 character string
<i>utf32Len</i>	Address of the variable for receiving UTF-32 character string length (32-bit word count)
<i>copyBuf</i>	Address of the buffer for receiving the copied character string
<i>copyMax</i>	Size (32-bit word count) of the buffer for receiving the copied character string
<i>copyLen</i>	Address of the variable for receiving the length of the copied character string (32-bit word count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

This function performs copying of character strings with UTF-32 code check.

It is possible to perform copying with endianness conversion by specifying character string endianness in advance with `sceCesSetUtf32StrEndian()`.

BOM can be added or removed through specification with `sceCesSetUcsPolicyOutputBom()`.

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

Specify the address of the buffer storing the UTF-32 character string in *utf32str*.

SCE CONFIDENTIAL

Specify the size (32-bit word count) of the buffer storing the UTF-32 character string in *utf32max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (32-bit word count) of the UTF-32 character string used for copying will return to *utf32Len*. The NULL termination character will not be included.

Specify the address of the buffer for copying the character string specified with *utf32str* in *copyBuf*. NULL termination is always guaranteed for the buffer passed to this argument.

Specify the size (32-bit word count) allowed for writing to *copyBuf* in *copyMax*. Specify a size taking into account the part for writing the NULL termination character.

The length (32-bit word count) of the character string successfully copied to *copyBuf* will return to *copyLen*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *utf32max*, and that all processing has been completed normally.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed to *utf32str*. 0 is stored in both *utf32Len* and *copyLen*.

If SCE_CES_ERROR_INVALID_ENCODE returns, it means that encoding has been determined to be invalid because a value equal or greater than 0x00110000, which is outside the UTF-32 valid range, has been found in the character string specified in *utf32str*.

If SCE_CES_ERROR_ILLEGAL_CODE returns, it means that the code has been determined to be illegal because code in the U+D800 to U+DFFF range reserved as a surrogate area was found in the character string specified in *utf32str*.

If SCE_CES_ERROR_INVALID_DST_BUFFER returns, it means that a NULL pointer has been passed to *copyBuf*.

If SCE_CES_ERROR_DST_BUFFER_END returns, it means that the buffer specified as output destination has run out, and that the character string could not be output completely.

The value that returns to *utf32Len* and *copyLen* in case of an error is the value up to the character processed successfully.

Use `sceCesUtf32StrGetCopyLen()` if you only want to look up output character string length.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`sceCesUcsContextInit()`, `sceCesUtf32StrGetCopyLen()`

sceCesUtf16StrToCopyStr

Copy UTF-16 character strings

Definition

```
#include <ces.h>
int sceCesUtf16StrToCopyStr(
    SceCesUcsContext* context,
    const uint16_t *utf16str,
    uint32_t utf16max,
    uint32_t *utf16Len,
    uint16_t *copyBuf,
    uint32_t copyMax,
    uint32_t *copyLen
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>utf16str</i>	Address of the buffer storing UTF-16 character string
<i>utf16max</i>	Size (16-bit word count) of the buffer storing UTF-16 character string
<i>utf16Len</i>	Address of the variable for receiving UTF-16 character string length (16-bit word count)
<i>copyBuf</i>	Address of the buffer for receiving the copied character string
<i>copyMax</i>	Size (16-bit word count) of the buffer for receiving the copied character string
<i>copyLen</i>	Address of the variable for receiving the length of the copied character string (16-bit word count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

This function performs copying of character strings with UTF-16 code check.

It is possible to perform copying with endianness conversion by specifying character string endianness in advance with `sceCesSetUtf16StrEndian()`.

BOM can be added or removed through specification with `sceCesSetUcsPolicyOutputBom()`.

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

Specify the address of the buffer storing the UTF-16 character string in *utf16str*.

SCE CONFIDENTIAL

Specify the size (16-bit word count) of the buffer storing the UTF-16 character string in *utf16max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (16-bit word count) of the UTF-16 character string used for copying will return to *utf16Len*. The NULL termination character will not be included.

Specify the address of the buffer for copying the character string specified with *utf16str* in *copyBuf*. NULL termination is always guaranteed for the buffer passed to this argument.

Specify the size (16-bit word count) allowed for writing to *copyBuf* in *copyMax*. Specify a size taking into account the part for writing the NULL termination character.

The length (16-bit word count) of the character string successfully copied to *copyBuf* will return to *copyLen*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *utf16max*, and that all processing has been completed normally.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed to *utf16str*. 0 is stored in both *utf16Len* and *copyLen*.

If SCE_CES_ERROR_SRC_BUFFER_END returns, it means that the character string specified in *utf16str* has been interrupted in the midst of a code representing one character due to limitation by *utf16max*.

If SCE_CES_ERROR_ILLEGAL_CODE is returned, it means the code has been determined to be illegal because codes in the range between U+D800 and U+DFFF that is reserved as surrogate area were not used in pairs.

If SCE_CES_ERROR_INVALID_DST_BUFFER returns, it means that a NULL pointer has been passed to *copyBuf*.

If SCE_CES_ERROR_DST_BUFFER_END returns, it means that the buffer specified as output destination has run out, and that the character string could not be output completely.

The value that returns to *utf16Len* and *copyLen* in case of an error is the value up to the character processed successfully.

Use `sceCesUtf16StrGetCopyLen()` if you only want to look up output character string length.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`sceCesUcsContextInit()`, `sceCesUtf16StrGetCopyLen()`

SCE CONFIDENTIAL

sceCesUtf8StrToCopyStr

Copy UTF-8 character strings

Definition

```
#include <ces.h>
int sceCesUtf8StrToCopyStr (
    SceCesUcsContext* context,
    const uint8_t *utf8str,
    uint32_t utf8max,
    uint32_t *utf8Len,
    uint8_t *copyBuf,
    uint32_t copyMax,
    uint32_t *copyLen
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>utf8str</i>	Address of the buffer storing UTF-8 character string
<i>utf8max</i>	Size (byte count) of the buffer storing UTF-8 character string
<i>utf8Len</i>	Address of the variable for receiving UTF-8 character string length (byte count)
<i>copyBuf</i>	Address of the buffer for receiving the copied character string
<i>copyMax</i>	Size (byte count) of the buffer for receiving the copied character string
<i>copyLen</i>	Address of the variable for receiving the length of the copied character string (byte count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

This function performs copying of character strings with UTF-8 character string code check.

You can specify whether to add or remove BOM when copying with `sceCesSetUcsPolicyOutputBom()`.

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

Specify the address of the buffer storing the UTF-8 character string in *utf8str*.

SCE CONFIDENTIAL

Specify the size (byte count) of the buffer storing the UTF-8 character string in *utf8max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (byte count) of the UTF-8 character string used for copying will return to *utf8Len*. The NULL termination character will not be included.

Specify the address of the buffer for copying the character string specified with *utf8str* in *copyBuf*. NULL termination is always guaranteed for the buffer passed to this argument.

Specify the size (byte count) allowed for writing to *copyBuf* in *copyMax*. Specify a size taking into account the part for writing the NULL termination character.

The length (byte count) of the character string successfully copied to *copyBuf* will return to *copyLen*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *utf8max*, and that all processing has been completed normally.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed to *utf8str*. 0 is stored in both *utf8Len* and *copyLen*.

If SCE_CES_ERROR_SRC_BUFFER_END returns, it means that the character string specified in *utf8str* has been interrupted in the midst of a code representing one character due to limitation by *utf8max*.

If SCE_CES_ERROR_INVALID_ENCODE returns, it means encoding has been determined to be invalid because the character string specified in *utf8str* contains byte strings that cannot be recognized as UTF-8.

If SCE_CES_ERROR_ILLEGAL_CODE returns, it means that the code has been found invalid because the character string specified in *utf8str* contains codes in the U+D800 to U+DFFF range reserved as a surrogate area, or encoding with an unnecessarily large number of bytes (such as representing U+0000 as the 2-byte sequence C0,80).

If SCE_CES_ERROR_INVALID_DST_BUFFER returns, it means that a NULL pointer has been passed to *copyBuf*.

If SCE_CES_ERROR_DST_BUFFER_END returns, it means that the buffer specified as output destination has run out, and that the character string could not be output completely.

The value that returns to *utf8Len* and *copyLen* in case of an error is the value up to the character processed successfully.

Use `sceCesUtf8StrGetCopyLen()` if you only want to look up output character string length.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`sceCesUcsContextInit()`, `sceCesUtf8StrGetCopyLen()`

SCE CONFIDENTIAL

sceCesUcs2StrToCopyStr

Copy UCS-2 character strings

Definition

```
#include <ces.h>
int sceCesUcs2StrToCopyStr (
    SceCesUcsContext* context,
    const uint16_t *ucs2str,
    uint32_t ucs2max,
    uint32_t *ucs2Len,
    uint16_t *copyBuf,
    uint32_t copyMax,
    uint32_t *copyLen
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>ucs2str</i>	Address of the buffer storing UCS-2 character string
<i>ucs2max</i>	Size (16-bit word count) of the buffer storing UCS-2 character string
<i>ucs2Len</i>	Address of the variable for receiving UCS-2 character string length (16-bit word count)
<i>copyBuf</i>	Address of the buffer for receiving the copied character string
<i>copyMax</i>	Size (32-bit word count) of the buffer for receiving the copied character string
<i>copyLen</i>	Address of the variable for receiving the length of the copied character string (16-bit word count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

This function performs copying of character strings with UCS-2 character string code check.

(This function treats UCS-2 code arrays as character strings terminating with U+0000.)

You can specify whether to add or remove BOM character when copying with `sceCesSetUcsPolicyOutputBom()`.

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

Specify the address of the buffer storing the UCS-2 character string in *ucs2str*.

Specify the size (16-bit word count) of the buffer storing the UCS-2 character string in *ucs2max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified. In this case, *ucs2str*'s termination must always be U+0000.

SCE CONFIDENTIAL

The length (16-bit code count) of the UCS-2 character string used for copying will return to *ucs2Len*. The NULL termination character will not be included.

Specify the address of the buffer for copying the character string specified with *ucs2str* in *copyBuf*. Specify the size (16-bit word count) allowed for writing to *copyBuf* in *copyMax*. Specify a size taking into account the part for writing the NULL termination character.

The length (16-bit word count) of the character string successfully copied to *copyBuf* will return to *copyLen*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *ucs2max*, and that all processing has been completed normally.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed to *ucs2str*. 0 is stored in both *ucs2Len* and *copyLen*.

If SCE_CES_ERROR_ILLEGAL_CODE returns, it means that the code has been determined to be illegal because code in the U+D800 to U+DFFF range reserved as a surrogate area was found in the character string specified in *ucs2str*.

If SCE_CES_ERROR_INVALID_DST_BUFFER returns, it means that a NULL pointer has been passed to *copyBuf*.

If SCE_CES_ERROR_DST_BUFFER_END returns, it means that the buffer specified as output destination has run out, and that the character string could not be output completely.

The value that returns to *ucs2Len* and *copyLen* in case of an error is the value up to the character processed successfully.

Use `sceCesUcs2StrGetCopyLen()` if you only want to look up character string length.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`sceCesUcsContextInit()`, `sceCesUcs2StrGetCopyLen()`

Retrieving Character String Length by Encoding Scheme

sceCesUtf32StrGetUtf16Len

Retrieve character string length of UTF-32 character strings represented in UTF-16

Definition

```
#include <ces.h>
int sceCesUtf32StrGetUtf16Len (
    SceCesUcsContext* context,
    const uint32_t *utf32str,
    uint32_t utf32max,
    uint32_t *utf32Len,
    uint32_t *utf16Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>utf32str</i>	Address of the buffer storing UTF-32 character string
<i>utf32max</i>	Size (32-bit word count) of the buffer storing UTF-32 character string
<i>utf32Len</i>	Address of the variable for receiving UTF-32 character string length (32-bit word count)
<i>utf16Len</i>	Address of the variable for receiving UTF-16 character string length (16-bit word count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code

Description

This function retrieves the length of UTF-32 character strings when changed to UTF-16 character strings.

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

Specify the address of the buffer storing the UTF-32 character string in *utf32str*.

Specify the size (32-bit word count) of the buffer storing the UTF-32 character string in *utf32max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (32-bit word count) of the UTF-32 character string that has been determined to be representable in UTF-16 will return to *utf32Len*. The NULL termination character will not be included.

SCE CONFIDENTIAL

The length (16-bit word count) of the character string successfully represented in UTF-16 will return to *utf16Len*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *utf32max*, and that all processing has been completed normally.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed to *utf32str*. 0 is stored in both *utf32Len* and *utf16Len*.

If SCE_CES_ERROR_INVALID_ENCODE returns, it means that encoding has been determined to be invalid because a value equal or greater than 0x00110000, which is outside the UTF-32 valid range, has been found in the character string specified in *utf32str*.

If SCE_CES_ERROR_ILLEGAL_CODE returns, it means that the code has been determined to be illegal because code in the U+D800 to U+DFFF range reserved as a surrogate area was found in the character string specified in *utf32str*.

The value that returns to *utf32Len* and *utf16Len* in case of an error is the value up to the character processed successfully.

Use *sceCesUtf32StrToUtf16Str()* if you also wish to obtain the character string itself.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

SceCesUcsContext, *sceCesUcsContextInit()*, *sceCesUtf32StrToUtf16Str()*

sceCesUtf32StrGetUtf8Len

Retrieve character string length of UTF-32 character strings represented in UTF-8

Definition

```
#include <ces.h>
int sceCesUtf32StrGetUtf8Len (
    SceCesUcsContext* context,
    const uint32_t *utf32str,
    uint32_t utf32max,
    uint32_t *utf32Len,
    uint32_t *utf8Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>utf32str</i>	Address of the buffer storing UTF-32 character string
<i>utf32max</i>	Size (32-bit word count) of the buffer storing UTF-32 character string
<i>utf32Len</i>	Address of the variable for receiving UTF-32 character string length (32-bit word count)
<i>utf8Len</i>	Address of the variable for receiving UTF-8 character string length (byte count)

Return Values

Returns `SCE_OK (0)` as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
<code>SCE_CES_ERROR_INVALID_SRC_BUFFER</code>	<code>0x805C0010</code>	Specified source buffer is invalid
<code>SCE_CES_ERROR_INVALID_ENCODE</code>	<code>0x805C0014</code>	Source encoding determined to be invalid
<code>SCE_CES_ERROR_ILLEGAL_CODE</code>	<code>0x805C0015</code>	Illegal code detected in source character code

Description

This function retrieves the length of UTF-32 character strings when changed to UTF-8 character strings.

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

Specify the address of the buffer storing the UTF-32 character string in *utf32str*.

Specify the size (32-bit word count) of the buffer storing the UTF-32 character string in *utf32max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (32-bit word count) of the UTF-32 character string that has been determined to be representable in UTF-8 will return to *utf32Len*. The NULL termination character will not be included. The NULL termination character will not be included.

The length (byte count) of the character string successfully represented in UTF-8 will return to *utf8Len*. The NULL termination character will not be included.

If `SCE_OK` returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *utf32max*, and that all processing has been completed normally.

SCE CONFIDENTIAL

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed to `utf32str`. 0 is stored in both `utf32Len` and `utf8Len`.

If `SCE_CES_ERROR_INVALID_ENCODE` returns, it means that encoding has been determined to be invalid because a value equal or greater than 0x00110000, which is outside the UTF-32 valid range, has been found in the character string specified in `utf32str`.

If `SCE_CES_ERROR_ILLEGAL_CODE` returns, it means that the code has been determined to be illegal because code in the U+D800 to U+DFFF range reserved as a surrogate area was found in the character string specified in `utf32str`.

The value that returns to `utf32Len` and `utf8Len` in case of an error is the value up to the character processed successfully.

Use `sceCesUtf32StrToUtf8Str()` if you also wish to obtain the character string itself.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`SceCesUcsContext`, `sceCesUcsContextInit()`, `sceCesUtf32StrToUtf8Str()`

sceCesUtf32StrGetUcs2Len

Retrieve character string length of UTF-32 character strings represented in UCS-2

Definition

```
#include <ces.h>
int sceCesUtf32StrGetUcs2Len (
    SceCesUcsContext* context,
    const uint32_t *utf32str,
    uint32_t utf32max,
    uint32_t *utf32Len,
    uint32_t *ucs2Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>utf32str</i>	Address of the buffer storing UTF-32 character string
<i>utf32max</i>	Size (32-bit word count) of the buffer storing UTF-32 character string
<i>utf32Len</i>	Address of the variable for receiving UTF-32 character string length (32-bit word count)
<i>ucs2Len</i>	Address of the variable for receiving UCS-2 character code count

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_OUT_OF_CODE_RANGE	0x805C0024	Character outside the representable code range of the output destination encoding scheme is detected

Description

This function retrieves the length of UTF-32 character strings when changed to UCS-2 character strings.

(This function treats UCS-2 code arrays as character strings terminating with U+0000.)

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

Specify the address of the buffer storing the UTF-32 character string in *utf32str*.

Specify the size (32-bit word count) of the buffer storing the UTF-32 character string in *utf32max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (32-bit word count) of the UTF-32 character string that has been determined to be representable in UCS-2 will return to *utf32Len*. The NULL termination character will not be included. The NULL termination character will not be included.

SCE CONFIDENTIAL

The length (16-bit word count) of the character string successfully represented in UCS-2 will return to *ucs2Len*. The NULL termination character will not be included.

If *SCE_OK* returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *utf32max*, and that all processing has been completed normally.

If *SCE_CES_ERROR_INVALID_SRC_BUFFER* returns, it means that a NULL pointer has been passed to *utf32str*. 0 is stored in both *utf32Len* and *ucs2Len*.

If *SCE_CES_ERROR_INVALID_ENCODE* returns, it means that encoding has been determined to be invalid because a value equal or greater than 0x00110000, which is outside the UTF-32 valid range, has been found in the character string specified in *utf32str*.

If *SCE_CES_ERROR_ILLEGAL_CODE* returns, it means that the code has been determined to be illegal because code in the U+D800 to U+DFFF range reserved as a surrogate area was found in the character string specified in *ucs2str*.

If *SCE_CES_ERROR_OUT_OF_CODE_RANGE* returns, it means that there are no problems with the source character string itself, but that it includes character codes equal or greater than U+00010000, which cannot be represented in UCS-2.

The value that returns to *utf32Len* and *ucs2Len* in case of an error is the value up to the character processed successfully.

Use *sceCesUtf32StrToUcs2Str()* if you also wish to obtain the character string itself.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

SceCesUcsContext, *sceCesUcsContextInit()*, *sceCesUtf32StrToUcs2Str()*

sceCesUtf16StrGetUtf32Len

Retrieve character string length of UTF-16 character strings represented in UTF-32

Definition

```
#include <ces.h>
int sceCesUtf16StrGetUtf32Len (
    SceCesUcsContext* context,
    const uint16_t *utf16str,
    uint32_t utf16max,
    uint32_t *utf16Len,
    uint32_t *utf32Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>utf16str</i>	Address of the buffer storing UTF-16 character string
<i>utf16max</i>	Size (16-bit word count) of the buffer storing UTF-16 character string
<i>utf16Len</i>	Address of the variable for receiving UTF-16 character string length (16-bit word count)
<i>utf32Len</i>	Address of the variable for receiving UTF-32 character string length (32-bit word count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code

Description

This function retrieves the length of UTF-16 character strings when changed to UTF-32 character strings.

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

Specify the address of the buffer storing the UTF-16 character string in *utf16str*.

Specify the size (16-bit word count) of the buffer storing the UTF-16 character string in *utf16max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (16-bit word count) of the UTF-16 character string that has been determined to be representable in UTF-32 will return to *utf16Len*. The NULL termination character will not be included.

The length (32-bit word count) of the character string successfully represented in UTF-32 will return to *utf32Len*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *utf16max*, and that all processing has been completed normally.

SCE CONFIDENTIAL

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed to `utf16str`. 0 is stored in both `utf16Len` and `utf32Len`.

If `SCE_CES_ERROR_SRC_BUFFER_END` returns, it means that the character string specified in `utf16str` has been interrupted in the midst of a code representing one character due to limitation by `utf16max`.

If `SCE_CES_ERROR_ILLEGAL_CODE` returns, it means the code has been determined to be illegal because codes in the range between U+D800 and U+DFFF that is reserved as surrogate area were not used in pairs in the character string specified in `utf16str`.

The value that returns to `utf16Len` and `utf32Len` in case of an error is the value up to the character processed successfully.

Use `sceCesUtf16StrToUtf32Str()` if you also wish to obtain the character string itself.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`SceCesUcsContext`, `sceCesUcsContextInit()`, `sceCesUtf16StrToUtf32Str()`

SCE CONFIDENTIAL

sceCesUtf16StrGetUtf8Len

Retrieve character string length of UTF-16 character strings represented in UTF-8

Definition

```
#include <ces.h>
int sceCesUtf16StrGetUtf8Len (
    SceCesUcsContext* context,
    const uint16_t *utf16str,
    uint32_t utf16max,
    uint32_t *utf16Len,
    uint32_t *utf8Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>utf16str</i>	Address of the buffer storing UTF-16 character string
<i>utf16max</i>	Size (16-bit word count) of the buffer storing UTF-16 character string
<i>utf16Len</i>	Address of the variable for receiving UTF-16 character string length (16-bit word count)
<i>utf8Len</i>	Address of the variable for receiving UTF-8 character string length (byte count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code

Description

This function retrieves the length of UTF-16 character strings when changed to UTF-8 character strings.

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

Specify the address of the buffer storing the UTF-16 character string in *utf16str*.

Specify the size (16-bit word count) of the buffer storing the UTF-16 character string in *utf16max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (16-bit word count) of the UTF-16 character string that has been determined to be representable in UTF-8 will return to *utf16Len*. The NULL termination character will not be included.

The length (byte count) of the character string successfully represented in UTF-8 will return to *utf8Len*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *utf16max*, and that all processing has been completed normally.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed to *utf16str*. 0 is stored in both *utf16Len* and *utf8Len*.

©SCEI

SCE CONFIDENTIAL

If `SCE_CES_ERROR_SRC_BUFFER_END` returns, it means that the character string specified in *utf16str* has been interrupted in the midst of a code representing one character due to limitation by *utf16max*.

If `SCE_CES_ERROR_ILLEGAL_CODE` returns, it means the code has been determined to be illegal because codes in the range between U+D800 and U+DFFF that is reserved as surrogate area were not used in pairs in the character string specified in *utf16str*.

The value that returns to *utf16Len* and *utf8Len* in case of an error is the value up to the character processed successfully.

Use `sceCesUtf16StrToUtf8Str()` if you also wish to obtain the character string itself.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`SceCesUcsContext`, `sceCesUcsContextInit()`, `sceCesUtf16StrToUtf8Str()`

sceCesUtf16StrGetUcs2Len

Retrieve character string length of UTF-16 character strings represented in UCS-2

Definition

```
#include <ces.h>
int sceCesUtf16StrGetUcs2Len (
    SceCesUcsContext* context,
    const uint16_t *utf16str,
    uint32_t utf16max,
    uint32_t *utf16Len,
    uint32_t *ucs2Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>utf16str</i>	Address of the buffer storing UTF-16 character string
<i>utf16max</i>	Size (16-bit word count) of the buffer storing UTF-16 character string
<i>utf16Len</i>	Address of the variable for receiving UTF-16 character string length (16-bit word count)
<i>ucs2Len</i>	Address of the variable for receiving UCS-2 character code count

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_OUT_OF_CODE_RANGE	0x805C0024	Character outside the representable code range of the output destination encoding scheme is detected

Description

This function retrieves the length of UTF-16 character strings when changed to UCS-2 character strings.

(This function treats UCS-2 code arrays as character strings terminating with U+0000.)

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

Specify the address of the buffer storing the UTF-16 character string in *utf16str*.

Specify the size (16-bit word count) of the buffer storing the UTF-16 character string in *utf16max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (16-bit word count) of the UTF-16 character string that has been determined to be representable in UCS-2 will return to *utf16Len*. The NULL termination character will not be included.

SCE CONFIDENTIAL

The length (16-bit word count) of the character string successfully represented in UCS-2 will return to *ucs2Len*. The NULL termination character will not be included.

If *SCE_OK* returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *utf16max*, and that all processing has been completed normally.

If *SCE_CES_ERROR_INVALID_SRC_BUFFER* returns, it means that a NULL pointer has been passed to *utf16str*. 0 is stored in both *utf16Len* and *ucs2Len*.

If *SCE_CES_ERROR_SRC_BUFFER_END* returns, it means that the character string specified in *utf16str* has been interrupted in the midst of a code representing one character due to limitation by *utf16max*.

If *SCE_CES_ERROR_ILLEGAL_CODE* returns, it means the code has been determined to be illegal because codes in the range between U+D800 and U+DFFF that is reserved as surrogate area were not used in pairs in the character string specified in *utf16str*.

If *SCE_CES_ERROR_OUT_OF_CODE_RANGE* returns, it means that a character code equal or greater than U+00010000, which is outside the range that can be represented with UCS-2, was contained in the character string specified in *utf16str*.

The value that returns to *utf16Len* and *ucs2Len* in case of an error is the value up to the character processed successfully.

Use *sceCesUtf16StrToUcs2Str()* if you also wish to obtain the character string itself.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

SceCesUcsContext, *sceCesUcsContextInit()*, *sceCesUtf16StrToUcs2Str()*

sceCesUtf8StrGetUtf32Len

Retrieve character string length of UTF-8 character strings represented in UTF-32

Definition

```
#include <ces.h>
int sceCesUtf8StrGetUtf32Len (
    SceCesUcsContext* context,
    const uint8_t *utf8str,
    uint32_t utf8max,
    uint32_t *utf8Len,
    uint32_t *utf32Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>utf8str</i>	Address of the buffer storing UTF-8 character string
<i>utf8max</i>	Size (byte count) of the buffer storing UTF-8 character string
<i>utf8Len</i>	Address of the variable for receiving UTF-8 character string length (byte count)
<i>utf32Len</i>	Address of the variable for receiving UTF-32 character string length (32-bit word count)

Return Values

Returns `SCE_OK (0)` as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
<code>SCE_CES_ERROR_INVALID_SRC_BUFFER</code>	<code>0x805C0010</code>	Specified source buffer is invalid
<code>SCE_CES_ERROR_SRC_BUFFER_END</code>	<code>0x805C0011</code>	Specified source buffer is insufficient
<code>SCE_CES_ERROR_INVALID_ENCODE</code>	<code>0x805C0014</code>	Source encoding determined to be invalid
<code>SCE_CES_ERROR_ILLEGAL_CODE</code>	<code>0x805C0015</code>	Illegal code detected in source character code
<code>SCE_CES_ERROR_OUT_OF_CODE_RANGE</code>	<code>0x805C0024</code>	Character outside the representable code range of the output destination encoding scheme is detected

Description

This function retrieves the length of UTF-8 character strings when changed to UTF-32 character strings.

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

Specify the address of the buffer storing the UTF-8 character string in *utf8str*.

Specify the size (byte count) of the buffer storing the UTF-8 character string in *utf8max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (byte count) of the UTF-8 character string that has been determined to be representable in UTF-32 will return to *ucs8Len*. The NULL termination character will not be included.

SCE CONFIDENTIAL

The length (32-bit word count) of the character string successfully represented in UTF-32 will return to *utf32Len*. The NULL termination character will not be included.

If *SCE_OK* returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *utf8max*, and that all processing has been completed normally.

If *SCE_CES_ERROR_INVALID_SRC_BUFFER* returns, it means that a NULL pointer has been passed to *utf8str*. 0 is stored in both *utf8Len* and *utf32Len*.

If *SCE_CES_ERROR_SRC_BUFFER_END* returns, it means that the character string specified in *utf8str* has been interrupted in the midst of a code representing one character due to limitation by *utf8max*.

If *SCE_CES_ERROR_INVALID_ENCODE* returns, it means encoding has been determined to be invalid because the character string specified in *utf8str* contains byte strings that cannot be recognized as UTF-8.

If *SCE_CES_ERROR_ILLEGAL_CODE* returns, it means that the code has been found invalid because the character string specified in *utf8str* contains codes in the U+D800 to U+DFFF range reserved as a surrogate area, or encoding with an unnecessarily large number of bytes (such as representing U+0000 as the 2-byte sequence C0,80).

If *SCE_CES_ERROR_OUT_OF_CODE_RANGE* returns, it means that the character string specified in *utf8str* contained character code equal or greater than U+00110000 that cannot be handled by UTF-32.

The value that returns to *utf8Len* and *utf32Len* in case of an error is the value up to the character processed successfully.

Use *sceCesUtf8StrToUtf32Str()* if you also wish to obtain the character string itself.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

SceCesUcsContext, *sceCesUcsContextInit()*, *sceCesUtf8StrToUtf32Str()*

SCE CONFIDENTIAL

sceCesUtf8StrGetUtf16Len

Retrieve character string length of UTF-8 character strings represented in UTF-16

Definition

```
#include <ces.h>
int sceCesUtf8StrGetUtf16Len (
    SceCesUcsContext* context,
    const uint8_t *utf8str,
    uint32_t utf8max,
    uint32_t *utf8Len,
    uint32_t *utf16Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>utf8str</i>	Address of the buffer storing UTF-8 character string
<i>utf8max</i>	Size (byte count) of the buffer storing UTF-8 character string
<i>utf8Len</i>	Address of the variable for receiving UTF-8 character string length (byte count)
<i>utf16Len</i>	Address of the variable for receiving UTF-16 character string length (16-bit word count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_OUT_OF_CODE_RANGE	0x805C0024	Character outside the representable code range of the output destination encoding scheme is detected

Description

This function retrieves the length of UTF-8 character strings when changed to UTF-16 character strings.

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

Specify the address of the buffer storing the UTF-8 character string in *utf8str*.

Specify the size (byte count) of the buffer storing the UTF-8 character string in *utf8max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (byte count) of the UTF-8 character string that has been determined to be representable in UTF-16 will return to *ucs8Len*. The NULL termination character will not be included.

SCE CONFIDENTIAL

The length (16-bit word count) of the character string successfully represented in UTF-16 will return to *utf16Len*. The NULL termination character will not be included.

If `SCE_OK` returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *utf8max*, and that all processing has been completed normally.

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed to *utf8str*. 0 is stored in both *utf8Len* and *utf16Len*.

If `SCE_CES_ERROR_SRC_BUFFER_END` returns, it means that the character string specified in *utf8str* has been interrupted in the midst of a code representing one character due to limitation by *utf8max*.

If `SCE_CES_ERROR_INVALID_ENCODE` returns, it means encoding has been determined to be invalid because the character string specified in *utf8str* contains byte strings that cannot be recognized as UTF-8.

If `SCE_CES_ERROR_ILLEGAL_CODE` returns, it means that the code has been found invalid because the character string specified in *utf8str* contains codes in the U+D800 to U+DFFF range reserved as a surrogate area, or encoding with an unnecessarily large number of bytes (such as representing U+0000 as the 2-byte sequence C0,80).

If `SCE_CES_ERROR_OUT_OF_CODE_RANGE` returns, it means that a character code equal or greater than U+00110000, which is outside the range that can be represented with UTF-16, was contained in the character string specified in *utf8str*.

The value that returns to *utf8Len* and *utf16Len* in case of an error is the value up to the character processed successfully.

Use `sceCesUtf8StrToUtf16Str()` if you also wish to obtain the character string itself.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`SceCesUcsContext`, `sceCesUcsContextInit()`, `sceCesUtf8StrToUtf16Str()`

sceCesUtf8StrGetUcs2Len

Retrieve character string length of UTF-8 character strings represented in UCS-2

Definition

```
#include <ces.h>
int sceCesUtf8StrGetUcs2Len (
    SceCesUcsContext* context,
    const uint8_t *utf8str,
    uint32_t utf8max,
    uint32_t *utf8Len,
    uint32_t *ucs2Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>utf8str</i>	Address of the buffer storing UTF-8 character string
<i>utf8max</i>	Size (byte count) of the buffer storing UTF-8 character string
<i>utf8Len</i>	Address of the variable for receiving UTF-8 character string length (byte count)
<i>ucs2Len</i>	Address of the variable for receiving UCS-2 character code count

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_OUT_OF_CODE_RANGE	0x805C0024	Character outside the representable code range of the output destination encoding scheme is detected

Description

This function retrieves the length of UTF-8 character strings when changed to UCS-2 character strings. (This function treats UCS-2 code arrays as character strings terminating with U+0000.)

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

Specify the address of the buffer storing the UTF-8 character string in *utf8str*.

Specify the size (byte count) of the buffer storing the UTF-8 character string in *utf8max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (byte count) of the UTF-8 character string that has been determined to be representable in UCS-2 will return to *ucs2Len*. The NULL termination character will not be included.

SCE CONFIDENTIAL

The length (16-bit word count) of the character string successfully represented in UCS-2 will return to *ucs2Len*. The NULL termination character will not be included.

If *SCE_OK* returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *utf8max*, and that all processing has been completed normally.

If *SCE_CES_ERROR_INVALID_SRC_BUFFER* returns, it means that a NULL pointer has been passed to *utf8str*. 0 is stored in both *utf8Len* and *ucs2Len*.

If *SCE_CES_ERROR_SRC_BUFFER_END* returns, it means that the character string specified in *utf8str* has been interrupted in the midst of a code representing one character due to limitation by *utf8max*.

If *SCE_CES_ERROR_INVALID_ENCODE* returns, it means encoding has been determined to be invalid because the character string specified in *utf8str* contains byte strings that cannot be recognized as UTF-8.

If *SCE_CES_ERROR_ILLEGAL_CODE* returns, it means that the code has been found invalid because the character string specified in *utf8str* contains codes in the U+D800 to U+DFFF range reserved as a surrogate area, or encoding with an unnecessarily large number of bytes (such as representing U+0000 as the 2-byte sequence C0,80).

If *SCE_CES_ERROR_OUT_OF_CODE_RANGE* returns, it means that a character code equal or greater than U+00010000, which is outside the range that can be represented with UCS-2, was contained in the character string specified in *utf8str*.

The value that returns to *utf8Len* and *ucs2Len* in case of an error is the value up to the character processed successfully.

Use *sceCesUtf8StrToUcs2Str()* if you also wish to obtain the character string itself.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

SceCesUcsContext, *sceCesUcsContextInit()*, *sceCesUtf8StrToUcs2Str()*

sceCesUcs2StrGetUtf32Len

Retrieve character string length of UCS-2 character strings represented in UTF-32

Definition

```
#include <ces.h>
int sceCesUcs2StrGetUtf32Len (
    SceCesUcsContext* context,
    const uint16_t *ucs2str,
    uint32_t ucs2max,
    uint32_t *ucs2Len,
    uint32_t *utf32Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>ucs2str</i>	Address of the buffer storing UCS-2 character string
<i>ucs2max</i>	Size (16-bit word count) of the buffer storing UCS-2 character string
<i>ucs2Len</i>	Address of the variable for receiving UCS-2 character string length (16-bit word count)
<i>utf32Len</i>	Address of the variable for receiving UTF-32 character string length (32-bit word count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code

Description

This function retrieves the length of UCS-2 character strings when changed to UTF-32 character strings.

(This function treats UCS-2 code arrays as character strings terminating with U+0000.)

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

Specify the address of the buffer storing the UCS-2 character string in *ucs2str*.

Specify the size (16-bit word count) of the buffer storing the UCS-2 character string in *ucs2max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified. In this case, *ucs2str*'s termination must always be U+0000.

The length (16-bit word count) of the UCS-2 character string that has been determined to be representable in UTF-32 will return to *ucs2Len*. The NULL termination character will not be included.

The length (32-bit word count) of the character string successfully represented in UTF-32 will return to *utf32Len*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *ucs2max*, and that all processing has been completed normally.

SCE CONFIDENTIAL

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed to `ucs2str`. 0 is stored in both `ucs2Len` and `utf32Len`.

If `SCE_CES_ERROR_ILLEGAL_CODE` returns, it means that the code has been determined to be illegal because code in the U+D800 to U+DFFF range reserved as a surrogate area was found in the character string specified in `ucs2str`.

The value that returns to `ucs2Len` and `utf32Len` in case of an error is the value up to the character processed successfully.

Use `sceCesUcs2StrToUtf32Str()` if you also wish to obtain the character string itself.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`SceCesUcsContext`, `sceCesUcsContextInit()`, `sceCesUcs2StrToUtf32Str()`

sceCesUcs2StrGetUtf16Len

Retrieve character string length of UCS-2 character strings represented in UTF-16

Definition

```
#include <ces.h>
int sceCesUcs2StrGetUtf16Len (
    SceCesUcsContext* context,
    const uint16_t *ucs2str,
    uint32_t ucs2max,
    uint32_t *ucs2Len,
    uint32_t *utf16Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>ucs2str</i>	Address of the buffer storing UCS-2 character string
<i>ucs2max</i>	Size (16-bit word count) of the buffer storing UCS-2 character string
<i>ucs2Len</i>	Address of the variable for receiving UCS-2 character string length (16-bit word count)
<i>utf16Len</i>	Address of the variable for receiving UTF-16 character string length (16-bit word count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code

Description

This function retrieves the length of UCS-2 character strings when changed to UTF-16 character strings.

(This function treats UCS-2 code arrays as character strings terminating with U+0000.)

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

Specify the address of the buffer storing the UCS-2 character string in *ucs2str*.

Specify the size (16-bit word count) of the buffer storing the UCS-2 character string in *ucs2max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified. In this case, *ucs2str*'s termination must always be U+0000.

The length (16-bit word count) of the UCS-2 character string that has been determined to be representable in UTF-16 will return to *ucs2Len*. The NULL termination character will not be included.

The length (16-bit word count) of the character string successfully represented in UTF-16 will return to *utf16Len*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *ucs2max*, and that all processing has been completed normally.

SCE CONFIDENTIAL

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed to `ucs2str`. 0 is stored in both `ucs2Len` and `utf16Len`.

If `SCE_CES_ERROR_ILLEGAL_CODE` returns, it means that the code has been determined to be illegal because code in the U+D800 to U+DFFF range reserved as a surrogate area was found in the character string specified in `ucs2str`.

The value that returns to `ucs2Len` and `utf16Len` in case of an error is the value up to the character processed successfully.

Use `sceCesUcs2StrToUtf16Str()` if you also wish to obtain the character string itself.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`SceCesUcsContext`, `sceCesUcsContextInit()`, `sceCesUcs2StrToUtf16Str()`

sceCesUcs2StrGetUtf8Len

Retrieve character string length of UCS-2 character strings represented in UTF-8

Definition

```
#include <ces.h>
int sceCesUcs2StrGetUtf8Len (
    SceCesUcsContext* context,
    const uint16_t *ucs2str,
    uint32_t ucs2max,
    uint32_t *ucs2Len,
    uint32_t *utf8Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>ucs2str</i>	Address of the buffer storing UCS-2 character string
<i>ucs2max</i>	Size (16-bit word count) of the buffer storing UCS-2 character string
<i>ucs2Len</i>	Address of the variable for receiving UCS-2 character string length (16-bit word count)
<i>utf8Len</i>	Address of the variable for receiving UTF-8 character code count

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code

Description

This function retrieves the length of UCS-2 character strings when changed to UTF-8 character strings. (This function treats UCS-2 code arrays as character strings terminating with U+0000.)

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

Specify the address of the buffer storing the UCS-2 character string in *ucs2str*.

Specify the size (16-bit word count) of the buffer storing the UCS-2 character string in *ucs2max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified. In this case, *ucs2str*'s termination must always be U+0000.

The length (16-bit word count) of the UCS-2 character string that has been determined to be representable in UTF-8 will return to *ucs2Len*. The NULL termination character will not be included.

The length (byte count) of the character string successfully represented in UTF-8 will return to *utf8Len*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *ucs2max*, and that all processing has been completed normally.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed to *ucs2str*. 0 will be stored in both *ucs2Len* and *utf8Len*.

SCE CONFIDENTIAL

If `SCE_CES_ERROR_ILLEGAL_CODE` returns, it means that the code has been determined to be illegal because code in the U+D800 to U+DFFF range reserved as a surrogate area was found in the character string specified in *ucs2str*.

The value that returns to *ucs2Len* and *utf8Len* in case of an error is the value up to the character processed successfully.

Use `sceCesUcs2StrToUtf8Str()` if you also wish to obtain the character string itself.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`SceCesUcsContext`, `sceCesUcsContextInit()`, `sceCesUcs2StrToUtf8Str()`

sceCesUtf32StrGetCopyLen

Retrieve the length of the character string obtained by copying it as a UTF-32 character string

Definition

```
#include <ces.h>
int sceCesUtf32StrGetCopyLen (
    SceCesUcsContext* context,
    const uint32_t *utf32str,
    uint32_t utf32max,
    uint32_t *utf32Len,
    uint32_t *copyLen
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>utf32str</i>	Address of the buffer storing UTF-32 character string
<i>utf32max</i>	Size (32-bit word count) of the buffer storing UTF-32 character string
<i>utf32Len</i>	Address of the variable for receiving the length (32-bit word count) of the UTF-32 character string
<i>copyLen</i>	Address of the variable for receiving the length (32-bit word count) of the character string that can be copied as UTF-32

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code

Description

This function retrieves the length of the character string obtained by copying a UTF-32 character string with `sceCesUtf32StrToCopyStr()`.

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

Specify the address of the buffer storing the UTF-32 character string in *utf32str*.

Specify the size (32-bit word count) of the buffer storing the UTF-32 character string in *utf32max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (32-bit word count) of the UTF-32 character string recognized when copied will return to *utf32Len*. The NULL termination character will not be included.

The length (32-bit word count) of the character string obtained by copying will return to *copyLen*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *utf32max*, and that all processing has been completed normally.

SCE CONFIDENTIAL

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed to `utf32str`. 0 is stored in both `utf32Len` and `utf8Len`.

If `SCE_CES_ERROR_INVALID_ENCODE` returns, it means that encoding has been determined to be invalid because a value equal or greater than 0x00110000, which is outside the UTF-32 valid range, has been found in the character string specified in `utf32str`.

If `SCE_CES_ERROR_ILLEGAL_CODE` returns, it means that the code has been determined to be illegal because code in the U+D800 to U+DFFF range reserved as a surrogate area was found in the character string specified in `utf32str`.

The value that returns to `utf32Len` and `copyLen` in case of an error is the value up to the character processed successfully.

Use `sceCesUtf32StrToCopyStr()` if you also wish to obtain the character string itself.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`SceCesUcsContext`, `sceCesUcsContextInit()`, `sceCesUtf32StrToUtf8Str()`

sceCesUtf16StrGetCopyLen

Retrieve the length of a character string that can be extracted as a UTF-16 character string

Definition

```
#include <ces.h>
int sceCesUtf16StrGetCopyLen (
    SceCesUcsContext* context,
    const uint16_t *utf16str,
    uint32_t utf16max,
    uint32_t *utf16Len,
    uint32_t *copyLen
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>utf16str</i>	Address of the buffer storing UTF-16 character string
<i>utf16max</i>	Size (16-bit word count) of the buffer storing UTF-16 character string
<i>utf16Len</i>	Address of the variable for receiving UTF-16 character string length (16-bit word count)
<i>copyLen</i>	Address of the variable for receiving the length (16-bit word count) of the character string that can be copied as UTF-16

Return Values

Returns `SCE_OK (0)` as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
<code>SCE_CES_ERROR_INVALID_SRC_BUFFER</code>	<code>0x805C0010</code>	Specified source buffer is invalid
<code>SCE_CES_ERROR_SRC_BUFFER_END</code>	<code>0x805C0011</code>	Specified source buffer is insufficient
<code>SCE_CES_ERROR_ILLEGAL_CODE</code>	<code>0x805C0015</code>	Illegal code detected in source character code

Description

This function retrieves the length of the character string obtained by copying a UTF-32 character string with `sceCesUtf16StrToCopyStr()`.

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

Specify the address of the buffer storing the UTF-16 character string in *utf16str*.

Specify the size (16-bit word count) of the buffer storing the UTF-16 character string in *utf16max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (16-bit word count) of the UTF-16 character string recognized when copied will return to *utf16Len*. The NULL termination character will not be included.

The length (16-bit word count) of the character string obtained by copying will return to *copyLen*. The NULL termination character will not be included.

If `SCE_OK` returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *utf16max*, and that all processing has been completed normally.

SCE CONFIDENTIAL

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed to *utf16str*. 0 is stored in both *utf16Len* and *utf8Len*.

If `SCE_CES_ERROR_SRC_BUFFER_END` returns, it means that the character string specified in *utf16str* has been interrupted in the midst of a code representing one character due to limitation by *utf16max*.

If `SCE_CES_ERROR_ILLEGAL_CODE` is returned, it means the code has been determined to be illegal because codes in the range between U+D800 and U+DFFF that is reserved as surrogate area were not used in pairs.

The value that returns to *utf16Len* and *copyLen* in case of an error is the value up to the character processed successfully.

Use `sceCesUtf16StrToCopyStr()` if you also wish to obtain the character string itself.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`SceCesUcsContext`, `sceCesUcsContextInit()`, `sceCesUtf16StrToUtf8Str()`

sceCesUtf8StrGetCopyLen

Retrieve character string length of UTF-8 character strings represented in UTF-16

Definition

```
#include <ces.h>
int sceCesUtf8StrGetCopyLen (
    SceCesUcsContext* context,
    const uint8_t *utf8str,
    uint32_t utf8max,
    uint32_t *utf8Len,
    uint32_t *copyLen
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>utf8str</i>	Address of the buffer storing UTF-8 character string
<i>utf8max</i>	Size (byte count) of the buffer storing UTF-8 character string
<i>utf8Len</i>	Address of the variable for receiving UTF-8 character string length (byte count)
<i>copyLen</i>	Address of the variable for receiving the length (byte count) of the character string that can be copied as UTF-8

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code

Description

This function retrieves the length of the character string obtained by copying a UTF-8 character string with `sceCesUtf8StrToCopyStr()`.

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

Specify the address of the buffer storing the UTF-8 character string in *utf8str*.

Specify the size (byte count) of the buffer storing the UTF-8 character string in *utf8max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (byte count) of the UTF-8 character string recognized when copied will return to *utf8Len*. The NULL termination character will not be included.

The length (byte count) of the character string obtained by copying will return to *copyLen*. The NULL termination character will not be included.

SCE CONFIDENTIAL

If `SCE_OK` returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *utf8max*, and that all processing has been completed normally.

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed to *utf8str*. 0 is stored in both *utf8Len* and *utf16Len*.

If `SCE_CES_ERROR_SRC_BUFFER_END` returns, it means that the character string specified in *utf8str* has been interrupted in the midst of a code representing one character due to limitation by *utf8max*.

If `SCE_CES_ERROR_INVALID_ENCODE` returns, it means encoding has been determined to be invalid because the character string specified in *utf8str* contains byte strings that cannot be recognized as UTF-8.

If `SCE_CES_ERROR_ILLEGAL_CODE` returns, it means that the code has been found invalid because the character string specified in *utf8str* contains codes in the U+D800 to U+DFFF range reserved as a surrogate area, or encoding with an unnecessarily large number of bytes (such as representing U+0000 as the 2-byte sequence C0,80).

The value that returns to *utf8Len* and *copyLen* in case of an error is the value up to the character processed successfully.

Use `sceCesUtf8StrToCopyStr()` if you also wish to obtain the character string itself.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`SceCesUcsContext`, `sceCesUcsContextInit()`, `sceCesUtf8StrToUtf16Str()`

sceCesUcs2StrGetCopyLen

Retrieve the length of the copied UCS-2 character strings

Definition

```
#include <ces.h>
int sceCesUcs2StrGetCopyLen (
    SceCesUcsContext* context,
    const uint16_t *ucs2str,
    uint32_t ucs2max,
    uint32_t *ucs2Len,
    uint32_t *copyLen
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>ucs2str</i>	Address of the buffer storing UCS-2 character string
<i>ucs2max</i>	Size (16-bit word count) of the buffer storing UCS-2 character string
<i>ucs2Len</i>	Address of the variable for receiving UCS-2 character string length (16-bit word count)
<i>copyLen</i>	Address of the variable for receiving the length (16-bit word count) of the character string that can be copied as UCS-2

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code

Description

This function retrieves the length of the character string obtained by copying a UCS-2 character string with `sceCesUcs2StrToCopyStr()`.

(This function treats UCS-2 code arrays as character strings terminating with U+0000.)

Specify the context initialized with `sceCesUcsContextInit()` in *context*.

Specify the address of the buffer storing the UCS-2 character string in *ucs2str*. (This function treats UCS-2 code arrays as character strings terminating with U+0000.)

Specify the size (16-bit word count) of the buffer storing the UCS-2 character string in *ucs2max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified. In this case, *ucs2str*'s termination must always be U+0000.

The length (16-bit code count) of the UCS-2 character string recognized when copied will return to *ucs2Len*. The NULL termination character will not be included.

The length (16-bit word count) of the character string obtained by copying will return to *copyLen*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *ucs2max*, and that all processing has been completed normally.

SCE CONFIDENTIAL

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed to `ucs2str`. 0 will be stored in both `ucs2Len` and `utf8Len`.

If `SCE_CES_ERROR_ILLEGAL_CODE` returns, it means that the code has been determined to be illegal because code in the U+D800 to U+DFFF range reserved as a surrogate area was found in the character string specified in `ucs2str`.

The value that returns to `ucs2Len` and `copyLen` in case of an error is the value up to the character processed successfully.

Use `sceCesUcs2StrToCopyStr()` if you also wish to obtain the character string itself.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`SceCesUcsContext`, `sceCesUcsContextInit()`, `sceCesUcs2StrToUtf8Str()`

Other Character Sets and UCS Character String Functions

Character String Conversion Context of MBCS (Including SBCS) and UCS

SceCesMbcUcsContext

Context for MBCS character string conversion

Definition

```
#include <ces.h>
typedef union SceCesMbcUcsContext{
    SceCesUcsContext cesUcsCtx,
    /* omitted */
} SceCesMbcUcsContext;
```

Members

cesUcsCtx Context for handling Unicode character strings

Description

This is a context type used for multi-byte character sets (including single-byte character sets) and UCS character string conversion.

Before use, it is necessary to perform initialization with `sceCesMbcUcsContextInit()`.

cesUcsCtx is a context used when handling Unicode character strings.

The arguments of `sceCesUcsStrGetEncodingSize()` and `sceCesUcsStrConvertEncoding()` receive the address of this member.

See Also

```
sceCesMbcUcsContextInit(), sceCesMbcStrGetUtf32Len(),
sceCesMbcStrToUtf32Str(), sceCesUtf32StrGetMbcLen(),
sceCesUtf32StrToMbcStr(), sceCesMbcStrGetUtf16Len(),
sceCesMbcStrToUtf16Str(), sceCesUtf16StrGetMbcLen(),
sceCesUtf16StrToMbcStr(), sceCesMbcStrGetUtf8Len(), sceCesMbcStrToUtf8Str(),
sceCesUtf8StrGetMbcLen(), sceCesUtf8StrToMbcStr(), sceCesMbcStrGetUcs2Len(),
sceCesMbcStrToUcs2Str(), sceCesUcs2StrGetMbcLen(), sceCesUcs2StrToMbcStr(),
sceCesUcsStrGetEncodingSize(), sceCesUcsStrConvertEncoding()
```

SCE CONFIDENTIAL

sceCesMbcUcsContextInit

Initialize the context for MBCS character string conversion

Definition

```
#include <ces.h>
int sceCesMbcUcsContextInit(
    SceCesMbcUcsContext *context,
    const SceCesMbcUcsProfile *profile
)
```

Arguments

<i>context</i>	Address of the context for MBCS character string conversion
<i>profile</i>	Address of the profile indicating MBCS/UCS conversion information

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns the following error code (negative value) in case of error.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid

Description

This function initializes the context used when converting multi-byte character sets (including single-byte character sets) and UCS character strings.

The initialized context is required as the argument of functions handling MBCS character strings.

When using the context initialized with this function, it is possible to handle MBCS character strings with integrated functions such as `sceCesUcsStrGetEncodingSize()` and `sceCesUcsStrConvertEncoding()`.

Specify the address of the `SceCesMbcUcsContext` data type allocated by the caller in *context*.

Specify the address of `SceCesMbcUcsProfile` obtained with the procedure described below in *profile*.

Firstly, if you wish to perform conversion of a multi-byte character set and UCS, obtain the address of each CES's UCS conversion profile with the return value of an initialization function whose name begins by `sceCesUcsProfileInit`.

If you wish to perform conversion of a single-byte character set and UCS, obtain the address of the UCS conversion profile of the single-byte character set with the return value of a function whose name begins by `sceCesRefersUcsProfile`.

The address of the UCS conversion profile of each CES retrieved can be obtained as a `const SceCesMbcUcsProfile*` type address by using the macro function `sceCesGetMbcUcsProfile()`. Specify this in *profile*.

If SCE_CES_ERROR_INVALID_PARAMETER returns, it means that a NULL pointer has been passed to *context* or *profile*.

SCE CONFIDENTIAL

Examples

```

// SBCS
{
    SceCesSbcsUcsProfile *sProf;
    SceCesMbcsUcsProfile *mProf;
    SceCesMbcsUcsContext mctx;

    sProf = sceCesRefersUcsProfileIso8859_15();
    mProf = sceCesGetMbcsUcsProfile( sProf );
    sceCesMbcsUcsContextInit( &mctx, mProf );
    :
}

// MBCS
static SceCesUcsProfileSheet s_sheet[1];
static SceCesSJisUcsProfile *s_uprof_sjis;

s_uprof_sjis = sceCesUcsProfileInitSJis( &s_sheet[0] );
{
    SceCesMbcsUcsContext mctx;
    SceCesMbcsUcsProfile *mProf;

    mProf = sceCesGetMbcsUcsProfile( s_uprof_sjis );
    sceCesMbcsUcsContextInit( &mctx, mProf );
    :
}

```

Notes

This function is not multi-thread safe.

Context entity must be divided by thread for multithreading.

See Also

```

sceCesMbcsStrGetUtf32Len(), sceCesMbcsStrToUtf32Str(),
sceCesUtf32StrGetMbcsLen(), sceCesUtf32StrToMbcsStr(),
sceCesMbcsStrGetUtf16Len(), sceCesMbcsStrToUtf16Str(),
sceCesUtf16StrGetMbcsLen(), sceCesUtf16StrToMbcsStr(),
sceCesMbcsStrGetUtf8Len(), sceCesMbcsStrToUtf8Str(), sceCesUtf8StrGetMbcsLen(),
sceCesUtf8StrToMbcsStr(), sceCesMbcsStrGetUcs2Len(), sceCesMbcsStrToUcs2Str(),
sceCesUcs2StrGetMbcsLen(), sceCesUcs2StrToMbcsStr(),
sceCesUcsStrGetEncodingSize(), sceCesUcsStrConvertEncoding()

```

SCE CONFIDENTIAL

sceCesMbcsUcsContextInitCopy

Copy Initialization of the context for MBCS character string conversion

Definition

```
#include <ces.h>
int sceCesMbcsUcsContextInitCopy (
    SceCesMbcsUcsContext *context,
    const SceCesMbcsUcsContext *context0
)
```

Arguments

context Address of the context for MBCS character string conversion to be initialized
context0 Address of the context for MBCS character string conversion of the copy source

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns the following error code (negative value) in case of error.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid

Description

This function initializes the context used when converting multi-byte character sets (including single-byte character sets) and UCS character strings by copying the setting status from other already initialized contexts.

Specify the address of the *SceCesMbcsUcsContext* data type allocated by the caller in *context*.

Specify the address of *SceCesMbcsUcsContext* that has already been initialized with *sceCesMbcsUcsContextInit()* to *context0*. The settings and status of the context specified in *context0* will be copied to the context specified in *context*.

The reference addresses of a profile, etc. specified when the context of the copy source is initialized with *sceCesMbcsUcsContextInit()* will be copied to and used by the newly initialized context. Note that the *SceCesUcsProfileSheet* data type entity, etc. used by the copy source will also be referenced by the copier.

If SCE_CES_ERROR_INVALID_PARAMETER returns, it means that a NULL pointer has been passed to *context* or *context0*.

Notes

This function is not multi-thread safe.

Context entity must be divided by thread for multithreading.

See Also

SceCesMbcsUcsContext(), *sceCesMbcsUcsContextInit()*

Character String Conversion Functions for Handling MBCS (Including SBCS) and UCS

sceCesMbcStrToUtf32Str

Convert from MBCS character string to UTF-32 character string

Definition

```
#include <ces.h>
int sceCesMbcStrToUtf32Str(
    SceCesMbcUcsContext* context,
    const uint8_t *mbcsStr,
    uint32_t mbcsMax,
    uint32_t *mbcsLen,
    uint32_t *utf32buf,
    uint32_t utf32max,
    uint32_t *utf32Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>mbcsStr</i>	Address of the buffer storing MBCS character string
<i>mbcsMax</i>	Size (byte count) of the buffer storing MBCS character string
<i>mbcsLen</i>	Address of the variable for receiving MBCS character string length (byte count)
<i>utf32buf</i>	Address of the buffer for receiving UTF-32 character string
<i>utf32max</i>	Size (32-bit word count) of the buffer for receiving UTF-32 character string
<i>utf32Len</i>	Address of the variable for receiving UTF-32 character string length (32-bit word count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

SCE CONFIDENTIAL

Description

This function converts MBCS character strings to UTF-32 character strings.

Specify the context initialized with `sceCesMbcUcsContextInit()` in *context*.

Specify the address of the buffer storing the MBCS character string in *mbcsStr*.

Specify the size (byte count) of the buffer storing the MBCS character string in *mbcsMax*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (byte count) of the MBCS character string used for conversion to UTF-32 will return to *mbcsLen*. The NULL termination character will not be included.

In *utf32buf*, specify the address of the buffer storing the character string specified in *mbcsStr* and represented in UTF-32. NULL termination is always guaranteed for the buffer passed to this argument.

Specify the size (32-bit word count) allowed for writing to *utf32buf* in *utf32max*. Specify a size taking into account the part for writing the NULL termination character.

The length (32-bit word count) of the character string successfully converted to UTF-32 will return to *utf32Len*. The NULL termination character will not be included.

If `SCE_OK` returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *mbcsMax*, and that all processing has been completed normally.

If `SCE_CES_ERROR_INVALID_PARAMETER` returns, it means that a NULL pointer has been specified to *context*.

If `SCE_CES_ERROR_INVALID_PROFILE` returns, it means that the profile set in *context* has been determined to be invalid.

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed to *mbcsStr*. 0 is stored in both *mbcsLen* and *utf32Len*.

If `SCE_CES_ERROR_INVALID_ENCODE` returns, it means that the encoding has been determined to be invalid because it includes byte values that cannot be recognized, such as in the case where the byte string passed to *mbcsStr* includes values within the MBCS's holding area.

The return of `SCE_CES_ERROR_UNASSIGNED_CODE` indicates that the character string specified in *mbcsStr* included characters without mapping to UCS.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been passed to *utf32buf*.

If `SCE_CES_ERROR_DST_BUFFER_END` returns, it means that the buffer specified as output destination has run out, and that the character string could not be output completely.

The value that returns to *mbcsLen* and *utf32Len* in case of an error is the value up to the character processed successfully.

Use `sceCesMbcStrGetUtf32Len()` if you only want to look up output character string length.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`sceCesMbcUcsContextInit()`, `sceCesMbcStrGetUtf32Len()`,
`sceCesUtf32StrToMbcStr()`

SCE CONFIDENTIAL

sceCesMbcStrToUtf16Str

Convert from MBCS character string to UTF-16 character string

Definition

```
#include <ces.h>
int sceCesMbcStrToUtf16Str(
    SceCesMbcUcsContext* context,
    const uint8_t *mbcsStr,
    uint32_t mbcMax,
    uint32_t *mbcsLen,
    uint16_t *utf16buf,
    uint32_t utf16max,
    uint32_t *utf16Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>mbcsStr</i>	Address of the buffer storing MBCS character string
<i>mbcsMax</i>	Size (byte count) of the buffer storing MBCS character string
<i>mbcsLen</i>	Address of the variable for receiving MBCS character string length (byte count)
<i>utf16buf</i>	Address of the buffer for receiving UTF-16 character string
<i>utf16max</i>	Size (16-bit word count) of the buffer for receiving UTF-16 character string
<i>utf16Len</i>	Address of the variable for receiving UTF-16 character string length (16-bit word count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

This function converts MBCS character strings to UTF-16 character strings.

Specify the context initialized with `sceCesMbcUcsContextInit()` in *context*.

Specify the address of the buffer storing the MBCS character string in *mbcsStr*.

Specify the size (byte count) of the buffer storing the MBCS character string in *mbcsMax*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (byte count) of the MBCS character string used for conversion to UTF-16 will return to *mbcsLen*. The NULL termination character will not be included.

In *utf16buf*, specify the address of the buffer storing the character string specified in *mbcsStr* and represented in UTF-16. NULL termination is always guaranteed for the buffer passed to this argument.

Specify the size (16-bit word count) allowed for writing to *utf16buf* in *utf16max*. Specify a size taking into account the part for writing the NULL termination character.

The length (16-bit word count) of the character string successfully converted to UTF-16 will return to *utf16Len*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *mbcsMax*, and that all processing has been completed normally.

If SCE_CES_ERROR_INVALID_PARAMETER returns, it means that a NULL pointer has been specified to *context*.

If SCE_CES_ERROR_INVALID_PROFILE returns, it means that the profile set in *context* has been determined to be invalid.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed to *mbcsStr*. 0 is stored in both *mbcsLen* and *utf16Len*.

If SCE_CES_ERROR_INVALID_ENCODE returns, it means that the encoding has been determined to be invalid because it includes byte values that cannot be recognized, such as in the case where the byte string passed to *mbcsStr* includes values within the MBCS's holding area.

The return of SCE_CES_ERROR_UNASSIGNED_CODE indicates that the character string specified in *mbcsStr* included characters without mapping to UCS.

If SCE_CES_ERROR_INVALID_DST_BUFFER returns, it means that a NULL pointer has been passed to *utf16buf*.

If SCE_CES_ERROR_DST_BUFFER_END returns, it means that the buffer specified as output destination has run out, and that the character string could not be output completely.

The value that returns to *mbcsLen* and *utf16Len* in case of an error is the value up to the character processed successfully.

Use `sceCesMbcStrGetUtf16Len()` if you only want to look up output character string length.

Notes

This function is not multi-thread safe.
Context must be divided for multithreading.

See Also

`sceCesMbcUcsContextInit()`, `sceCesMbcStrGetUtf16Len()`,
`sceCesUtf16StrToMbcStr()`

SCE CONFIDENTIAL

sceCesMbcStrToUtf8Str

Convert from MBCS character string to UTF-8 character string

Definition

```
#include <ces.h>
int sceCesMbcStrToUtf8Str(
    SceCesMbcUcsContext* context,
    const uint8_t *mbcsStr,
    uint32_t mbcMax,
    uint32_t *mbcsLen,
    uint8_t *utf8buf,
    uint32_t utf8max,
    uint32_t *utf8Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>mbcsStr</i>	Address of the buffer storing MBCS character string
<i>mbcsMax</i>	Size (byte count) of the buffer storing MBCS character string
<i>mbcsLen</i>	Address of the variable for receiving MBCS character string length (byte count)
<i>utf8buf</i>	Address of the buffer for receiving UTF-8 character string
<i>utf8max</i>	Size (byte count) of the buffer for receiving UTF-8 character string
<i>utf8Len</i>	Address of the variable for receiving UTF-8 character string length (byte count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

This function converts MBCS character strings to UTF-8 character strings.

Specify the context initialized with `sceCesMbcUcsContextInit()` in *context*.

Specify the address of the buffer storing the MBCS character string in *mbcsStr*.

Specify the size (byte count) of the buffer storing the MBCS character string in *mbcsMax*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (byte count) of the MBCS character string used for conversion to UTF-8 will return to *mbcsLen*. The NULL termination character will not be included.

In *utf8buf*, specify the address of the buffer storing the character string specified in *mbcsStr* and represented in UTF-8. NULL termination is always guaranteed for the buffer passed to this argument.

Specify the size (byte count) allowed for writing to *utf8buf* in *utf8max*. Specify a size taking into account the part for writing the NULL termination character.

The length (byte count) of the character string successfully converted to UTF-8 will return to *utf8Len*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *mbcsMax*, and that all processing has been completed normally.

If SCE_CES_ERROR_INVALID_PARAMETER returns, it means that a NULL pointer has been specified to *context*.

If SCE_CES_ERROR_INVALID_PROFILE returns, it means that the profile set in *context* has been determined to be invalid.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed to *mbcsStr*. 0 is stored in both *mbcsLen* and *utf8Len*.

If SCE_CES_ERROR_INVALID_ENCODE returns, it means that the encoding has been determined to be invalid because it includes byte values that cannot be recognized, such as in the case where the byte string passed to *mbcsStr* includes values within the MBCS's holding area.

The return of SCE_CES_ERROR_UNASSIGNED_CODE indicates that the character string specified in *mbcsStr* included characters without mapping to UCS.

If SCE_CES_ERROR_INVALID_DST_BUFFER returns, it means that a NULL pointer has been passed to *utf8buf*.

If SCE_CES_ERROR_DST_BUFFER_END returns, it means that the buffer specified as output destination has run out, and that the character string could not be output completely.

The value that returns to *mbcsLen* and *utf8Len* in case of an error is the value up to the character processed successfully.

Use `sceCesMbcStrGetUtf8Len()` if you only want to look up output character string length.

Notes

This function is not multi-thread safe.
Context must be divided for multithreading.

See Also

`sceCesMbcUcsContextInit()`, `sceCesMbcStrGetUtf8Len()`,
`sceCesUtf8StrToMbcStr()`

sceCesMbcStrToUcs2Str

Convert from MBCS character string to UCS-2 character string

Definition

```
#include <ces.h>
int sceCesMbcStrToUcs2Str(
    SceCesMbcUcsContext* context,
    const uint8_t *mbcsStr,
    uint32_t mbcMax,
    uint32_t *mbcsLen,
    uint16_t *ucs2buf,
    uint32_t ucs2max,
    uint32_t *ucs2Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>mbcsStr</i>	Address of the buffer storing MBCS character string
<i>mbcsMax</i>	Size (byte count) of the buffer storing MBCS character string
<i>mbcsLen</i>	Address of the variable for receiving MBCS character string length (byte count)
<i>ucs2buf</i>	Address of the buffer for receiving UCS-2 character string
<i>ucs2max</i>	Size (16-bit word count) of the buffer for receiving UCS-2 character string
<i>ucs2Len</i>	Address of the variable for receiving UCS-2 character string length (16-bit word count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

This function converts MBCS character strings to UCS-2 character strings.
(This function treats UCS-2 code arrays as character strings terminating with U+0000.)

Specify the context initialized with `sceCesMbcUcsContextInit()` in *context*.

Specify the address of the buffer storing the MBCS character string in *mbcsStr*.

Specify the size (byte count) of the buffer storing the MBCS character string in *mbcsMax*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (byte count) of the MBCS character string used for conversion to UCS-2 will return to *mbcsLen*. The NULL termination character will not be included.

In *ucs2buf*, specify the address of the buffer storing the character string specified in *mbcsStr* and represented in UCS-2. NULL termination is always guaranteed for the buffer passed to this argument.

Specify the size (16-bit word count) allowed for writing to *ucs2buf* in *ucs2max*. Specify a size taking into account the part for writing the NULL termination character.

The length (16-bit word count) of the character string successfully converted to UCS-2 will return to *ucs2Len*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *mbcsMax*, and that all processing has been completed normally.

If SCE_CES_ERROR_INVALID_PARAMETER returns, it means that a NULL pointer has been specified to *context*.

If SCE_CES_ERROR_INVALID_PROFILE returns, it means that the profile set in *context* has been determined to be invalid.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed to *mbcsStr*. 0 is stored in both *mbcsLen* and *ucs2Len*.

If SCE_CES_ERROR_INVALID_ENCODE returns, it means that the encoding has been determined to be invalid because it includes byte values that cannot be recognized, such as in the case where the byte string passed to *mbcsStr* includes values within the MBCS's holding area.

The return of SCE_CES_ERROR_UNASSIGNED_CODE indicates that the character string specified in *mbcsStr* included characters without mapping to UCS.

If SCE_CES_ERROR_INVALID_DST_BUFFER returns, it means that a NULL pointer has been passed to *ucs2buf*.

If SCE_CES_ERROR_DST_BUFFER_END returns, it means that the buffer specified as output destination has run out, and that the character string could not be output completely.

The value that returns to *utf8Len* and *ucs2Len* in case of an error is the value up to the character processed successfully.

Use `sceCesMbcStrGetUcs2Len()` if you only want to look up output character string length.

Notes

This function is not multi-thread safe.
Context must be divided for multithreading.

See Also

`sceCesMbcUcsContextInit()`, `sceCesUcs2StrToMbcStr()`

sceCesUtf32StrToMbcStr

Convert from UTF-32 character string to MBCS character string

Definition

```
#include <ces.h>
int sceCesUtf32StrToMbcStr (
    SceCesMbcUcsContext* context,
    const uint32_t *utf32str,
    uint32_t utf32max,
    uint32_t *utf32Len,
    uint8_t *mbcsBuf,
    uint32_t mbcsMax,
    uint32_t *mbcsLen
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>utf32str</i>	Address of the buffer storing UTF-32 character string
<i>utf32max</i>	Size of the buffer storing UTF-32 character string
<i>utf32Len</i>	Address of the variable for receiving UTF-32 character string length (32-bit word count)
<i>mbcsBuf</i>	Address of the buffer for receiving MBCS character string
<i>mbcsMax</i>	Size (byte count) of the buffer for receiving MBCS character string
<i>mbcsLen</i>	Address of the variable for receiving MBCS character string length (byte count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

This function converts UTF-32 character strings to MBCS character strings.

Specify the context initialized with `sceCesMbcUcsContextInit()` in *context*.

Specify the address of the buffer storing the UTF-32 character string in *utf32str*.

Specify the size (32-bit word count) of the buffer storing the UTF-32 character string in *utf32max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (32-bit word count) of the UTF-32 character string used for conversion to MBCS will return to *utf32Len*. The NULL termination character will not be included.

In *mbcsBuf*, specify the address of the buffer storing the character string specified in *utf32str* after conversion to MBCS. NULL termination is always guaranteed for the buffer passed to this argument.

Specify the size (byte count) allowed for writing to *mbcsBuf* in *mbcsMax*. Specify a size taking into account the part for writing the NULL termination character.

The length (byte count) of the character string successfully converted to MBCS will return to *mbcsLen*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *utf32max*, and that all processing has been completed normally.

If SCE_CES_ERROR_INVALID_PARAMETER returns, it means that a NULL pointer has been specified to *context*.

If SCE_CES_ERROR_INVALID_PROFILE returns, it means that the profile set in *context* has been determined to be invalid.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed to *utf32str*. 0 is stored in both *utf32Len* and *utf8Len*.

If SCE_CES_ERROR_INVALID_ENCODE returns, it means that encoding has been determined to be invalid because a value equal or greater than 0x00110000, which is outside the UTF-32 valid range, has been found in the character string specified in *utf32str*.

If SCE_CES_ERROR_ILLEGAL_CODE returns, it means that the code has been determined to be illegal because code in the U+D800 to U+DFFF range reserved as a surrogate area was found in the character string specified in *utf32str*.

The return of SCE_CES_ERROR_UNASSIGNED_CODE indicates that the character string specified in *utf32str* contains characters that cannot be represented in MBCS.

If SCE_CES_ERROR_INVALID_DST_BUFFER returns, it means that a NULL pointer has been passed to *mbcsBuf*.

If SCE_CES_ERROR_DST_BUFFER_END returns, it means that the buffer specified as output destination has run out, and that the character string could not be output completely.

The value that returns to *utf32Len* and *mbcsLen* in case of an error is the value up to the character processed successfully.

Use `sceCesUtf32StrGetMbcLen()` if you only want to look up character string length.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`sceCesMbcUcsContextInit()`, `sceCesUtf32StrGetMbcLen()`,
`sceCesMbcStrToUtf32Str()`

SCE CONFIDENTIAL

sceCesUtf16StrToMbcStr

Convert from UTF-16 character string to MBCS character string

Definition

```
#include <ces.h>
int sceCesUtf16StrToMbcStr (
    SceCesMbcUcsContext* context,
    const uint16_t *utf16str,
    uint32_t utf16max,
    uint32_t *utf16Len,
    uint8_t *mbcsBuf,
    uint32_t mbcsMax,
    uint32_t *mbcsLen
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>utf16str</i>	Address of the buffer storing UTF-16 character string
<i>utf16max</i>	Size (16-bit word count) of the buffer storing UTF-16 character string
<i>utf16Len</i>	Address of the variable for receiving UTF-16 character string length (16-bit word count)
<i>mbcsBuf</i>	Address of the buffer for receiving MBCS character string
<i>mbcsMax</i>	Size (byte count) of the buffer for receiving MBCS character string
<i>mbcsLen</i>	Address of the variable for receiving MBCS character string length (byte count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

This function converts UTF-16 character strings to MBCS character strings.

Specify the context initialized with `sceCesMbcUcsContextInit()` in *context*.

Specify the address of the buffer storing the UTF-16 character string in *utf16str*.

SCE CONFIDENTIAL

Specify the size (16-bit word count) of the buffer storing the UTF-16 character string in *utf16max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (16-bit word count) of the UTF-16 character string used for conversion to UTF-8 will return to *utf16Len*. The NULL termination character will not be included.

In *mbcsBuf*, specify the address of the buffer storing the character string specified in *utf8str* after conversion to MBCS. NULL termination is always guaranteed for the buffer passed to this argument.

Specify the size (byte count) allowed for writing to *mbcsBuf* in *mbcsMax*. Specify a size taking into account the part for writing the NULL termination character.

The length (byte count) of the character string successfully converted to MBCS will return to *mbcsLen*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *utf16max*, and that all processing has been completed normally.

If SCE_CES_ERROR_INVALID_PARAMETER returns, it means that a NULL pointer has been specified to *context*.

If SCE_CES_ERROR_INVALID_PROFILE returns, it means that the profile set in *context* has been determined to be invalid.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed to *utf16str*. 0 is stored in both *utf16Len* and *mbcsLen*.

If SCE_CES_ERROR_SRC_BUFFER_END returns, it means that the character string specified in *utf16str* has been interrupted in the midst of a code representing one character due to limitation by *utf16max*.

If SCE_CES_ERROR_ILLEGAL_CODE returns, it means the code has been determined to be illegal because codes in the range between U+D800 and U+DFFF that is reserved as surrogate area were not used in pairs in the character string specified in *utf16str*.

The return of SCE_CES_ERROR_UNASSIGNED_CODE indicates that the character string specified in *utf16str* contains characters that cannot be represented in MBCS.

If SCE_CES_ERROR_INVALID_DST_BUFFER returns, it means that a NULL pointer has been passed to *mbcsBuf*.

If SCE_CES_ERROR_DST_BUFFER_END returns, it means that the buffer specified as output destination has run out, and that the character string could not be output completely.

The value that returns to *utf16Len* and *mbcsLen* in case of an error is the value up to the character processed successfully.

Use `sceCesUtf16StrGetMbcLen()` if you only want to look up character string length.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`sceCesMbcUcsContextInit()`, `sceCesUtf16StrGetMbcLen()`,
`sceCesUtf16StrToMbcStr()`

SCE CONFIDENTIAL

sceCesUtf8StrToMbcStr

Convert from UTF-8 character string to MBCS character string

Definition

```
#include <ces.h>
int sceCesUtf8StrToMbcStr(
    SceCesMbcUcsContext* context,
    const uint8_t *utf8str,
    uint32_t utf8max,
    uint32_t *utf8Len,
    uint8_t *mbcsBuf,
    uint32_t mbcMax,
    uint32_t *mbcsLen
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>utf8str</i>	Address of the buffer storing UTF-8 character string
<i>utf8max</i>	Size of the buffer storing UTF-8 character string
<i>utf8Len</i>	Address of the variable for receiving UTF-8 character string length (byte count)
<i>mbcsBuf</i>	Address of the buffer for receiving MBCS character string
<i>mbcsMax</i>	Size (byte count) of the buffer for receiving MBCS character string
<i>mbcsLen</i>	Address of the variable for receiving MBCS character string length (byte count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

This function converts UTF-8 character strings to MBCS character strings.

Specify the context initialized with `sceCesMbcUcsContextInit()` in *context*.

Specify the address of the buffer storing the UTF-8 character string in *utf8str*.

SCE CONFIDENTIAL

Specify the size (byte count) of the buffer storing the UTF-8 character string in *utf8max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (byte count) of the UTF-8 character string used for conversion to MBCS will return to *utf8Len*. The NULL termination character will not be included.

In *mbcsBuf*, specify the address of the buffer storing the character string specified in *utf8str* after conversion to MBCS. NULL termination is always guaranteed for the buffer passed to this argument.

Specify the size (byte count) allowed for writing to *mbcsBuf* in *mbcsMax*. Specify a size taking into account the part for writing the NULL termination character.

The length (byte count) of the character string successfully converted to MBCS will return to *mbcsLen*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *utf8max*, and that all processing has been completed normally.

If SCE_CES_ERROR_INVALID_PARAMETER returns, it means that a NULL pointer has been specified to *context*.

If SCE_CES_ERROR_INVALID_PROFILE returns, it means that the profile set in *context* has been determined to be invalid.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed to *utf8str*. 0 is stored in both *utf8Len* and *mbcsLen*.

If SCE_CES_ERROR_SRC_BUFFER_END returns, it means that the character string specified in *utf8str* has been interrupted in the midst of a code representing one character due to limitation by *utf8max*.

If SCE_CES_ERROR_INVALID_ENCODE returns, it means encoding has been determined to be invalid because the character string specified in *utf8str* contains byte strings that cannot be recognized as UTF-8.

If SCE_CES_ERROR_ILLEGAL_CODE returns, it means that the code has been found invalid because the character string specified in *utf8str* contains codes in the U+D800 to U+DFFF range reserved as a surrogate area, or encoding with an unnecessarily large number of bytes (such as representing U+0000 as the 2-byte sequence C0,80).

SCE_CES_ERROR_UNASSIGNED_CODE indicates that the character string specified in *utf8str* contains characters that cannot be represented in MBCS.

If SCE_CES_ERROR_INVALID_DST_BUFFER returns, it means that a NULL pointer has been passed to *mbcsBuf*.

If SCE_CES_ERROR_DST_BUFFER_END returns, it means that the buffer specified as output destination has run out, and that the character string could not be output completely.

The value that returns to *utf8Len* and *mbcsLen* in case of an error is the value up to the character processed successfully.

Use `sceCesUtf8StrGetMbcLen()` if you only want to look up character string length.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`sceCesMbcUcsContextInit()`, `sceCesUtf8StrGetMbcLen()`,
`sceCesMbcStrToUtf8Str()`

SCE CONFIDENTIAL

sceCesUcs2StrToMbcStr

Convert from UCS-2 character string to MBCS character string

Definition

```
#include <ces.h>
int sceCesUcs2StrToMbcStr(
    SceCesMbcUcsContext* context,
    const uint16_t *ucs2str,
    uint32_t ucs2max,
    uint32_t *ucs2Len,
    uint8_t *mbcsBuf,
    uint32_t mbcsMax,
    uint32_t *mbcsLen
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>ucs2str</i>	Address of the buffer storing UCS-2 character string
<i>ucs2max</i>	Size (16-bit word count) of the buffer storing UCS-2 character string
<i>ucs2Len</i>	Address of the variable for receiving UCS-2 character string length (16-bit word count)
<i>mbcsBuf</i>	Address of the buffer for receiving MBCS character string
<i>mbcsMax</i>	Size (byte count) of the buffer for receiving MBCS character string
<i>mbcsLen</i>	Address of the variable for receiving MBCS character string length (byte count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

This function converts UCS-2 character strings to MBCS character strings.

(This function treats UCS-2 code arrays as character strings terminating with U+00000.)

Specify the context initialized with `sceCesMbcUcsContextInit()` in *context*.

Specify the address of the buffer storing the UCS-2 character string in *ucs2str*.

SCE CONFIDENTIAL

Specify the size (16-bit word count) of the buffer storing the UCS-2 character string in *ucs2max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (16-bit word count) of the UCS-2 character string used for conversion to MBCS will return to *ucs2Len*. The NULL termination character will not be included.

In *mbcsBuf*, specify the address of the buffer storing the character string specified in *ucs2str* after conversion to MBCS. NULL termination is always guaranteed for the buffer passed to this argument.

Specify the size (byte count) allowed for writing to *mbcsBuf* in *mbcsMax*. Specify a size taking into account the part for writing the NULL termination character.

The length (byte count) of the character string successfully converted to MBCS will return to *mbcsLen*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *ucs2max*, and that all processing has been completed normally.

If SCE_CES_ERROR_INVALID_PARAMETER returns, it means that a NULL pointer has been specified to *context*.

If SCE_CES_ERROR_INVALID_PROFILE returns, it means that the profile set in *context* has been determined to be invalid.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed to *ucs2str*. 0 is stored in both *ucs2Len* and *mbcsLen*.

If SCE_CES_ERROR_ILLEGAL_CODE returns, it means that the code has been determined to be illegal because code in the U+D800 to U+DFFF range reserved as a surrogate area was found in the character string specified in *ucs2str*.

SCE_CES_ERROR_UNASSIGNED_CODE indicates that the character string specified in *ucs2str* contains characters that cannot be represented in MBCS.

If SCE_CES_ERROR_INVALID_DST_BUFFER returns, it means that a NULL pointer has been passed to *mbcsBuf*.

If SCE_CES_ERROR_DST_BUFFER_END returns, it means that the buffer specified as output destination has run out, and that the character string could not be output completely.

The value that returns to *ucs2Len* and *mbcsLen* in case of an error is the value up to the character processed successfully.

Use `sceCesUcs2StrGetMbcLen()` if you only want to look up character string length.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`sceCesMbcUcsContextInit()`, `sceCesUcs2StrGetMbcLen()`,
`sceCesMbcStrToUcs2Str()`

Character String Length Retrieval Functions for Handling MBCS (Including SBCS) and UCS

sceCesMbcStrGetUtf32Len

Retrieve character string length of MBCS character strings in UTF-32

Definition

```
#include <ces.h>
int sceCesMbcStrGetUtf32Len (
    SceCesMbcUcsContext* context,
    const uint8_t *mbcsStr,
    uint32_t mbcsMax,
    uint32_t *mbcsLen,
    uint32_t *utf32Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>mbcsStr</i>	Address of the buffer storing MBCS character string
<i>mbcsMax</i>	Size (byte count) of the buffer storing MBCS character string
<i>mbcsLen</i>	Address of the variable for receiving MBCS character string length (byte count)
<i>utf32Len</i>	Address of the variable for receiving UTF-32 character string length (32-bit word count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined

Description

This function retrieves character string length of MBCS character strings when converted to UTF-32 character strings.

Specify the context initialized with `sceCesMbcUcsContextInit()` in *context*.

Specify the address of the buffer storing the MBCS character string in *mbcsStr*. It recognizes character strings up to the NULL termination character.

SCE CONFIDENTIAL

Specify the size (byte count) of the buffer storing the MBCS character string in *mbcsMax*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (32-bit word count) of the UTF-32 character string used for conversion to MBCS will return to *mbcsLen*. The NULL termination character will not be included.

The length (32-bit word count) of the character string successfully converted to UTF-32 will return to *utf32Len*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *mbcsMax*, and that all processing has been completed normally.

If SCE_CES_ERROR_INVALID_PARAMETER returns, it means that a NULL pointer has been specified to *context*.

If SCE_CES_ERROR_INVALID_PROFILE returns, it means that the profile set in *context* has been determined to be invalid.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed to *mbcsStr*. 0 is stored in both *mbcsLen* and *utf32Len*.

If SCE_CES_ERROR_INVALID_ENCODE returns, it means that the encoding has been determined to be invalid because it includes byte values that cannot be recognized, such as in the case where the byte string passed to *mbcsStr* includes values within the MBCS's holding area.

The return of SCE_CES_ERROR_UNASSIGNED_CODE indicates that the character string specified in *mbcsStr* included characters without mapping to UCS.

The value that returns to *mbcsLen* and *utf32Len* in case of an error is the value up to the character processed successfully.

Use `sceCesMbcStrToUtf32Str()` if you want to retrieve the converted character string itself.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`sceCesMbcUcsContextInit()`, `sceCesMbcStrToUtf32Str()`

sceCesMbcStrGetUtf16Len

Retrieve character string length of MBCS character strings in UTF-16

Definition

```
#include <ces.h>
int sceCesMbcStrGetUtf16Len (
    SceCesMbcUcsContext* context,
    const uint8_t *mbcsStr,
    uint32_t mbcMax,
    uint32_t *mbcsLen,
    uint32_t *utf16Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>mbcsStr</i>	Address of the buffer storing MBCS character string
<i>mbcsMax</i>	Size (byte count) of the buffer storing MBCS character string
<i>mbcsLen</i>	Address of the variable for receiving MBCS character string length (byte count)
<i>utf16Len</i>	Address of the variable for receiving UTF-16 character string length (16-bit word count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined

Description

This function retrieves character string length of MBCS character strings when converted to UTF-16 character strings.

Specify the context initialized with `sceCesMbcUcsContextInit()` in *context*.

Specify the address of the buffer storing the MBCS character string in *mbcsStr*.

Specify the size (byte count) of the buffer storing the MBCS character string in *mbcsMax*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (16-bit word count) of the UTF-16 character string used for conversion to MBCS will return to *mbcsLen*. The NULL termination character will not be included.

The length (16-bit word count) of the character string successfully converted to UTF-16 will return to *utf16Len*. The NULL termination character will not be included.

SCE CONFIDENTIAL

If `SCE_OK` returns, it means that processing has been completed up to the NULL termination character or to the character range specified in `mbcsMax`, and that all processing has been completed normally.

If `SCE_CES_ERROR_INVALID_PARAMETER` returns, it means that a NULL pointer has been specified to `context`.

If `SCE_CES_ERROR_INVALID_PROFILE` returns, it means that the profile set in `context` has been determined to be invalid.

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed to `mbcsStr`. 0 is stored in both `mbcsLen` and `utf16Len`.

If `SCE_CES_ERROR_INVALID_ENCODE` returns, it means that the encoding has been determined to be invalid because it includes byte values that cannot be recognized, such as in the case where the byte string passed to `mbcsStr` includes values within the MBCS's holding area.

The return of `SCE_CES_ERROR_UNASSIGNED_CODE` indicates that the character string specified in `mbcsStr` included characters without mapping to UCS.

The value that returns to `mbcsLen` and `utf16Len` in case of an error is the value up to the character processed successfully.

Use `sceCesMbcStrToUtf16Str()` if you want to retrieve the converted character string itself.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`sceCesMbcUcsContextInit()`, `sceCesMbcStrToUtf16Str()`

sceCesMbcStrGetUtf8Len

Retrieve character string length of MBCS character strings in UTF-8

Definition

```
#include <ces.h>
int sceCesMbcStrGetUtf8Len (
    SceCesMbcUcsContext* context,
    const uint8_t *mbcsStr,
    uint32_t mbcMax,
    uint32_t *mbcsLen,
    uint32_t *utf8Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>mbcsStr</i>	Address of the buffer storing MBCS character string
<i>mbcMax</i>	Size (byte count) of the buffer storing MBCS character string
<i>mbcsLen</i>	Address of the variable for receiving MBCS character string length (byte count)
<i>utf8Len</i>	Address of the variable for receiving UTF-8 character string length (byte count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined

Description

This function retrieves character string length of MBCS character strings when converted to UTF-8 character strings.

Specify the context initialized with `sceCesMbcUcsContextInit()` in *context*.

Specify the address of the buffer storing the MBCS character string in *mbcsStr*.

Specify the size (byte count) of the buffer storing the MBCS character string in *mbcMax*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (byte count) of the UTF-8 character string used for conversion to MBCS will return to *mbcsLen*. The NULL termination character will not be included.

The length (byte count) of the character string successfully converted to UTF-8 will return to *utf8Len*. The NULL termination character will not be included.

SCE CONFIDENTIAL

If `SCE_OK` returns, it means that processing has been completed up to the NULL termination character or to the character range specified in `mbcsMax`, and that all processing has been completed normally.

If `SCE_CES_ERROR_INVALID_PARAMETER` returns, it means that a NULL pointer has been specified to `context`.

If `SCE_CES_ERROR_INVALID_PROFILE` returns, it means that the profile set in `context` has been determined to be invalid.

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed to `mbcsStr`. 0 is stored in both `mbcsLen` and `utf8Len`.

If `SCE_CES_ERROR_INVALID_ENCODE` returns, it means that the encoding has been determined to be invalid because it includes byte values that cannot be recognized, such as in the case where the byte string passed to `mbcsStr` includes values within the MBCS's holding area.

The return of `SCE_CES_ERROR_UNASSIGNED_CODE` indicates that the character string specified in `mbcsStr` included characters without mapping to UCS.

The value that returns to `mbcsLen` and `utf8Len` in case of an error is the value up to the character processed successfully.

Use `sceCesMbcStrToUtf8Str()` if you want to retrieve the converted character string itself.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`sceCesMbcUcsContextInit()`, `sceCesMbcStrToUtf8Str()`

sceCesMbcStrGetUcs2Len

Retrieve the number of character codes of MBCS character strings in UCS-2

Definition

```
#include <ces.h>
int sceCesMbcStrGetUcs2Len (
    SceCesMbcUcsContext* context,
    const uint8_t *mbcsStr,
    uint32_t mbcMax,
    uint32_t *mbcsLen,
    uint32_t *ucs2Len
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>mbcsStr</i>	Address of the buffer storing MBCS character string
<i>mbcMax</i>	Size (byte count) of the buffer storing MBCS character string
<i>mbcsLen</i>	Address of the variable for receiving MBCS character string length (byte count)
<i>ucs2Len</i>	Address of the variable for receiving UCS-2 character string length (16-bit word count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined

Description

This function retrieves character string length of MBCS character strings when converted to UCS-2 character strings.

(This function treats UCS-2 code arrays as character strings terminating with U+0000.)

Specify the context initialized with `sceCesMbcUcsContextInit()` in *context*.

Specify the address of the buffer storing the MBCS character string in *mbcsStr*. It recognizes character strings up to the NULL termination character.

Specify the size (byte count) of the buffer storing the MBCS character string in *mbcMax*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (byte count) of the MBCS character string used for conversion to UCS-2 will return to *mbcsLen*. The termination character will not be included.

SCE CONFIDENTIAL

The length (16-bit word count) of the character string successfully converted to UCS-2 will return to *ucs2Len*. The NULL termination character will not be included.

If *SCE_OK* returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *mbcsMax*, and that all processing has been completed normally.

If *SCE_CES_ERROR_INVALID_PARAMETER* returns, it means that a NULL pointer has been specified to *context*.

If *SCE_CES_ERROR_INVALID_PROFILE* returns, it means that the profile set in *context* has been determined to be invalid.

If *SCE_CES_ERROR_INVALID_SRC_BUFFER* returns, it means that a NULL pointer has been passed to *mbcsStr*. 0 is stored in both *mbcsLen* and *ucs2Len*.

If *SCE_CES_ERROR_INVALID_ENCODE* returns, it means that the encoding has been determined to be invalid because it includes byte values that cannot be recognized, such as in the case where the byte string passed to *mbcsStr* includes values within the MBCS's holding area.

The return of *SCE_CES_ERROR_UNASSIGNED_CODE* indicates that the character string specified in *mbcsStr* included characters without mapping to UCS.

The value that returns to *utf8Len* and *ucs2Len* in case of an error is the value up to the character processed successfully.

Use *sceCesMbcStrToUcs2Str()* if you want to retrieve the converted character string itself.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

sceCesMbcUcsContextInit(), *sceCesUcs2StrToMbcStr()*

sceCesUtf32StrGetMbcLen

Retrieve character string length of UTF-32 character strings in MBCS

Definition

```
#include <ces.h>
int sceCesUtf32StrGetMbcLen (
    SceCesMbcUcsContext* context,
    const uint32_t *utf32str,
    uint32_t utf32max,
    uint32_t *utf32Len,
    uint32_t *mbcsLen
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>utf32str</i>	Address of the buffer storing UTF-32 character string
<i>utf32max</i>	Size of the buffer storing UTF-32 character string
<i>utf32Len</i>	Address of the variable for receiving UTF-32 character string length (32-bit word count)
<i>mbcsLen</i>	Address of the variable for receiving MBCS character string length (byte count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined

Description

This function retrieves character string length of UTF-32 character strings when converted to MBCS character strings.

Specify the context initialized with `sceCesMbcUcsContextInit()` in *context*.

Specify the address of the buffer storing the UTF-32 character string in *utf32str*.

Specify the size (32-bit word count) of the buffer storing the UTF-32 character string in *utf32max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (32-bit word count) of the UTF-32 character string used for conversion to MBCS will return to *utf32Len*. The NULL termination character will not be included.

The length (byte count) of the character string successfully converted to MBCS will return to *mbcsLen*. The NULL termination character will not be included.

SCE CONFIDENTIAL

If `SCE_OK` returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *utf32max*, and that all processing has been completed normally.

If `SCE_CES_ERROR_INVALID_PARAMETER` returns, it means that a NULL pointer has been specified to *context*.

If `SCE_CES_ERROR_INVALID_PROFILE` returns, it means that the profile set in *context* has been determined to be invalid.

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed to *utf32str*. 0 is stored in both *utf32Len* and *utf8Len*.

If `SCE_CES_ERROR_INVALID_ENCODE` returns, it means that encoding has been determined to be invalid because a value equal or greater than 0x00110000, which is outside the UTF-32 valid range, has been found in the character string specified in *utf32str*.

If `SCE_CES_ERROR_ILLEGAL_CODE` returns, it means that the code has been determined to be illegal because code in the U+D800 to U+DFFF range reserved as a surrogate area was found in the character string specified in *utf32str*.

`SCE_CES_ERROR_UNASSIGNED_CODE` indicates that the character string specified in *utf32str* contains characters that cannot be represented in MBCS.

The value that returns to *utf32Len* and *mbcsLen* in case of an error is the value up to the character processed successfully.

Use `sceCesUtf32StrToMbcStr()` if you want to retrieve the converted character string itself.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`sceCesMbcUcsContextInit()`, `sceCesUtf32StrToMbcStr()`

sceCesUtf16StrGetMbcLen

Retrieve character string length of UTF-16 character strings in MBCS

Definition

```
#include <ces.h>
int sceCesUtf16StrGetMbcLen (
    SceCesMbcUcsContext* context,
    const uint16_t *utf16str,
    uint32_t utf16max,
    uint32_t *utf16Len,
    uint32_t *mbcsLen
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>utf16str</i>	Address of the buffer storing UTF-16 character string
<i>utf16max</i>	Size (16-bit word count) of the buffer storing UTF-16 character string
<i>utf16Len</i>	Address of the variable for receiving UTF-16 character string length (16-bit word count)
<i>mbcsLen</i>	Address of the variable for receiving MBCS character string length (byte count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined

Description

This function retrieves character string length of UTF-16 character strings when converted to MBCS character strings.

Specify the context initialized with `sceCesMbcUcsContextInit()` in *context*.

Specify the address of the buffer storing the UTF-16 character string in *utf16str*.

Specify the size (16-bit word count) of the buffer storing the UTF-16 character string in *utf16max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (16-bit word count) of the UTF-16 character string used for conversion to UTF-8 will return to *utf16Len*. The NULL termination character will not be included.

The length (byte count) of the character string successfully converted to MBCS will return to *mbcsLen*. The NULL termination character will not be included.

If `SCE_OK` returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *utf16max*, and that all processing has been completed normally.

If `SCE_CES_ERROR_INVALID_PARAMETER` returns, it means that a NULL pointer has been specified to *context*.

If `SCE_CES_ERROR_INVALID_PROFILE` returns, it means that the profile set in *context* has been determined to be invalid.

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed to *utf16str*. 0 is stored in both *utf16Len* and *mbcsLen*.

If `SCE_CES_ERROR_SRC_BUFFER_END` returns, it means that the character string specified in *utf16str* has been interrupted in the midst of a code representing one character due to limitation by *utf16max*.

If `SCE_CES_ERROR_ILLEGAL_CODE` returns, it means the code has been determined to be illegal because codes in the range between U+D800 and U+DFFF that is reserved as surrogate area were not used in pairs in the character string specified in *utf16str*.

`SCE_CES_ERROR_UNASSIGNED_CODE` indicates that the character string specified in *utf16str* contains characters that cannot be represented in MBCS.

The value that returns to *utf16Len* and *mbcsLen* in case of an error is the value up to the character processed successfully.

Use `sceCesUtf16StrToMbcStr()` if you want to retrieve the converted character string itself.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`sceCesMbcUcsContextInit()`, `sceCesUtf16StrToMbcStr()`

SCE CONFIDENTIAL

sceCesUtf8StrGetMbcLen

Retrieve character string length of UTF-8 character strings in MBCS

Definition

```
#include <ces.h>
int sceCesUtf8StrGetMbcLen (
    SceCesMbcUcsContext* context,
    const uint8_t *utf8str,
    uint32_t utf8max,
    uint32_t *utf8Len,
    uint32_t *mbcsLen
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>utf8str</i>	Address of the buffer storing UTF-8 character string
<i>utf8max</i>	Size of the buffer storing UTF-8 character string
<i>utf8Len</i>	Address of the variable for receiving UTF-8 character string length (byte count)
<i>mbcsLen</i>	Address of the variable for receiving MBCS character string length (byte count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined

Description

This function retrieves character string length of UTF-8 character strings when converted to MBCS character strings.

Specify the context initialized with `sceCesMbcUcsContextInit()` in *context*.

Specify the address of the buffer storing the UTF-8 character string in *utf8str*.

Specify the size (byte count) of the buffer storing the UTF-8 character string in *utf8max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (byte count) of the UTF-8 character string used for conversion to MBCS will return to *utf8Len*. The NULL termination character will not be included.

SCE CONFIDENTIAL

The length (byte count) of the character string successfully converted to MBCS will return to *mbcsLen*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *utf8max*, and that all processing has been completed normally.

If SCE_CES_ERROR_INVALID_PARAMETER returns, it means that a NULL pointer has been specified to *context*.

If SCE_CES_ERROR_INVALID_PROFILE returns, it means that the profile set in *context* has been determined to be invalid.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed to *utf8str*. 0 is stored in both *utf8Len* and *mbcsLen*.

If SCE_CES_ERROR_SRC_BUFFER_END returns, it means that the character string specified in *utf8str* has been interrupted in the midst of a code representing one character due to limitation by *utf8max*.

If SCE_CES_ERROR_INVALID_ENCODE returns, it means encoding has been determined to be invalid because the character string specified in *utf8str* contains byte strings that cannot be recognized as UTF-8.

If SCE_CES_ERROR_ILLEGAL_CODE returns, it means that the code has been found invalid because the character string specified in *utf8str* contains codes in the U+D800 to U+DFFF range reserved as a surrogate area, or encoding with an unnecessarily large number of bytes (such as representing U+0000 as the 2-byte sequence C0,80).

SCE_CES_ERROR_UNASSIGNED_CODE indicates that the character string specified in *utf8str* contains characters that cannot be represented in MBCS.

The value that returns to *utf8Len* and *mbcsLen* in case of an error is the value up to the character processed successfully.

Use `sceCesUtf8StrToMbcStr()` if you want to retrieve the converted character string itself.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`sceCesMbcUcsContextInit()`, `sceCesUtf8StrToMbcStr()`

sceCesUcs2StrGetMbcLen

Retrieve character string length of UCS-2 character strings in MBCS

Definition

```
#include <ces.h>
int sceCesUcs2StrGetMbcLen (
    SceCesMbcUcsContext* context,
    const uint16_t *ucs2str,
    uint32_t ucs2max,
    uint32_t *ucs2Len,
    uint32_t *mbcsLen
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>ucs2str</i>	Address of the buffer storing UCS-2 character string
<i>ucs2max</i>	Size (16-bit word count) of the buffer storing UCS-2 character string
<i>ucs2Len</i>	Address of the variable for receiving UCS-2 character string length (16-bit word count)
<i>mbcsLen</i>	Address of the variable for receiving MBCS character string length (byte count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined

Description

This function retrieves character string length of UCS-2 character strings when converted to MBCS character strings.

(This function treats UCS-2 code arrays as character strings terminating with U+00000.)

Specify the context initialized with `sceCesMbcUcsContextInit()` in *context*.

Specify the address of the buffer storing the UCS-2 character string in *ucs2str*.

Specify the size (16-bit word count) of the buffer storing the UCS-2 character string in *ucs2max*. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified.

The length (16-bit word count) of the UCS-2 character string used for conversion to MBCS will return to *ucs2Len*. The NULL termination character will not be included.

In *mbcsBuf*, specify the address of the buffer storing the character string specified in *ucs2str* after conversion to MBCS. NULL termination is always guaranteed for the buffer passed to this argument.

SCE CONFIDENTIAL

Specify the size (byte count) allowed for writing to *mbcsBuf* in *mbcsMax*. Specify a size taking into account the part for writing the NULL termination character.

The length (byte count) of the character string successfully converted to MBCS will return to *mbcsLen*. The NULL termination character will not be included.

If SCE_OK returns, it means that processing has been completed up to the NULL termination character or to the character range specified in *ucs2max*, and that all processing has been completed normally.

If SCE_CES_ERROR_INVALID_PARAMETER returns, it means that a NULL pointer has been specified to *context*.

If SCE_CES_ERROR_INVALID_PROFILE returns, it means that the profile set in *context* has been determined to be invalid.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed to *ucs2str*. 0 is stored in both *ucs2Len* and *mbcsLen*.

If SCE_CES_ERROR_ILLEGAL_CODE returns, it means that the code has been determined to be illegal because code in the U+D800 to U+DFFF range reserved as a surrogate area was found in the character string specified in *ucs2str*.

SCE_CES_ERROR_UNASSIGNED_CODE indicates that the character string specified in *ucs2str* contains characters that cannot be represented in MBCS.

If SCE_CES_ERROR_DST_BUFFER_END returns, it means that the buffer specified as output destination has run out, and that the character string could not be output completely.

The value that returns to *ucs2Len* and *mbcsLen* in case of an error is the value up to the character processed successfully.

Use `sceCesUcs2StrToMbcStr()` if you want to retrieve the converted character string itself.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`sceCesMbcUcsContextInit()`, `sceCesUcs2StrToMbcStr()`

Character String Functions for Handling UCS Integrated Interconversion

SCE CONFIDENTIAL

sceCesUcsStrGetEncodingSize

Retrieve size of character string in specified CES

Definition

```
#include <ces.h>
int sceCesUcsStrGetEncodingSize (
    SceCesUcsContext* context,
    int srcCes,
    const void *srcStr,
    uint32_t srcMax,
    uint32_t *srcLen,
    int dstCes,
    uint32_t *dstSize
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>srcCes</i>	Current character string encoding scheme
<i>srcStr</i>	Address of the buffer storing character string
<i>srcMax</i>	Size (byte count) of the buffer storing character string
<i>srcLen</i>	Address of the variable for receiving recognized length as character string (byte count)
<i>dstCes</i>	Encoding scheme of the character string whose size is to be retrieved
<i>dstSize</i>	Address of the variable for receiving the size (byte count) of the character string in CES specified in <i>dstCes</i>

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_OUT_OF_CODE_RANGE	0x805C0024	Character outside the representable code range of the output destination encoding scheme is detected

Description

This function retrieves the size of the character string when converted to the specified encoding scheme.

Specify the address of `SceCesUcsContext` initialized with `sceCesUcsContextInit()`, or the address of the `cesUcsCtx` member of `SceCesMbcUcsContext` initialized with `sceCesMbcUcsContextInit()` in `context`. If the former is passed to the argument, only conversion among Unicode CESs will be supported. If the latter is passed to the argument, interconversion among encoding schemes other than Unicode specified at the time of `SceCesMbcUcsContext` initialization will be supported.

Specify the current CES of the source character string with one of the following values in `srcCes`.

Value	Description
<code>SCE_CES_UTF32</code>	UTF-32
<code>SCE_CES_UTF32BE</code>	UTF-32 (big-endian)
<code>SCE_CES_UTF32LE</code>	UTF-32 (little-endian)
<code>SCE_CES_UTF16</code>	UTF-16
<code>SCE_CES_UTF16BE</code>	UTF-16 (big-endian)
<code>SCE_CES_UTF16LE</code>	UTF-16 (little-endian)
<code>SCE_CES_UTF8</code>	UTF-8
<code>SCE_CES_UCS2</code>	UCS-2
<code>SCE_CES_MBCS (*)</code>	Multi-byte character sets other than Unicode (<i>context</i> must be initialized with <code>sceCesMbcUcsContextInit()</code>)

(*) It is not possible to specify `SCE_CES_MBCS` in both `srcCes` and `dstCes`.

The endianness of `SCE_CES_UTF32` and `SCE_CES_UTF16` is set by `sceCesSetUcsPolicyDetectBom()`, `sceCesSetUtf32StrEndian()` and `sceCesSetUtf16StrEndian()`.

Specify the address of the buffer storing the source character string in `srcStr`.

Specify the size (byte count) of the buffer storing the source character string in `srcMax`. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified. In this case, `srcStr` termination must always be the NULL termination character.

The length (byte count) of the source character string of the converted character string will return to `srcLen`. The NULL termination character will not be included.

Specify the character encoding scheme of the character string whose size you wish to retrieve in `dstCes`. Settable values are the same as for `srcCes`.

`dstSize` stores the buffer size (byte count) necessary for the character string specified in `srcStr` when represented in the encoding scheme specified with `dstCes`. The value stored here includes the size of the NULL termination character.

If `SCE_OK` returns, it means that processing has been completed up to the NULL termination character or to the character range specified in `srcMax`, and that all processing has been completed normally.

If `SCE_CES_ERROR_INVALID_PARAMETER` returns, it means that an error has been found because the value specified in `context` and for the encoding scheme is not appropriate.

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed to `srcStr`.

If `SCE_CES_ERROR_SRC_BUFFER_END` returns, it means that the character string specified in `srcStr` has been interrupted in the midst of a code representing one character due to limitation by `srcMax`. This error will not return in cases where a variable-length encoding scheme is not specified for the source character string.

SCE CONFIDENTIAL

If `SCE_CES_ERROR_INVALID_ENCODE` returns, it means that an error has been found because the CES specified with *srcCes* was unable to recognize the code of the character string specified in *srcStr*.

If `SCE_CES_ERROR_ILLEGAL_CODE` returns, it means that an error was found because the character string specified in *srcStr* contained code determined to be illegal for the encoding scheme specified in *srcCes*.

`SCE_CES_ERROR_OUT_OF_CODE_RANGE` returns when the characters in the CES specified with *srcCes* cannot be converted to the CES specified in *dstCes* because of representable code value range limitations. For example, this error returns when attempting to convert a U+00010000 character from UTF-32 to UCS-2.

If `SCE_CES_ERROR_UNASSIGNED_CODE` returns, it means that the characters in the CES specified with *srcCes* was not convertible to the CES specified in *dstCes*. This error will not occur in the case of conversion among Unicode character encoding schemes.

The value that returns to *srcLen* and *dstSize* in case of an error is the value up to the character processed successfully. Since *dstSize* includes the size of the NULL termination character, it will store the size of the NULL termination character (from 1 to 4) even if no characters have been converted successfully.

Notes

This function is not multi-thread safe.
Context must be divided for multithreading.

See Also

`SceCesUcsContext`, `sceCesUcsContextInit()`, `SceCesMbcUcsContext`,
`sceCesMbcUcsContextInit()`, `sceCesSetUcsPolicyDetectBom()`,
`sceCesSetUcsPolicyOutputBom()`, `sceCesSetUtf32StrEndian()`,
`sceCesSetUtf16StrEndian()`, `sceCesUcsStrConvertEncoding()`

SCE CONFIDENTIAL

sceCesUcsStrConvertEncoding

Interconvert with character strings in Unicode CES

Definition

```
#include <ces.h>
int sceCesUcsStrConvertEncoding(
    SceCesUcsContext* context,
    int srcCes,
    const void *srcStr,
    uint32_t srcMax,
    uint32_t *srcLen,
    int dstCes,
    void *dstBuf,
    uint32_t dstMax,
    uint32_t *dstLen
)
```

Arguments

<i>context</i>	Context for UCS character string conversion
<i>srcCes</i>	Current character string encoding scheme
<i>srcStr</i>	Address of the buffer storing character string
<i>srcMax</i>	Size (byte count) of the buffer storing character string
<i>srcLen</i>	Address of the variable for receiving recognized length as character string (byte count)
<i>dstCes</i>	CES to convert to
<i>dstBuf</i>	Address of the buffer for receiving character strings after conversion
<i>dstMax</i>	Size (byte count) of the buffer for receiving character strings after conversion
<i>dstLen</i>	Address of the variable for receiving the length (byte count) of the character string after conversion

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected in source character code
SCE_CES_ERROR_OUT_OF_CODE_RANGE	0x805C0024	Character outside the representable code range of the output destination encoding scheme is detected

Description

This function copies character strings converting them to a specified CES.

This function supports conversion and copying among Unicode encoding schemes. When using a context that supports conversion to other character sets, this function supports interconversion with CESs other than Unicode.

Specify the address of `SceCesUcsContext` initialized with `sceCesUcsContextInit()`, or the address of the `cesUcsCtx` member of `SceCesMbcUcsContext` initialized with `sceCesMbcUcsContextInit()` in `context`. If the former is passed to the argument, only conversion among Unicode CESs will be supported. If the latter is passed to the argument, interconversion among encoding schemes other than Unicode specified at the time of `SceCesMbcUcsContext` initialization will be supported.

Specify the current CES of the source character string with one of the following values in `srcCes`.

Value	Description
<code>SCE_CES_UTF32</code>	UTF-32
<code>SCE_CES_UTF32BE</code>	UTF-32 (big-endian)
<code>SCE_CES_UTF32LE</code>	UTF-32 (little-endian)
<code>SCE_CES_UTF16</code>	UTF-16
<code>SCE_CES_UTF16BE</code>	UTF-16 (big-endian)
<code>SCE_CES_UTF16LE</code>	UTF-16 (little-endian)
<code>SCE_CES_UTF8</code>	UTF-8
<code>SCE_CES_UCS2</code>	UCS-2
<code>SCE_CES_MBCS (*)</code>	Multi-byte character sets other than Unicode (<code>context</code> must be initialized with <code>sceCesMbcUcsContextInit()</code>)

(*) It is not possible to specify `SCE_CES_MBCS` in both `srcCes` and `dstCes`.

The endianness of `SCE_CES_UTF32` and `SCE_CES_UTF16` is set by `sceCesSetUcsPolicyDetectBom()`, `sceCesSetUtf32StrEndian()` and `sceCesSetUtf16StrEndian()`.

Specify the address of the buffer storing the source character string in `srcStr`.

Specify the size (byte count) of the buffer storing the source character string in `srcMax`. In this way character recognition will be limited within the specified range. Recognition will not be limited if 0 is specified. In this case, `srcStr` termination must always be the NULL termination character.

The length (byte count) of the source character string of the converted character string will return to `srcLen`. The NULL termination character will not be included.

Specify the character encoding scheme of the character string whose size you wish to retrieve in `dstCes`. Settable values are the same as for `srcCes`.

Specify the address of the buffer for receiving the converted character string in `dstBuf`.

Specify the size (byte count) of the buffer for receiving the converted character string in `dstMax`.

The length (byte count) of the converted character string will return to `dstLen`. The NULL termination character will not be included.

If `SCE_OK` returns, it means that processing has been completed up to the NULL termination character or to the character range specified in `srcMax`, and that all processing has been completed normally.

If `SCE_CES_ERROR_INVALID_PARAMETER` returns, it means that an error has been found because the value specified in `context` and for the encoding scheme is not appropriate.

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed to `srcStr`.

If `SCE_CES_ERROR_SRC_BUFFER_END` returns, it means that the character string specified in `srcStr` has been interrupted in the midst of a code representing one character due to limitation by `srcMax`.

SCE CONFIDENTIAL

This error will not return in cases where a variable-length encoding scheme is not specified for the source character string.

In case of the error above, 0 will be stored in both *srcLen* and *dstLen*.

If `SCE_CES_ERROR_INVALID_ENCODE` returns, it means that an error has been found because the CES specified with *srcCes* was unable to recognize the code of the character string specified in *srcStr*.

If `SCE_CES_ERROR_ILLEGAL_CODE` returns, it means that an error was found because the character string specified in *srcStr* contained code determined to be illegal for the encoding scheme specified in *srcCes*.

`SCE_CES_ERROR_OUT_OF_CODE_RANGE` returns when the characters in the CES specified with *srcCes* cannot be converted to the CES specified in *dstCes* because of representable code value range limitations. For example, this error returns when attempting to convert a U+00010000 character from UTF-32 to UCS-2.

If `SCE_CES_ERROR_UNASSIGNED_CODE` returns, it means that the characters in the CES specified with *srcCes* was not convertible to the CES specified in *dstCes*. This error will not occur in the case of conversion among Unicode character encoding schemes.

With this function, if a buffer size equal or greater than that of the NULL termination character is set for *dstBuf* and *dstMax*, NULL termination to *dstBuf* will always be guaranteed.

The value that returns to *srcLen* and *dstLen* is the value up to the character processed successfully. The unit is fixed to byte count irrespective of the specified CES.

Also, since a NULL termination character is written in *dstBuf*, *dstLen* will not correspond to the written size. When subsequently adding the size corresponding to the NULL termination character to the *dstLen* value, keep in mind that the byte size of the NULL termination character is different for each CES.

`sceCesUcsStrGetEncodingSize()` is provided as a function for retrieving size.

Notes

This function is not multi-thread safe.

Context must be divided for multithreading.

See Also

`SceCesUcsContext`, `sceCesUcsContextInit()`, `SceCesMbcUcsContext`,
`sceCesMbcUcsContextInit()`, `sceCesSetUcsPolicyDetectBom()`,
`sceCesSetUcsPolicyOutputBom()`, `sceCesSetUtf32StrEndian()`,
`sceCesSetUtf16StrEndian()`, `sceCesUcsStrGetEncodingSize()`

Setting Functions for Character String Control and Error Handling

Setting Functions for Character String Control

sceCesSetStrBegin, sceCesSetStrLast, sceCesSetStrEnd, sceCesSetStrContinue

Specify continuous character string processing

Definition

```
#include <ces.h>
int sceCesSetStrBegin(
    SceCesUcsContext *context
)
int sceCesSetStrLast(
    SceCesUcsContext *context
)
int sceCesSetStrEnd(
    SceCesUcsContext *context
)
int sceCesSetStrContinue(
    SceCesUcsContext *context
)
```

Arguments

context Address of the context used for character string processing

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid

Description

The functions described here set into the context information on whether the character string processing function using the specified context is made to start character string processing after initializing the context status according to the settings as usual, or if it is made to treat the character string as being continued from the previously processed character string, starting character string processing after inheriting the context status; also, they set information on whether there is another character string after the character string that is processed.

`sceCesSetStrBegin()` makes the next character string processing function treat the character string as the character string to be put first and leaves an instruction in the context to make the subsequently called character string processing function treat the character string to be processed as the character string continuing the previous one and preceding another one until `sceCesSetStrLast()` or `sceCesSetStrEnd()` is called.

`sceCesSetStrLast()` leaves to the character string processing function that is next called the information that the character string to process is the continuation of the previous one, that the character string is the last one and that it requires termination processing. The specified context is the context for which `sceCesSetStrBegin()` is called. Operation for other contexts is the same as for `sceCesSetStrContinue()`. After processing the last character string, the calling effect of `sceCesSetStrBegin()` will end.

`sceCesSetStrEnd()` ends the calling effect of `sceCesSetStrBegin()`. From the character string processing function that is next called, operation will return to normal processing that the character string is processed as the first character string and as a complete character string without any character string following it. It can also be used to as a method for cancelling `sceCesSetStrContinue()`.

`sceCesSetStrContinue()` should be called for contexts for which `sceCesSetStrBegin()` has not been called, or contexts for which the effect of specifying continuation has ended with `sceCesSetStrLast()`, etc. It will not be effective with other contexts. `sceCesSetStrContinue()` leaves an instruction for the character string processing function that is called next to treat the character string to be processed as the continuation of the previous one.

Be sure not to forget to either call `sceCesSetStrLast()` to process the character string, or call `sceCesSetStrEnd()` if `sceCesSetStrBegin()` is called.

Specify the address of the initialized context used to call the character string processing function to *context*.

Since the functions explained here are macro functions, it is possible to specify `SceCesMbcUcsContext` data type address in addition to `SceCesUcsContext`.

If `SCE_CES_ERROR_INVALID_PARAMETER` returns, it means that NULL pointer is passed to *context*.

Call these functions when it is required to restart the character string processing by inheriting the recognition status of the context.

If it is specified in the context that the character string to be processed next must follow the previously processed character string, it is the user's responsibility to guarantee that the character string actually processed is the character string continued from the previous one.

The specification of the functions explained here is mainly intended for CESs that the characters are determined sequentially. When attempting to restart character string processing with encoding that requires termination processing other than the NULL character, it is possible that termination processing may have already been performed with the output of the previous character string. In order to connect continuing character strings with this type of encoding, termination processing must be suppressed by specifying a continuing character string with `sceCesSetStrBegin()`.

Using the functions described here will not have any effect for an encoding whereby character recognition and output do not depend on the status of the context.

For CESs currently supported, to process a character string by inheriting the status of the context is meaningful only when the endianness recognition is dynamically processed with Unicode. By determining the endianness of the character string to be treated, the processing explained here will not be necessary.

Notes

This function is not multi-thread safe.

Context entity must be divided by thread for multithreading.

sceCesSetStrMaxCharCount

Specify the limit number of characters for character string processing

Definition

```
#include <ces.h>
int sceCesSetStrMaxCharCount (
    SceCesUcsContext *context,
    uint32_t srcCharMax,
    uint32_t dstCharMax
)
```

Arguments

<i>context</i>	Address of the context used for character string processing
<i>srcCharMax</i>	Limit number of input characters
<i>dstCharMax</i>	Limit number of output characters

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid

Description

This function sets the upper limit of the number of characters contained in a character string processed by a character string processing function.

Through this function, the character string processing function will return SCE_OK when characters up to the limit number are normally processed.

Specify the address of the initialized context used to call the character string processing function to *context*. Since the functions explained here are macro functions, it is possible to specify *SceCesMbcUcsContext* data type address in addition to *SceCesUcsContext*.

Set the number of characters that is allowed to be recognized to *srcCharMax*. Setting 0 means that no limit is specified.

Set the number of characters that is allowed to be output to *dstCharMax*. Setting 0 means that no limit is specified.

Since this settings will remain in effect until being changed, return the settings to the status for which no limit is specified with `sceCesSetStrMaxCharCount(context, 0, 0)` when the limitation becomes no longer needed.

The number of characters is specified in terms of the character codes, and thus, a special code such as BOM code of Unicode is also counted as one character. If it is required to limit the number of characters with BOM excluded, use `sceCesSetUcsPolicyOutputBom()` to limit the number of output characters by setting BOM to be excluded.

If SCE_CES_ERROR_INVALID_PARAMETER returns, it means that a NULL pointer has been passed to *context*.

SCE CONFIDENTIAL

Notes

This function is not multi-thread safe.

Context entity must be divided by thread for multithreading.

000004892117

Setting Functions for Error Handling

sceCesSetErrorOperation

Specify operation to handle character string conversion error

Definition

```
#include <ces.h>
int sceCesSetErrorOperation (
    SceCesUcsContext *context,
    int errCaseValue,
    int errOperation
)
```

Arguments

<i>context</i>	Address of the context used for character string processing
<i>errCaseValue</i>	Specifies the error type
<i>errOperation</i>	enum constant indicating the error handling method

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid

Description

This function specifies operations to the context to handle an error occurred when a character string processing function is performed.

Specify the address of the initialized context used to call the character string processing function to *context*. Since the functions explained here are macro functions, it is possible to specify *SceCesMbcUcsContext* data type address in addition to *SceCesUcsContext*.

Specify one of the following values indicating the error case to *errCaseValue*.

Value	Description
SCE_CES_ERROR_INVALID_ENCODE	Applied to the input character string encoding error
SCE_CES_ERROR_ILLEGAL_CODE	Applied to the input character code value error
SCE_CES_ERROR_OUT_OF_CODE_RANGE	Applied to the error arising from the detection of invalid code that is outside the range of output destination's CES
SCE_CES_ERROR_UNASSIGNED_CODE	Applied to the error occurred when the code point of output destination's encoding scheme is not defined

Specify one of the following values indicating the error handling method to *errOperation*.

Value	Description
SCE_CES_ERR_OPERATION_RETURN	No error operation is performed. Reports the error by returning the function. (Default value)
SCE_CES_ERR_OPERATION_SKIP	Outputs the result by skipping the error part
SCE_CES_ERR_OPERATION_REPLACE	Outputs the result by replacing the error to the replacement characters
SCE_CES_ERR_OPERATION_OK	Outputs the result by ignoring the error

Returning `SCE_CES_ERROR_INVALID_PARAMETER` means that it is judged that an error occurs because a `NULL` pointer has been specified to *context*, or invalid value has been specified to *errCase* or *errOperation*.

Examples

```
{
    sceCesUcsContext( &ctx );
    //Skip characters that cannot be expressed with the conversion narrowing
the code range
    sceCesSetErrorOperation( &ctx, SCE_CES_ERROR_OUT_OF_CODE_RANGE,
                             SCE_CES_ERR_OPERATION_SKIP );

    //UTF-16 -> UCS-2
    sceCesUtf16StrGetUcs2Len( &ctx,
                              utf16str, utf16max, &utf16Len, &ucs2Len );

    ucs2buf = malloc( ucs2Len + 1 );
    sceCesUtf16StrToUcs2Str( &ctx,
                             utf16str, utf16max, &utf16Len,
                             ucs2buf, ucs2Len+1, &ucs2Len );
}

{
    SceCesMbcUcsContext mctx;
    SceCesUcsProfileSheet sheet;
    const SceCesSJisUcsProfile* profile;
    profile = sceCesUcsProfileInitSJis( &sheet ); //S-JIS

    sceCesMbcUcsContextInit( &mctx, sceCesGetMbcUcsProfile(profile) );
    //Replace the characters having no conversion information to the
    //specific characters (Error character -> '_' )
    sceCesSetErrorOperation( &mctx, SCE_CES_ERROR_UNASSIGNED_CODE,
                             SCE_CES_ERR_OPERATION_REPLACE );
    sceCesSetReplacementCharUCode( &mctx, '_' );
    //UTF-16 -> MBCS
    sceCesUtf16StrGetMbcLen( &mctx,
                              utf16str, utf16max, &utf16Len, &mbcsLen );
    mbcsBuf = malloc( mbcsLen + 1 );
    sceCesUtf16StrToMbcStr( &mctx,
                             utf16str, utf16max, &utf16Len,
                             mbcsBuf, mbcsLen+1, &mbcsLen );
}

{
    sceCesUcsContext( &ctx );
    //Lower the error detection to read a specific encoding that is extended
    //for handling UTF-8, etc.
    sceCesSetErrorOperation( &ctx, SCE_CES_ERROR_ILLEGAL_CODE,
                             SCE_CES_ERR_OPERATION_OK );

    //Receive the code even if it is encoded to UTF-8 in five or six bytes
```

SCE CONFIDENTIAL

```
sceCesSetErrorOperation( &ctx, SCE_CES_ERROR_OUT_OF_CODE_RANGE,  
                        SCE_CES_ERR_OPERATION_OK );  
//UTF-8 -> UTF-32 (Corresponds to UCS-4)  
uint32_t utf32buf[BUFMAX];  
sceCesUtf8StrToUtf32Str( &ctx,  
                        &utf8str[0], 0, &utf8Len,  
                        &utf32buf[0], sizeof(utf32buf)/4, &utf32Len );  
}
```

Notes

This function is not multi-thread safe.

Context entity must be divided by thread for multithreading.

000004892117

SCE CONFIDENTIAL

sceCesUnsetErrorOperation

Cancel all operations for character string conversion error handling

Definition

```
#include <ces.h>
int sceCesUnsetErrorOperation(
    SceCesUcsContext *context
)
```

Arguments

context Address of the context used for character string processing

Return Values

Returns SCE_OK(0) as the value of the function for success.

Returns the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid

Description

This function cancels all the error operations set in the specified context with `sceCesSetErrorOperation()`. In other words, this function returns the settings to the default value set immediately after the context initialization, that is, to the settings with no error operation specified.

Specify the address of the initialized context used to call the character string processing function to *context*. Since the functions explained here are macro functions, it is possible to specify `SceCesMbcUcsContext` data type address in addition to `SceCesUcsContext`.

If SCE_CES_ERROR_INVALID_PARAMETER returns, it means that a NULL pointer has been specified to *context*.

Notes

This function is not multi-thread safe.

Context entity must be divided by thread for multithreading.

SCE CONFIDENTIAL

sceCesSetReplacementCharUnicode

Set replacement character code

Definition

```
#include <ces.h>
int sceCesSetReplacementCharUnicode (
    SceCesUcsContext *context,
    uint32_t ucode
)
```

Arguments

context Address of the context for character string conversion
ucode Replacement character code

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected

Description

This function specifies the character used as a replacement character in Unicode character.

When “character replacement” is specified as the error operation for a character string with `sceCesSetErrorOperation()`, the specified character will be used if an error occurs within a character string processing function.

Specify the address of the initialized context used to call the character string function to *context*.

Since the function explained here is a macro function, it is possible to specify `SceCesMbcUcsContext` data type address in addition to `SceCesUcsContext`.

Specify the character used as a replacement character in Unicode value to *ucode*.

If a character that cannot be used in character sets other than Unicode is specified, the character replacement will not be performed in CESs other than Unicode.

If SCE_CES_ERROR_INVALID_PARAMETER returns, it means that a NULL pointer has been passed to *context*.

If SCE_CES_ERROR_ILLEGAL_CODE returns, it means that an inappropriate value has been passed to *ucode*.

Notes

This function is not multi-thread safe.

Context entity must be divided by thread for multithreading.

sceCesSetUcsReplacementCharCode, sceCesSetMbcSReplacementCharCode, sceCesSetMbcSReplacementCharUCode

Set replacement character code (individual specification)

Definition

```
#include <ces.h>
int sceCesSetUcsReplacementCharCode (
    SceCesMbcSucsContext *context,
    uint32_t ucode
)
int sceCesSetMbcSReplacementCharCode (
    SceCesMbcSucsContext *context,
    uint32_t code
)
int sceCesSetMbcSReplacementCharUCode (
    SceCesMbcSucsContext *context,
    uint32_t ucode
)
```

Arguments

context Address of the context for MBCS character string conversion to be initialized
ucode, code Replacement character code

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal code detected

Description

The functions explained here are the functions that specify the character used as a replacement character for Unicode and for other character sets respectively.

`sceCesSetUcsReplacementCharCode()` specifies the replacement character used for a Unicode character string, and `sceCesSetMbcSReplacementCharCode()` specifies the code value (32-bit value) of the replacement character used for a multi-byte character string.

`sceCesSetMbcSReplacementCharUCode()` specifies the replacement character used for a multi-byte character string in Unicode character.

Both settings are overwritten with `sceCesSetReplacementCharUCode()`.

Specify the address of the initialized context used to call the character string function to *context*. These functions explained here are macro functions. For `sceCesSetUcsReplacementCharCode()`, It is possible to specify `SceCesUcsContext` data type address in addition to `SceCesMbcSucsContext` data type address.

Specify the character used as a replacement character in Unicode value to *ucode*.

SCE CONFIDENTIAL

For *code*, specify the character used as a replacement character in the code value according to the character sets. (For example, 1 byte character of '_' is specified as 0x5f, and 2-byte character of "\x81\x40" is specified as 0x8140.)

If `SCE_CES_ERROR_INVALID_PARAMETER` returns, it means that a NULL pointer has been passed to *context*.

If `SCE_CES_ERROR_ILLEGAL_CODE` returns, it means that an inappropriate value has been passed to *ucode*.

Notes

This function is not multi-thread safe.

Context entity must be divided by thread for multithreading.

000004892117

UCS Conversion Profiles of Single-Byte Character Sets

7-Bit Character Set UCS Conversion Profiles

sceCesRefersUcsProfileAscii

Reference the profile holding ASCII and UCS conversion information

Definition

```
#include <ces.h>
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileAscii( void )
```

Return Values

Address of the profile holding ASCII and UCS conversion information

Description

References the address of the profile holding ASCII and UCS conversion information.

Use the address obtained with the return value by passing it to functions requiring a profile as argument.

Notes

This function is multi-thread safe.

See Also

SceCesSbcsUcsProfile, sceCesMbcsUcsContextInit(), sceCesSbcToUtf32(),
 sceCesUtf32ToSbc(), sceCesSbcToUtf32be(), sceCesUtf32beToSbc(),
 sceCesSbcToUtf32le(), sceCesUtf32leToSbc(), sceCesSbcToUtf16(),
 sceCesUtf16ToSbc(), sceCesSbcToUtf16be(), sceCesUtf16leToSbc(),
 sceCesSbcToUtf16le(), sceCesUtf16beToSbc(), sceCesSbcToUtf8(),
 sceCesUtf8ToSbc(), sceCesSbcToUcs2(), sceCesUcs2ToSbc()

sceCesRefersUcsProfileJisX0201Roman, sceCesRefersUcsProfileJisX0201RomanTilde0x7e

Reference the profile holding JIS X 0201Roman and UCS conversion information

Definition

```
#include <ces.h>
const SceCesSbcsUcsProfile*
sceCesRefersUcsProfileJisX0201Roman(void)
const SceCesSbcsUcsProfile*
sceCesRefersUcsProfileJisX0201RomanTilde0x7e(void)
```

Return Values

Address of the profile holding JIS X 0201Roman and UCS conversion information

Address of the profile holding JIS X 0201Roman (Tilde :0x7e) and UCS conversion information

Description

References the address of the profile holding JIS X 0201Roman and UCS conversion information.

JIS X 0201Roman is a JIS X 0201 character set in the 7-bit range, and does not contain 8-bit *katakana*.

Characters with different mapping from ASCII are as follows:

`sceCesRefersUcsProfileJisX0201Roman()`

JIS X 0201	UCS	Character Description
0x5C	U+00A5	Yen sign
0x7e	U+203E	Overline

`sceCesRefersUcsProfileJisX0201RomanTilde0x7e()`

JIS X 0201	UCS	Character Description
0x5C	U+00A5	Yen sign

Use the address obtained with the return value by passing it to functions requiring a profile as argument.

Notes

This function is multi-thread safe.

See Also

`SceCesSbcsUcsProfile`, `sceCesMbcUcsContextInit()`, `sceCesSbcToUtf32()`,
`sceCesUtf32ToSbc()`, `sceCesSbcToUtf32be()`, `sceCesUtf32beToSbc()`,
`sceCesSbcToUtf32le()`, `sceCesUtf32leToSbc()`, `sceCesSbcToUtf16()`,
`sceCesUtf16ToSbc()`, `sceCesSbcToUtf16be()`, `sceCesUtf16leToSbc()`,
`sceCesSbcToUtf16le()`, `sceCesUtf16beToSbc()`, `sceCesSbcToUtf8()`,
`sceCesUtf8ToSbc()`, `sceCesSbcToUcs2()`, `sceCesUcs2ToSbc()`

sceCesRefersUcsProfileGbT1988, sceCesRefersUcsProfileGbT1988Tilde0x7e

Reference the profile holding GB/T 1988 and UCS conversion information

Definition

```
#include <ces.h>
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileGbT1988(void)
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileGbT1988Tilde0x7e(void)
```

Return Values

Address of the profile holding GB/T 1988 and UCS conversion information

Address of the profile holding GB/T 1988 (Tilde: 0x7e) and UCS conversion information

Description

References the address of the profile holding GB/T 1988 and UCS conversion information.

Characters with different mapping from ASCII are as follows:

`sceCesRefersUcsProfileGbT1988()`

GB/T 1988	UCS	Character Description
0x24	U+00A5	Yen sign
0x7e	U+203E	Overline

`sceCesRefersUcsProfileGbT1988Tilde0x7e()`

GB/T 1988	UCS	Character Description
0x24	U+00A5	Yen sign

Use the address obtained with the return value by passing it to functions requiring a profile as argument.

Notes

This function is multi-thread safe.

See Also

`SceCesSbcsUcsProfile`, `sceCesMbcUcsContextInit()`, `sceCesSbcToUtf32()`,
`sceCesUtf32ToSbc()`, `sceCesSbcToUtf32be()`, `sceCesUtf32beToSbc()`,
`sceCesSbcToUtf32le()`, `sceCesUtf32leToSbc()`, `sceCesSbcToUtf16()`,
`sceCesUtf16ToSbc()`, `sceCesSbcToUtf16be()`, `sceCesUtf16leToSbc()`,
`sceCesSbcToUtf16le()`, `sceCesUtf16beToSbc()`, `sceCesSbcToUtf8()`,
`sceCesUtf8ToSbc()`, `sceCesSbcToUcs2()`, `sceCesUcs2ToSbc()`

sceCesRefersUcsProfileKsX1003, sceCesRefersUcsProfileKsX1003Tilde0x7e

Reference the profile holding KS X 1003 and UCS conversion information

Definition

```
#include <ces.h>
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileKsX1003(void)
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileKsX1003Tilde0x7e(void)
```

Return Values

Address of the profile holding KS X 1003 and UCS conversion information

Address of the profile holding KS X 1003 (0x7e:Tilde) and UCS conversion information

Description

References the address of the profile holding KS X 1003 and UCS conversion information.

Characters with different mapping from ASCII are as follows:

sceCesRefersUcsProfileKsX1003(void)

KS X 1003	UCS	Character Description
0x5c	U+20A9	Won sign
0x7E	U+203E	Overline

sceCesRefersUcsProfileKsX1003Tilde0x7e(void)

KS X 1003	UCS	Character Description
0x5c	U+20A9	Won sign

Use the address obtained with the return value by passing it to functions requiring a profile as argument.

Notes

This function is multi-thread safe.

See Also

SceCesSbcsUcsProfile, **sceCesMbcSucsContextInit**(), **sceCesSbcToUtf32**(), **sceCesUtf32ToSbc**(), **sceCesSbcToUtf32be**(), **sceCesUtf32beToSbc**(), **sceCesSbcToUtf32le**(), **sceCesUtf32leToSbc**(), **sceCesSbcToUtf16**(), **sceCesUtf16ToSbc**(), **sceCesSbcToUtf16be**(), **sceCesUtf16leToSbc**(), **sceCesSbcToUtf16le**(), **sceCesUtf16beToSbc**(), **sceCesSbcToUtf8**(), **sceCesUtf8ToSbc**(), **sceCesSbcToUcs2**(), **sceCesUcs2ToSbc**()

8-Bit Character Set UCS Conversion Profiles

sceCesRefersUcsProfileIso8859_1,
sceCesRefersUcsProfileIso8859_2,
sceCesRefersUcsProfileIso8859_3,
sceCesRefersUcsProfileIso8859_4,
sceCesRefersUcsProfileIso8859_5,
sceCesRefersUcsProfileIso8859_6,
sceCesRefersUcsProfileIso8859_7,
sceCesRefersUcsProfileIso8859_8,
sceCesRefersUcsProfileIso8859_9,
sceCesRefersUcsProfileIso8859_10,
sceCesRefersUcsProfileIso8859_11,
sceCesRefersUcsProfileIso8859_13,
sceCesRefersUcsProfileIso8859_14,
sceCesRefersUcsProfileIso8859_15,
sceCesRefersUcsProfileIso8859_16

Reference the profile holding ISO/IEC 8859 and UCS conversion information

Definition

```

#include <ces.h>
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileIso8859_1( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileIso8859_2( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileIso8859_3( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileIso8859_4( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileIso8859_5( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileIso8859_6( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileIso8859_7( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileIso8859_8( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileIso8859_9( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileIso8859_10( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileIso8859_11( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileIso8859_13( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileIso8859_14( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileIso8859_15( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileIso8859_16( void )

```

Return Values

Address of the profile holding ISO/IEC 8859 and UCS conversion information

SCE CONFIDENTIAL

Description

References the address of the profile holding ISO/IEC 8859 and UCS conversion information.

Profile reference functions are available for ISO/IEC 8859-1 to 11, and 13 to 16. Select and use as necessary.

Use the address obtained with the return value by passing it to functions requiring a profile as argument.

Notes

This function is multi-thread safe.

See Also

`SceCesSbcsUcsProfile, sceCesMbcsUcsContextInit(), sceCesSbcToUtf32(),`
`sceCesUtf32ToSbc(), sceCesSbcToUtf32be(), sceCesUtf32beToSbc(),`
`sceCesSbcToUtf32le(), sceCesUtf32leToSbc(), sceCesSbcToUtf16(),`
`sceCesUtf16ToSbc(), sceCesSbcToUtf16be(), sceCesUtf16leToSbc(),`
`sceCesSbcToUtf16le(), sceCesUtf16beToSbc(), sceCesSbcToUtf8(),`
`sceCesUtf8ToSbc(), sceCesSbcToUcs2(), sceCesUcs2ToSbc()`

sceCesRefersUcsProfileJisX0201, sceCesRefersUcsProfileJisX0201Tilde0x7e, sceCesRefersUcsProfileAsciiWithKatakana

Reference the profile holding JIS X 0201 and UCS conversion information

Definition

```
#include <ces.h>
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileJisX0201( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileJisX0201Tilde0x7e( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileAsciiWithKatakana( void )
```

Return Values

Address of the profile holding conversion information of character sets including each type of JIS X 0201 character and UCS

Description

References the address of the profile holding JIS X 0201 and UCS conversion information. These functions work with the JIS X 0201 *katakana* implemented with an 8-bit extension.

Characters with different mapping from ASCII are as follows:

API	0x5c	0x7e	0xa1 to 0xdf
sceCesRefersUcsProfileJisX0201() In conformity with JIS X 0201	Yen sign (U+00A5)	OVERLINE (U+203E)	Half-width <i>katakana</i> (U+FF61 to U+FF9F)
sceCesRefersUcsProfileJisX0201Tilde0x7e() JIS X 0201 (OVERLINE is handled with TILDE)	Yen sign (U+00A5)	TILDE (U+007E)	
sceCesRefersUcsProfileAsciiWithKatakana() ASCII + JIS X 0201 <i>katakana</i>	REVERSE SOLIDUS (U+005C)	TILDE (U+007E)	

(Grey areas are the same as ASCII)

Use the address obtained with the return value by passing it to functions requiring a profile as argument.

Notes

This function is multi-thread safe.

See Also

SceCesSbcsUcsProfile, sceCesMbcUcsContextInit(), sceCesSbcToUtf32(),
sceCesUtf32ToSbc(), sceCesSbcToUtf32be(), sceCesUtf32beToSbc(),
sceCesSbcToUtf32le(), sceCesUtf32leToSbc(), sceCesSbcToUtf16(),
sceCesUtf16ToSbc(), sceCesSbcToUtf16be(), sceCesUtf16leToSbc(),
sceCesSbcToUtf16le(), sceCesUtf16beToSbc(), sceCesSbcToUtf8(),
sceCesUtf8ToSbc(), sceCesSbcToUcs2(), sceCesUcs2ToSbc()

sceCesRefersUcsProfileKoi8R, sceCesRefersUcsProfileKoi8U

Reference the profile holding KOI8 (R/U) and UCS conversion information

Definition

```
#include <ces.h>
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileKoi8R( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileKoi8U( void )
```

Return Values

Address of the profile holding KOI8-R and UCS conversion information

Address of the profile holding KOI8-U and UCS conversion information

Description

The functions described here reference the address of the profile holding KOI8 and UCS conversion information.

Profile reference functions are available for KOI8-R and KOI8-U. Select and use as necessary.

Use the address obtained with the return value by passing it to functions requiring a profile as argument.

Notes

This function is multi-thread safe.

See Also

SceCesSbcsUcsProfile, sceCesMbcSbcsUcsContextInit(), sceCesSbcToUtf32(),
sceCesUtf32ToSbc(), sceCesSbcToUtf32be(), sceCesUtf32beToSbc(),
sceCesSbcToUtf32le(), sceCesUtf32leToSbc(), sceCesSbcToUtf16(),
sceCesUtf16ToSbc(), sceCesSbcToUtf16be(), sceCesUtf16leToSbc(),
sceCesSbcToUtf16le(), sceCesUtf16beToSbc(), sceCesSbcToUtf8(),
sceCesUtf8ToSbc(), sceCesSbcToUcs2(), sceCesUcs2ToSbc()

**sceCesRefersUcsProfileCp1250,
 sceCesRefersUcsProfileCp1251,
 sceCesRefersUcsProfileCp1252,
 sceCesRefersUcsProfileCp1253,
 sceCesRefersUcsProfileCp1254,
 sceCesRefersUcsProfileCp1255,
 sceCesRefersUcsProfileCp1256,
 sceCesRefersUcsProfileCp1257,
 sceCesRefersUcsProfileCp1258**

Reference profiles holding code pages from 1250 to 1258 and UCS conversion information

Definition

```
#include <ces.h>
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileCp1250( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileCp1251( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileCp1252( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileCp1253( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileCp1254( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileCp1255( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileCp1256( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileCp1257( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileCp1258( void )
```

Return Values

Address of the profile holding code page 1250 and UCS conversion information
 Address of the profile holding code page 1251 and UCS conversion information
 Address of the profile holding code page 1252 and UCS conversion information
 Address of the profile holding code page 1253 and UCS conversion information
 Address of the profile holding code page 1254 and UCS conversion information
 Address of the profile holding code page 1255 and UCS conversion information
 Address of the profile holding code page 1256 and UCS conversion information
 Address of the profile holding code page 1257 and UCS conversion information
 Address of the profile holding code page 1258 and UCS conversion information

Description

The functions described here reference the address of the profiles holding conversion information for UCS and the Microsoft Windows code pages from 1250 to 1258 provided in the library.

Each code page has its specific profile reference function. Select and use as necessary.

Use the address obtained with the return value by passing it to functions requiring a profile as argument.

SCE CONFIDENTIAL

Notes

This function is multi-thread safe.

See Also

`SceCesSbcsUcsProfile`, `sceCesMbcsUcsContextInit()`, `sceCesSbcToUtf32()`,
`sceCesUtf32ToSbc()`, `sceCesSbcToUtf32be()`, `sceCesUtf32beToSbc()`,
`sceCesSbcToUtf32le()`, `sceCesUtf32leToSbc()`, `sceCesSbcToUtf16()`,
`sceCesUtf16ToSbc()`, `sceCesSbcToUtf16be()`, `sceCesUtf16leToSbc()`,
`sceCesSbcToUtf16le()`, `sceCesUtf16beToSbc()`, `sceCesSbcToUtf8()`,
`sceCesUtf8ToSbc()`, `sceCesSbcToUcs2()`, `sceCesUcs2ToSbc()`

000004892117

sceCesRefersUcsProfileCp437,
sceCesRefersUcsProfileCp737,
sceCesRefersUcsProfileCp775,
sceCesRefersUcsProfileCp850,
sceCesRefersUcsProfileCp852,
sceCesRefersUcsProfileCp855,
sceCesRefersUcsProfileCp857,
sceCesRefersUcsProfileCp858,
sceCesRefersUcsProfileCp860,
sceCesRefersUcsProfileCp861,
sceCesRefersUcsProfileCp862,
sceCesRefersUcsProfileCp863,
sceCesRefersUcsProfileCp864,
sceCesRefersUcsProfileCp865,
sceCesRefersUcsProfileCp866,
sceCesRefersUcsProfileCp869,
sceCesRefersUcsProfileCp874

Reference the profiles holding OEM code page and UCS conversion information

Definition

```

#include <ces.h>
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileCp437( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileCp737( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileCp775( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileCp850( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileCp852( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileCp855( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileCp857( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileCp858( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileCp860( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileCp861( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileCp862( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileCp863( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileCp864( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileCp865( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileCp866( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileCp869( void )
const SceCesSbcsUcsProfile* sceCesRefersUcsProfileCp874( void )

```

SCE CONFIDENTIAL

Return Values

Address of the profile holding code page 437 and UCS conversion information
 Address of the profile holding code page 737 and UCS conversion information
 Address of the profile holding code page 775 and UCS conversion information
 Address of the profile holding code page 850 and UCS conversion information
 Address of the profile holding code page 852 and UCS conversion information
 Address of the profile holding code page 855 and UCS conversion information
 Address of the profile holding code page 857 and UCS conversion information
 Address of the profile holding code page 858 and UCS conversion information
 Address of the profile holding code page 860 and UCS conversion information
 Address of the profile holding code page 861 and UCS conversion information
 Address of the profile holding code page 862 and UCS conversion information
 Address of the profile holding code page 863 and UCS conversion information
 Address of the profile holding code page 864 and UCS conversion information
 Address of the profile holding code page 865 and UCS conversion information
 Address of the profile holding code page 866 and UCS conversion information
 Address of the profile holding code page 869 and UCS conversion information
 Address of the profile holding code page 874 and UCS conversion information

Description

The functions described here reference the address of the “profile holding conversion information for UCS and the Microsoft OEM code pages” provided in the library.

The number following `sceCesRefersUcsProfileCp` in the name of the function indicates the code page number. Select the profile reference function corresponding to the number of the code page you wish to use.

Use the address obtained with the return value by passing it to functions requiring a profile as argument.

Notes

This function is multi-thread safe.

See Also

`SceCesSbcsUcsProfile`, `sceCesMbcSucsContextInit()`, `sceCesSbcToUtf32()`,
`sceCesUtf32ToSbc()`, `sceCesSbcToUtf32be()`, `sceCesUtf32beToSbc()`,
`sceCesSbcToUtf32le()`, `sceCesUtf32leToSbc()`, `sceCesSbcToUtf16()`,
`sceCesUtf16ToSbc()`, `sceCesSbcToUtf16be()`, `sceCesUtf16leToSbc()`,
`sceCesSbcToUtf16le()`, `sceCesUtf16beToSbc()`, `sceCesSbcToUtf8()`,
`sceCesUtf8ToSbc()`, `sceCesSbcToUcs2()`, `sceCesUcs2ToSbc()`

UCS Conversion Profiles of Multi-Byte Character Sets

UCS Conversion Profiles Shared by MBCS

SceCesUcsProfileSheet

Data type for holding conversion rules between Unicode and other character sets

Definition

```
#include <ces.h>
typedef struct SceCesUcsProfileSheet{
    /* omitted */
} SceCesUcsProfileSheet;
```

Description

This is a data type for holding conversion rules between Unicode and other character sets.

The entity of this data type is allocated by the user.

(This structure is necessary when handling multi-byte character sets. Conversion profile reference functions with names beginning by `sceCesRefersUcsProfile` are available in the library for UCS conversion profile of single-byte character sets; therefore, for these it is not necessary to use this structure, and there is no initialization function available.)

It is possible to receive this data type with the address of the UCS conversion profile type of each CES by initializing this data type with the UCS conversion profile initialization function for each CES (functions whose name begins by `sceCesUcsProfileInit`).

After initialization, while the data type is being referenced as UCS conversion profile for each CES, the memory in which the data type is stored must not be freed.

See Also

```
sceCesUcsProfileInitSJis(), sceCesUcsProfileInitBig5(),
sceCesUcsProfileInitGb18030(), sceCesUcsProfileInitUhc()
```

SceCesMbcUcsProfile, sceCesGetMbcUcsProfile

Data types for holding multi-byte character sets and Unicode conversion rules

Definition

```
#include <ces.h>
typedef struct SceCesMbcUcsProfile{
    /* omitted */
} SceCesMbcUcsProfile;

#define sceCesGetMbcUcsProfile( profile ) /* omitted */
```

Description

The `SceCesMbcUcsProfile` type is a data type for holding conversion rules between multi-byte character sets (including single-byte character sets) and Unicode.

It is possible to obtain the reference address with the return value of `sceCesGetMbcUcsProfile()`.

profile requires the address of the UCS conversion profiles for character sets of each type of CES.

Obtain the address of the UCS conversion profiles for character sets of each type of CES with the following method.

The type of the UCS conversion profile of single-byte character sets can be obtained as the return value of profile reference functions whose name begins by `sceCesRefersUcsProfile`.

The UCS conversion profile of multi-byte character sets can be obtained as the return value of profile initialization functions whose name begins by `sceCesUcsProfileInit`.

Examples

```
static SceCesUcsProfileSheet s_sjisSheet;
const SceCesMbcUcsProfile *mProf;
SceCesMbcUcsContext mctx;

mProf = sceCesGetMbcUcsProfile( sceCesUcsProfileInitSJis( &s_sjisSheet ) );
sceCesUcs2ToMbc( ucs2, mProf, &mbcBuf, mbcMax, &mbcLen );

mProf = sceCesGetMbcUcsProfile( sceCesRefersUcsProfileIso8859_15() );
sceCesMbcUcsContextInit( &mctx, mprof );
sceCesMbcStrGetUtf16Len( &mctx, mbcStr, mbcMax, &mbcsLen, &utf16Len );
utf16Max = utf16Len + 1;
utf16Str = malloc( utf16Max * sizeof(uint16_t) );
sceCesMbcStrToUtf16Str( &mctx, mbcStr, mbcMax, &mbcsLen,
                        utf16Str, utf16Max, &utf16Len );
```

See Also

`SceCesSbcsUcsProfile`, `SceCesSJisUcsProfile`, `SceCesEucJpUcsProfile`,
`SceCesBig5UcsProfile`, `SceCesGbUcsProfile`, `SceCesEucCnUcsProfile`,
`SceCesUhcUcsProfile`, `SceCesEucKrUcsProfile`

UCS Conversion Profile of Shift_JIS

SceCesSJisUcsProfile

Profile holding Shift_JIS character sets/UCS conversion information

Definition

```
#include <ces.h>
typedef struct SceCesSJisUcsProfile{
    /* omitted */
} SceCesSJisUcsProfile;
```

Description

SceCesSJisUcsProfile is a profile type holding Shift_JIS character sets/UCS conversion information.

The address of this data type can be retrieved with one of the functions whose name begins by sceCesUcsProfileInit, listed under “See Also”.

Reference pages are not provided for the following 1-character processing functions using this data type.

Refer to the description pages of the 1-character processing functions provided for SceCesMbcUcsProfile, which has the similar argument structure, except for receiving this data type address as the profile argument.

Function for SceCesSJisUcsProfile	Function for SceCesMbcUcsProfile
sceCesSJisToUtf32 ()	sceCesMbcToUtf32 ()
sceCesSJisToUtf32be ()	sceCesMbcToUtf32be ()
sceCesSJisToUtf32le ()	sceCesMbcToUtf32le ()
sceCesSJisToUtf16 ()	sceCesMbcToUtf16 ()
sceCesSJisToUtf16be ()	sceCesMbcToUtf16be ()
sceCesSJisToUtf16le ()	sceCesMbcToUtf16le ()
sceCesSJisToUtf8 ()	sceCesMbcToUtf8 ()
sceCesSJisToUcs2 ()	sceCesMbcToUcs2 ()
sceCesUtf32ToSJis ()	sceCesUtf32ToMbc ()
sceCesUtf32beToSJis ()	sceCesUtf32beToMbc ()
sceCesUtf32leToSJis ()	sceCesUtf32leToMbc ()
sceCesUtf16ToSJis ()	sceCesUtf16ToMbc ()
sceCesUtf16beToSJis ()	sceCesUtf16beToMbc ()
sceCesUtf16leToSJis ()	sceCesUtf16leToMbc ()
sceCesUtf8ToSJis ()	sceCesUtf8ToMbc ()
sceCesUcs2ToSJis ()	sceCesUcs2ToMbc ()

See Also

```
sceCesUcsProfileInitSJis1997X0208 (), sceCesUcsProfileInitSJis1997Cp932 (),
sceCesUcsProfileInitSJis2004X0213 (), sceCesUcsProfileInitSJis (),
sceCesGetMbcUcsProfile ()
```

SCE CONFIDENTIAL

sceCesUcsProfileInitSJis1997X0208

Initialize the profile holding Shift_JIS(JIS X 0208)/UCS conversion information

Definition

```
#include <ces.h>
SceCesSJisUcsProfile* sceCesUcsProfileInitSJis1997X0208(
    SceCesUcsProfileSheet* profSheet
)
```

Arguments

profSheet Address of the UCS profile sheet

Return Values

The address specified in *profSheet* returns as the *SceCesSJisUcsProfile** type.

Description

In *profSheet*, this function configures the profile holding "JIS X 0201 + JIS X 0208" character set/UCS conversion information, and returns its address as the *SceCesSJisUcsProfile* type.

Prepare the entity of the *SceCesUcsProfileSheet* type on the caller side and specify its address in *profSheet*.

The Shift_JIS/UCS conversion profile initialized with this function will use JIS X 0201 for single-byte characters. The code 0x5C will be used for the Yen sign and 0x7E will be treated as "overline", while it will not be possible to use "tilde" (U+007E,U+FF5E).

	UCS	Shift_JIS (JIS X 0201 + JIS X 0208)
Yen symbol	U+00A5	5c
Full-width Yen symbol	U+FFE5	818F
Overline	U+203E	7e
Full-width overline	U+FFE3	8150
Backslash	U+005C	815f
Full-width backslash	U+FF3C	----
Tilde	U+007E	----
Full-width tilde	U+FF5E	----

Mapping only supports JIS X 0208 for double-byte characters.

External character sets such as NEC special characters (characters such as ㉠ to ㉡, I to X, No, (株)) are not included.

Since, in general, customized mapping including external character sets is often used, problems may arise, such as failure to convert or to find the fonts for the converted characters.

Only use when you wish to handle JIS X 0201 + JIS X 0208 mapping.

Mapping including external character sets is provided with *sceCesUcsProfileInitSJis1997Cp932()*.

Notes

This function is not multi-thread safe.

Prepare profile entity by thread or perform exclusive control for multithreading.

©SCEI

SCE CONFIDENTIAL

See Also

SceCesUcsProfileSheet, SceCesSJisUcsProfile,
sceCesUcsProfileInitSJis1997Cp932 (), sceCesMbcUcsContextInit ()

000004892117

sceCesUcsProfileInitSJis1997Cp932, sceCesUcsProfileInitSJis

Initialize the profile holding Shift_JIS (cp932)/UCS conversion information

Definition

```
#include <ces.h>
SceCesSJisUcsProfile* sceCesUcsProfileInitSJis1997Cp932 (
    SceCesUcsProfileSheet* profSheet
)
#define sceCesUcsProfileInitSJis sceCesUcsProfileInitSJis1997Cp932
```

Arguments

profSheet Address of the UCS profile sheet

Return Values

The address specified in *profSheet* returns as the *SceCesSJisUcsProfile** type.

Description

In *profSheet*, the function described here configures the profile holding JIS (CP932)/UCS conversion information, and returns its address as the *SceCesSJisUcsProfile* type.

sceCesUcsProfileInitSJis1997Cp932() configures a profile allowing interconversion between Shift_JIS (corresponding to Microsoft code page 932) and Unicode.

Its abbreviated name is *sceCesUcsProfileInitSJis()*.

The character sets that this function can handle are ASCII + JIS X 0201 *katakana* + JIS X 0208: 1997 + external character sets (NEC special characters, NEC-selected IBM extended characters and IBM extended characters).

Prepare the entity of the *SceCesUcsProfileSheet* type on the caller side and specify its address in *profSheet*.

ASCII will be used instead of JIS X 0201 for 1-byte codes in Shift_JIS (CP932). The 0x5C code corresponds to backslash instead of '¥'. The 0x7E code corresponds to tilde instead of overline.

The Yen sign and the overline, which cannot be handled as half-width characters in Shift_JIS(CP932), will be handled as full-width characters in case of conversion following the order: UCS- > Shift_JIS -> UCS.

Use the following table for reference. The character codes in parenthesis in the table are those for which reversible conversion is not possible, and only conversion from UCS is supported.

	Shift_JIS (CP932)	UCS
Backslash	5c	U+005C
Full-width backslash	815F	U+FF3C
Tilde	7e	U+007E
Full-width tilde	----	U+FF5E
Yen sign	818f	U+FFE5(U+00A5)
Overline	8150	U+FFE3(U+203E)

Also, the JIS X 0208 and UCS character mapping present the following differences. At the time of conversion to Shift_JIS, these characters will be integrated with Unicode full-width character code.

JIS plane/row/cell		S-JIS	UCS (sceCesUcsProfile InitSJis1997Cp932 ())	UCS (sceCesUcsProfile InitSJis1997X0208 ())
1-1-32	↖	0x815F	FULLWIDTH REVERSE SOLIDUS U+FF3C (U+005C)	REVERSE SOLIDUS U+005C
1-1-81	¢	0x8191	FULLWIDHT CENT SIGN U+FFE0 (U+00A2)	CENT SIGN U+00A2
1-1-82	£	0x8192	FULLWIDHT POUND SIGN U+FFE1 (U+00A3)	POUND SIGN U+00A3
1-2-44	¬	0x91C A	FULLWIDTH NOT SIGN U+FFE2 (U+00AC)	NOT SIGN U+00AC

In addition, characters mapped to other characters with similar shapes will become the same character when converted to Shift_JIS.

JIS plane/row/cell		S-JIS	UCS (sceCesUcsProfile InitSJis1997Cp932 ())	UCS (sceCesUcsProfile InitSJis1997X0208 ())
1-1-29	—	0x815C	HORIZONTAL BAR U+2015 (U+2014)	EM DASH U+2014
1-1-33	~	0x8160	FULLWIDTH TILDE U+FF5E (U+301C)	CJK PUNCTUATION:WAVE DASH U+301C
1-1-34	//	0x8161	OPERATOR:PARALLEL TO U+2225 (U+2016)	DOUBLE VERTICAL LINE U+2016
1-1-61	—	0x817C	OPERATOR:MINUS SIGN U+FF0D (U+2212)	FULLWIDTH HYPHEN-MINUS U+2212

Characters overlapping within external character sets will be integrated into the same character when converted to Unicode, following this priority order: JIS X 0208, NEC special characters, NEC-selected IBM extended characters and IBM extended characters.

Notes

This function is not multi-thread safe.

Prepare profile entity by thread or perform exclusive control for multithreading.

See Also

SceCesUcsProfileSheet, SceCesSJisUcsProfile, sceCesMbcUcsContextInit ()

SCE CONFIDENTIAL

sceCesUcsProfileInitSJis2004X0213

Initialize the profile holding Shift_JIS (Shift_JIS2004)/UCS conversion information

Definition

```
#include <ces.h>
SceCesSJisUcsProfile* sceCesUcsProfileInitSJis2004X0213(
    SceCesUcsProfileSheet* profSheet
)
```

Arguments

profSheet Address of the UCS profile sheet

Return Values

The address specified in *profSheet* returns as the *SceCesSJisUcsProfile** type.

Description

In *profSheet*, this function configures the profile holding “ASCII + JIS X 0201 *katakana* + JIS X 0213:2004” character set/UCS conversion information, and returns its address as the *SceCesSJisUcsProfile* type.

Prepare the entity of the *SceCesUcsProfileSheet* type on the caller side and specify its address in *profSheet*.

This conversion profile is used when mapping in conformity with JIS X 0213:2004 mapping to UCS is necessary.

JIS X 0213 is a character set enabling the use of the so-called Level 3 and Level 4 *kanji*.

With the exception of the Σ character and characters overlapping with JIS X 0208, “NEC special characters” used as external characters in JIS X 0208 can be used without any problems because they are incorporated in JIS X 0213 with the same code points. Areas used for NEC-selected IBM special characters (extended *kanji*, etc.) cannot be used because other *kanjis* are defined in JIS X 0213.

Notes

This function is not multi-thread safe.

Prepare profile entity by thread or perform exclusive control for multithreading.

See Also

SceCesUcsProfileSheet, *SceCesSJisUcsProfile*, *sceCesMbcUcsContextInit()*

SCE CONFIDENTIAL

sceCesSJisUcsProfileSetSbcs

Set single-byte character sets handled by Shift_JIS

Definition

```
#include <ces.h>
int sceCesSJisUcsProfileSetSbcs (
    SceCesSJisUcsProfile* profile,
    const SceCesSbcsUcsProfile* sbcsUcsProf
)
```

Arguments

profile Address of UCS conversion profile of Shift_JIS
sbcsUcsProf Address of UCS conversion profile to be used for Shift_JIS single-byte character

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid

Description

This function sets the character set in the range represented with single-byte in Shift_JIS for the specified UCS conversion profile.

Specify the address of *SceCesSJisUcsProfile* obtained through initialization with an initialization function whose name begins by *sceCesUcsProfileInitSJis* in *profile*.

Specify the UCS conversion profile of the character set returned by one of the following functions in *sbcsUcsProf*:

UCS profile reference function of specifiable character sets	Mapping differences	Katakana
<i>sceCesRefersUcsProfileJisX0201()</i>	0x5c(Yen sign)	Yes
<i>sceCesRefersUcsProfileJisX0201Roman()</i>	0x7e(OVERLINE)	No
<i>sceCesRefersUcsProfileJisX0201Tilde0x7e()</i>	0x5c(Yen sign)	Yes
<i>sceCesRefersUcsProfileJisX0201RomanTilde0x7e()</i>	0x7e(TILDE)	No
<i>sceCesRefersUcsProfileAsciiWithKatakana()</i>	0x5c(BACK SOLIDUS)	Yes
<i>sceCesRefersUcsProfileAscii()</i>	0x7e(TILDE)	No

If SCE_CES_ERROR_INVALID_PROFILE returns, it means that a NULL pointer has been passed to *profile* or *sbcsUcsProf*, or that the profile that has been passed is not appropriate.

Notes

This function is not multi-thread safe.

Prepare profile entity by thread or perform exclusive control for multithreading.

SCE CONFIDENTIAL

See Also

SceCesSJisUcsProfile, SceCesSbcsUcsProfile,
sceCesUcsProfileInitSJis1997X0208(), sceCesUcsProfileInitSJis1997Cp932(),
sceCesUcsProfileInitSJis2004X0213(), sceCesUcsProfileInitSJis()

000004892117

UCS Conversion Profile of EUC-JP

SceCesEucJpUcsProfile

Profile holding EUC-JP character sets/UCS conversion information

Definition

```
#include <ces.h>
typedef struct SceCesEucJpUcsProfile{
    /* omitted */
} SceCesEucJpUcsProfile;
```

Description

This is a profile type holding EUC-JP character sets/UCS conversion information.

The address of this data type can be retrieved with one of the functions whose name begins by `sceCesUcsProfileInit`, listed under “See Also”.

Reference pages are not provided for the following 1-character processing functions using this data type.

Refer to the description pages of the 1-character processing functions provided for `SceCesMbcUcsProfile`, which has the similar argument structure, except for receiving this data type address as the profile argument.

Function for <code>SceCesEucJpUcsProfile</code>	Function for <code>SceCesMbcUcsProfile</code>
<code>sceCesEucJpToUtf32 ()</code>	<code>sceCesMbcToUtf32 ()</code>
<code>sceCesEucJpToUtf32be ()</code>	<code>sceCesMbcToUtf32be ()</code>
<code>sceCesEucJpToUtf32le ()</code>	<code>sceCesMbcToUtf32le ()</code>
<code>sceCesEucJpToUtf16 ()</code>	<code>sceCesMbcToUtf16 ()</code>
<code>sceCesEucJpToUtf16be ()</code>	<code>sceCesMbcToUtf16be ()</code>
<code>sceCesEucJpToUtf16le ()</code>	<code>sceCesMbcToUtf16le ()</code>
<code>sceCesEucJpToUtf8 ()</code>	<code>sceCesMbcToUtf8 ()</code>
<code>sceCesEucJpToUcs2 ()</code>	<code>sceCesMbcToUcs2 ()</code>
<code>sceCesUtf32ToEucJp ()</code>	<code>sceCesUtf32ToMbc ()</code>
<code>sceCesUtf32beToEucJp ()</code>	<code>sceCesUtf32beToMbc ()</code>
<code>sceCesUtf32leToEucJp ()</code>	<code>sceCesUtf32leToMbc ()</code>
<code>sceCesUtf16ToEucJp ()</code>	<code>sceCesUtf16ToMbc ()</code>
<code>sceCesUtf16beToEucJp ()</code>	<code>sceCesUtf16beToMbc ()</code>
<code>sceCesUtf16leToEucJp ()</code>	<code>sceCesUtf16leToMbc ()</code>
<code>sceCesUtf8ToEucJp ()</code>	<code>sceCesUtf8ToMbc ()</code>
<code>sceCesUcs2ToEucJp ()</code>	<code>sceCesUcs2ToMbc ()</code>

See Also

```
sceCesUcsProfileInitEucJp (), sceCesUcsProfileInitEucJpX0208 (),
sceCesUcsProfileInitEucJpX0208Ss2 (), sceCesUcsProfileInitEucJpX0208Ss2Ss3 (),
sceCesUcsProfileInitEucJpCp51932 (), sceCesUcsProfileInitEucJis2004 (),
sceCesGetMbcUcsProfile ()
```

SCE CONFIDENTIAL

sceCesUcsProfileInitEucJp, sceCesUcsProfileInitEucJpCp51932

Initialize the profile holding EUC-JP/UCS conversion information

Definition

```
#include <ces.h>
SceCesEucJpUcsProfile* sceCesUcsProfileInitEucJpCp51932 (
    SceCesUcsProfileSheet* profSheet
)
#define sceCesUcsProfileInitEucJp sceCesUcsProfileInitEucJpCp51932
```

Arguments

profSheet Conversion profile of JIS and UCS

Return Values

The address specified in *profSheet* returns as the `SceCesEucJpUcsProfile*` type.

Description

The function described here configures the profile holding EUC-JP (corresponding to Microsoft code page CP51932)/UCS conversion information in *profSheet*, and returns its address as the `SceCesEucJpUcsProfile` type.

`sceCesUcsProfileInitEucJpCp51932()` configures a profile that can handle the Microsoft code page CP51932 character set.

Its abbreviated name is `sceCesUcsProfileInitEucJp()`.

Supported character sets are "ASCII + JIS X 0201 *katakana* + JIS X 0208 + external character sets (NEC special characters + NEC-selected IBM extended *kanji*)".

As for control characters, the C1 character set (0x80 to 0x9f) is recognized in addition to the C0 character set (0x00 to 0x1f).

Prepare the entity of the `SceCesUcsProfileSheet` type on the caller side and specify its address in *profSheet*.

Notes

This function is not multi-thread safe.

Prepare profile entity by thread or perform exclusive control for multithreading.

See Also

`SceCesUcsProfileSheet`, `SceCesEucJpUcsProfile`, `sceCesMbcsUcsContextInit()`

SCE CONFIDENTIAL

sceCesUcsProfileInitEucJpX0208, sceCesUcsProfileInitEucJpX0208Ss2, sceCesUcsProfileInitEucJpX0208Ss2Ss3

Initialize the profile holding EUC-JP/UCS conversion information

Definition

```
#include <ces.h>
SceCesEucJpUcsProfile* sceCesUcsProfileInitEucJpX0208(
    SceCesUcsProfileSheet* profSheet
)
SceCesEucJpUcsProfile* sceCesUcsProfileInitEucJpX0208Ss2(
    SceCesUcsProfileSheet* profSheet
)
SceCesEucJpUcsProfile* sceCesUcsProfileInitEucJpX0208Ss2Ss3(
    SceCesUcsProfileSheet* profSheet
)
```

Arguments

profSheet Conversion profile of EUC-JP and UCS

Return Values

The address specified in *profSheet* returns as the *SceCesEucJpUcsProfile** type.

Description

The functions described here configure profile information that allows interconversion between EUC-JP and UCS, and returns its address as the *SceCesEucJpUcsProfile* type.

sceCesUcsProfileInitEucJpX0208() can handle ASCII and JIS X 0208 characters.

In addition to the above, *sceCesUcsProfileInitEucJpX0208Ss2()* also allows use of JIS X 0201 *katakana*. *sceCesUcsProfileInitEucJpX0208Ss2Ss3()* further allows the use of JIS X 0212 (supplementary *kanji* set).

As for control characters, the C1 character set (0x80 to 0x9f) is recognized in addition to the C0 character set (0x00 to 0x1f).

Prepare the entity of the *SceCesUcsProfileSheet* type on the caller side and specify its address in *profSheet*.

Notes

This function is not multi-thread safe.

Prepare profile entity by thread or perform exclusive control for multithreading.

See Also

SceCesUcsProfileSheet, *SceCesEucJpUcsProfile*, *sceCesMbcsUcsContextInit()*

SCE CONFIDENTIAL

sceCesUcsProfileInitEucJis2004

Initialize the profile holding EUC-JIS-2004/UCS conversion information

Definition

```
#include <ces.h>
SceCesEucJpUcsProfile* sceCesUcsProfileInitEucJis2004(
    SceCesUcsProfileSheet* profSheet
)
```

Arguments

profSheet Conversion profile of EUC-JP and UCS

Return Values

The address specified in *profSheet* returns as the *SceCesEucJpUcsProfile** type.

Description

The function described here configures profile information that allows interconversion between EUC-JIS-2004 and UCS, and returns its address as the *SceCesEucJpUcsProfile* type.

Supported character sets are "ASCII + JIS X 0201 *katakana* + JIS X 0213:2004 + JIS X 0212".

As for control characters, the C1 character set (0x80 to 0x9f) is recognized in addition to the C0 character set (0x00 to 0x1f).

Prepare the entity of the *SceCesUcsProfileSheet* type on the caller side and specify its address in *profSheet*.

Notes

This function is not multi-thread safe.

Prepare profile entity by thread or perform exclusive control for multithreading.

See Also

SceCesUcsProfileSheet, *SceCesEucJpUcsProfile*, *sceCesMbcsUcsContextInit* ()

UCS Conversion Profile of Big5

SceCesBig5UcsProfile

Profile holding Big5 character sets/UCS conversion information

Definition

```
#include <ces.h>
typedef struct SceCesBig5UcsProfile{
    /* omitted */
} SceCesBig5UcsProfile;
```

Description

This is the profile type holding Big5 character sets/UCS conversion information.

The address of this data type can be retrieved with one of the functions whose name begins by `sceCesUcsProfileInit`, listed under “See Also”.

Reference pages are not provided for the following 1-character processing functions using this data type.

Refer to the description pages of the 1-character processing functions provided for `SceCesMbcUcsProfile`, which has the similar argument structure, except for receiving this data type address as the profile argument.

Function for <code>SceCesBig5UcsProfile</code>	Function for <code>SceCesMbcUcsProfile</code>
<code>sceCesBig5ToUtf32 ()</code>	<code>sceCesMbcToUtf32 ()</code>
<code>sceCesBig5ToUtf32be ()</code>	<code>sceCesMbcToUtf32be ()</code>
<code>sceCesBig5ToUtf32le ()</code>	<code>sceCesMbcToUtf32le ()</code>
<code>sceCesBig5ToUtf16 ()</code>	<code>sceCesMbcToUtf16 ()</code>
<code>sceCesBig5ToUtf16be ()</code>	<code>sceCesMbcToUtf16be ()</code>
<code>sceCesBig5ToUtf16le ()</code>	<code>sceCesMbcToUtf16le ()</code>
<code>sceCesBig5ToUtf8 ()</code>	<code>sceCesMbcToUtf8 ()</code>
<code>sceCesBig5ToUcs2 ()</code>	<code>sceCesMbcToUcs2 ()</code>
<code>sceCesUtf32ToBig5 ()</code>	<code>sceCesUtf32ToMbc ()</code>
<code>sceCesUtf32beToBig5 ()</code>	<code>sceCesUtf32beToMbc ()</code>
<code>sceCesUtf32leToBig5 ()</code>	<code>sceCesUtf32leToMbc ()</code>
<code>sceCesUtf16ToBig5 ()</code>	<code>sceCesUtf16ToMbc ()</code>
<code>sceCesUtf16beToBig5 ()</code>	<code>sceCesUtf16beToMbc ()</code>
<code>sceCesUtf16leToBig5 ()</code>	<code>sceCesUtf16leToMbc ()</code>
<code>sceCesUtf8ToBig5 ()</code>	<code>sceCesUtf8ToMbc ()</code>
<code>sceCesUcs2ToBig5 ()</code>	<code>sceCesUcs2ToMbc ()</code>

See Also

```
sceCesUcsProfileInitBig5 (), sceCesUcsProfileInitBig5Cp950 (),
sceCesGetMbcUcsProfile ()
```

sceCesUcsProfileInitBig5, sceCesUcsProfileInitBig5Cp950

Initialize the profile holding Big5/UCS conversion information

Definition

```
#include <ces.h>
SceCesBig5UcsProfile* sceCesUcsProfileInitBig5Cp950
    SceCesUcsProfileSheet* profSheet
)
#define sceCesUcsProfileInitBig5 sceCesUcsProfileInitBig5Cp950
```

Arguments

profSheet Address of the UCS profile sheet

Return Values

The address specified in *profSheet* returns as the *SceCesBig5UcsProfile** type.

Description

In *profSheet*, the function described here configures the profile holding Big5 (CP950 base)/UCS conversion information, and returns its address as the *SceCesBig5UcsProfile* type.

The profile returned by *sceCesUcsProfileInitBig5Cp950()* holds mapping corresponding to CP950. As for character sets, this function can handle less frequently used characters in addition to Big5-1984, but it does not work with the Hong Kong Supplementary Character Set.

sceCesUcsProfileInitBig5() is the abbreviated notation of *sceCesUcsProfileInitBig5Cp950()*.

Prepare the entity of the *SceCesUcsProfileSheet* type on the caller side and specify its address in *profSheet*.

Single-byte character set characters are handled as ASCII.

Notes

This function is not multi-thread safe.

Prepare profile entity by thread or perform exclusive control for multithreading.

See Also

SceCesUcsProfileSheet, *SceCesBig5UcsProfile*, *sceCesMbcsUcsContextInit()*

UCS Conversion Profile of GB

SceCesGbUcsProfile

Profile holding GB character sets/UCS conversion information

Definition

```
#include <ces.h>
typedef struct SceCesGbUcsProfile{
    /* omitted */
} SceCesGbUcsProfile;
```

Description

This is the profile type holding GB character sets/UCS conversion information.

The address of this data type can be retrieved with one of the functions whose name begins by `sceCesUcsProfileInit`.

Reference pages are not provided for the following 1-character processing functions using this data type.

Refer to the description pages of the 1-character processing functions provided for `SceCesMbcUcsProfile`, which has the similar argument structure, except for receiving this data type address as the profile argument.

Function for <code>SceCesGbUcsProfile</code>	Function for <code>SceCesMbcUcsProfile</code>
<code>sceCesGbToUtf32()</code>	<code>sceCesMbcToUtf32()</code>
<code>sceCesGbToUtf32be()</code>	<code>sceCesMbcToUtf32be()</code>
<code>sceCesGbToUtf32le()</code>	<code>sceCesMbcToUtf32le()</code>
<code>sceCesGbToUtf16()</code>	<code>sceCesMbcToUtf16()</code>
<code>sceCesGbToUtf16be()</code>	<code>sceCesMbcToUtf16be()</code>
<code>sceCesGbToUtf16le()</code>	<code>sceCesMbcToUtf16le()</code>
<code>sceCesGbToUtf8()</code>	<code>sceCesMbcToUtf8()</code>
<code>sceCesGbToUcs2()</code>	<code>sceCesMbcToUcs2()</code>
<code>sceCesUtf32ToGb()</code>	<code>sceCesUtf32ToMbc()</code>
<code>sceCesUtf32beToGb()</code>	<code>sceCesUtf32beToMbc()</code>
<code>sceCesUtf32leToGb()</code>	<code>sceCesUtf32leToMbc()</code>
<code>sceCesUtf16ToGb()</code>	<code>sceCesUtf16ToMbc()</code>
<code>sceCesUtf16beToGb()</code>	<code>sceCesUtf16beToMbc()</code>
<code>sceCesUtf16leToGb()</code>	<code>sceCesUtf16leToMbc()</code>
<code>sceCesUtf8ToGb()</code>	<code>sceCesUtf8ToMbc()</code>
<code>sceCesUcs2ToGb()</code>	<code>sceCesUcs2ToMbc()</code>

See Also

```
sceCesUcsProfileInitGbk(), sceCesUcsProfileInitGb18030(),
sceCesUcsProfileInitEucCnGb2312(), sceCesUcsProfileInitGbkCp936(),
sceCesUcsProfileInitGb18030(), sceCesUcsProfileInitGb18030_2000(),
sceCesGetMbcUcsProfile()
```

sceCesUcsProfileInitGb18030, sceCesUcsProfileInitGb18030_2000

Initialize the profile holding GB18030/UCS conversion information

Definition

```
#include <ces.h>
SceCesGbUcsProfile* sceCesUcsProfileInitGb18030_2000(
    SceCesUcsProfileSheet* profSheet
)
#define sceCesUcsProfileInitGb18030 sceCesUcsProfileInitGb18030_2000
```

Arguments

profSheet Address of the UCS profile sheet

Return Values

The address specified in *profSheet* returns.

Description

In *profSheet*, this function configures the profile holding GB 18030 character set/UCS conversion information, and returns its address as the *SceCesGbUcsProfile* type.

sceCesUcsProfileInitGb18030_2000() configures a profile that can handle GB18030:2000 mapping.

sceCesUcsProfileInitGb18030() is the abbreviated notation (without the year) of *sceCesUcsProfileInitGb18030_2000()*.

Prepare the entity of the *SceCesUcsProfileSheet* type on the caller side and specify its address in *profSheet*.

ASCII is used as single-byte character set. (It is not possible to use the single-byte Euro currency sign (0x80) that can be used with *sceCesUcsProfileInitGb18030_2000()*. The Euro currency sign is defined as a 2-byte character with 0xA2 and 0xE3.)

Notes

This function is not multi-thread safe.

Prepare profile entity by thread or perform exclusive control for multithreading.

See Also

SceCesUcsProfileSheet, *SceCesGbUcsProfile*, *sceCesMbcsUcsContextInit()*

SCE CONFIDENTIAL

sceCesUcsProfileInitGb18030_2005

Initialize the profile holding GB 18030: 2005/UCS conversion information

Definition

```
#include <ces.h>
SceCesGbUcsProfile* sceCesUcsProfileInitGb18030_2005(
    SceCesUcsProfileSheet* profSheet
)
```

Arguments

profSheet Address of the UCS profile sheet

Return Values

The address specified in *profSheet* returns.

Description

In *profSheet*, this function configures the profile holding GB 18030: 2005 character set/UCS conversion information, and returns its address as the *SceCesGbUcsProfile* type.

Prepare the entity of the *SceCesUcsProfileSheet* type on the caller side and specify its address in *profSheet*.

ASCII is used as single-byte character set.

For GB 18030:2005, mapping to Unicode changes as described below.

GB code value	sceCesUcsProfileInitGb18030_2000	sceCesUcsProfileInitGb18030_2005
A8BC	Private area U+E7C7	LATIN SMALL LETTER M WITH ACUTE U+1E3F
8135F437	LATIN SMALL LETTER M WITH ACUTE U+1E3F	Private area U+E7C7

The 4-byte character of CJK Unified Ideographs Extension B (U+20000 to U+2FFFF) of UCS Supplementary Ideographic Plane in addition to UCSBMP Plane can be handled.

Notes

This function is not multi-thread safe.

Prepare profile entity by thread or perform exclusive control for multithreading.

See Also

SceCesUcsProfileSheet, *SceCesGbUcsProfile*, *sceCesMbcUcsContextInit()*,
sceCesUcsProfileInitGb18030_2000()

SCEI CONFIDENTIAL

sceCesUcsProfileInitGbk, sceCesUcsProfileInitGbkCp936

Initialize the profile holding GBK/UCS conversion information

Definition

```
#include <ces.h>
SceCesGbUcsProfile* sceCesUcsProfileInitGbkCp936 (
    SceCesUcsProfileSheet* profSheet
)
#define sceCesUcsProfileInitGbk sceCesUcsProfileInitGbkCp936
```

Arguments

profSheet Address of the UCS profile sheet

Return Values

The address specified in *profSheet* returns.

Description

In *profSheet*, this function configures the profile holding GBK (cp936)/UCS conversion information, and returns its address as the *SceCesGbUcsProfile* type.

sceCesUcsProfileInitGbkCp936() configures a profile that can handle mapping corresponding to code page 936.

Its abbreviated name is *sceCesUcsProfileInitGbk()*.

Prepare the entity of the *SceCesUcsProfileSheet* type on the caller side and specify its address in *profSheet*.

The address specified in *profSheet* will return as the return value.

ASCII is used as single-byte character set. In addition, 0x80 can be handled as the Euro currency sign (U+20AC).

It is compatible with other GB standards, except for 0x80 being handled as the Euro currency sign (U+20AC).

Also, the GBK characters that were not defined in ISO/IEC 10646-1:1993 will not be mapped to Unicode.

Notes

This function is not multi-thread safe.

Prepare profile entity by thread or perform exclusive control for multithreading.

See Also

SceCesUcsProfileSheet, *SceCesGbUcsProfile*, *sceCesMbcsUcsContextInit()*

SCE CONFIDENTIAL

sceCesGbUcsProfileSetSbcs

Set single-byte character sets handled by GB code

Definition

```
#include <ces.h>
int sceCesGbUcsProfileSetSbcs (
    SceCesGbUcsProfile* profile,
    const SceCesSbcsUcsProfile* sbcsUcsProf
)
```

Arguments

profile Address of UCS conversion profile of GB
sbcsUcsProf Address of UCS conversion profile to be used for GB single-byte character

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid

Description

This function sets the character set in the range represented in 7-bit (0x00-0x7f) in GB code for the specified UCS conversion profile.

Specify the address of *SceCesGbUcsProfile* obtained through initialization with an initialization function whose name begins by *sceCesUcsProfileInitGb* in *profile*.

Specify the UCS conversion profile of the character set returned by one of the following functions in *sbcsUcsProf*:

UCS profile reference function of specifiable character sets	Mapping differences
<i>sceCesRefersUcsProfileGbT1988</i> ()	0x24 (Yen sign) 0x7e(OVERLINE)
<i>sceCesRefersUcsProfileGbT1988Tilde0x7e</i> ()	0x24 (Yen sign) 0x7e(TILDE)
<i>sceCesRefersUcsProfileAscii</i> ()	0x24 (Dollar sign) 0x7e(TILDE)

If SCE_CES_ERROR_INVALID_PROFILE returns, it means that a NULL pointer has been passed to *profile* or *sbcsUcsProf*, or that the profile that has been passed is not appropriate.

Notes

This function is not multi-thread safe.

Prepare profile entity by thread or perform exclusive control for multithreading.

See Also

SceCesGbUcsProfile, *SceCesSbcsUcsProfile*, *sceCesUcsProfileInitGb18030* (),
sceCesUcsProfileInitGb18030_2005 (), *sceCesUcsProfileInitGbK* ()

©SCEI

SCE CONFIDENTIAL

sceCesGbUcsProfileSetUdaMapping

Set mapping of GB user defined area

Definition

```
#include <ces.h>
int sceCesGbUcsProfileSetUdaMapping (
    SceCesGbUcsProfile* profile,
    int udaMapping
)
```

Arguments

profile Address of UCS conversion profile of GB
udaMapping Value to specify the mapping method

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid

Description

This function specifies the mapping of user defined area in the double-character row of GB18030.

For *profile*, specify the address of the profile to which you wish to reflect the settings.

Specify one of the following values in *udaMapping*.

Value	Description
SCE_CES_GB_UDA_MAPPING_NONE	Does not perform mapping of user defined area
SCE_CES_GB_UDA_MAPPING_UCS_PUA	Enables the mapping to Unicode private area

IF SCE_CES_ERROR_INVALID_PROFILE returns, it means that a NULL pointer or inappropriate profile has been passed to *profile*.

IF SCE_CES_ERROR_INVALID_PARAMETER returns, it means that the value that has been specified to *udaMapping* is not valid.

Notes

This function is not multi-thread safe.

Prepare profile entity by thread or perform exclusive control for multithreading.

See Also

SceCesGbUcsProfile

SCE CONFIDENTIAL

sceCesGbUcsProfileSet4ByteCharRange

Set the range handling 4-byte characters of GB 18030

Definition

```
#include <ces.h>
int sceCesGbUcsProfileSet4ByteCharRange (
    SceCesGbUcsProfile* profile,
    int gb4byteRange
)
```

Arguments

profile Address of UCS conversion profile of GB
gb4byteRange Value indicating the character range which handles 4-byte characters of GB

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid

Description

This function sets the Unicode character range mapped by 4-byte characters included in GB18030.

For *profile*, specify the address of the GB18030 profile to which you wish to reflect the settings.

Specify one of the following values in *gb4byteRange*.

Value	Description
SCE_CES_GB_4BYTE_UCS_CJK_UIE_A	CJK Unified Ideographs Extension A
SCE_CES_GB_4BYTE_UCS_CJK_UIE_AB	CJK Unified Ideographs Extension A, CJK Unified Ideographs Extension B
SCE_CES_GB_4BYTE_UCS_BMP	UCS Basic Multilingual Plane (including CJK Unified Ideographs Extension A)
SCE_CES_GB_4BYTE_UCS_BMP_SIP	UCS Basic Multilingual Plane + UCS Supplementary Ideographic Plane (including CJK Unified Ideographs Extension B)
SCE_CES_GB_4BYTE_UCS_CODE_RANGE	All the Unicode range

For SCE_CES_GB_4BYTE_UCS_CJK_UIE_A and SCE_CES_GB_4BYTE_UCS_CJK_UIE_AB, the code point to which no character is defined is not included in the map range.

The default value of the profile initialized with `sceCesUcsProfileInitGb18030_2000()` is SCE_CES_GB_4BYTE_UCS_BMP.

The default value of the profile initialized with `sceCesUcsProfileInitGb18030_2005()` is SCE_CES_GB_4BYTE_UCS_BMP_SIP and only the value described in subsequent lines can be set.

If SCE_CES_ERROR_INVALID_PROFILE returns, it means that a NULL pointer has been passed to *profile*, or that the profile that has been passed is not appropriate.

If SCE_CES_ERROR_INVALID_PARAMETER returns, it means that the value that has been specified to *gb4byteRange* is not valid.

SCE CONFIDENTIAL

Notes

This function is not multi-thread safe.

Prepare profile entity by thread or perform exclusive control for multithreading.

See Also

SceCesGbUcsProfile

000004892117

UCS Conversion Profile of EUC-CN

SceCesEucCnUcsProfile

Profile holding EUC-CN character sets/UCS conversion information

Definition

```
#include <ces.h>
typedef struct SceCesEucCnUcsProfile{
    /* omitted */
} SceCesEucCnUcsProfile;
```

Description

This is the *profile* type holding character sets used by EUC-CN/UCS conversion information. The address of this data type can be retrieved with one of the functions whose name begins by `sceCesUcsProfileInit`, listed under “See Also”.

Reference pages are not provided for the following 1-character processing functions using this data type.

Refer to the description pages of the 1-character processing functions provided for `SceCesMbcUcsProfile`, which has the similar argument structure, except for receiving this data type address as the profile argument.

Function for <code>SceCesEucCnUcsProfile</code>	Function for <code>SceCesMbcUcsProfile</code>
<code>sceCesEucCnToUtf32()</code>	<code>sceCesMbcToUtf32()</code>
<code>sceCesEucCnToUtf32be()</code>	<code>sceCesMbcToUtf32be()</code>
<code>sceCesEucCnToUtf32le()</code>	<code>sceCesMbcToUtf32le()</code>
<code>sceCesEucCnToUtf16()</code>	<code>sceCesMbcToUtf16()</code>
<code>sceCesEucCnToUtf16be()</code>	<code>sceCesMbcToUtf16be()</code>
<code>sceCesEucCnToUtf16le()</code>	<code>sceCesMbcToUtf16le()</code>
<code>sceCesEucCnToUtf8()</code>	<code>sceCesMbcToUtf8()</code>
<code>sceCesEucCnToUcs2()</code>	<code>sceCesMbcToUcs2()</code>
<code>sceCesUtf32ToEucCn()</code>	<code>sceCesUtf32ToMbc()</code>
<code>sceCesUtf32beToEucCn()</code>	<code>sceCesUtf32beToMbc()</code>
<code>sceCesUtf32leToEucCn()</code>	<code>sceCesUtf32leToMbc()</code>
<code>sceCesUtf16ToEucCn()</code>	<code>sceCesUtf16ToMbc()</code>
<code>sceCesUtf16beToEucCn()</code>	<code>sceCesUtf16beToMbc()</code>
<code>sceCesUtf16leToEucCn()</code>	<code>sceCesUtf16leToMbc()</code>
<code>sceCesUtf8ToEucCn()</code>	<code>sceCesUtf8ToMbc()</code>
<code>sceCesUcs2ToEucCn()</code>	<code>sceCesUcs2ToMbc()</code>

See Also

```
sceCesUcsProfileInitEucCn(), sceCesUcsProfileInitEucCnGb2312(),
sceCesGetMbcUcsProfile()
```

sceCesUcsProfileInitEucCn, sceCesUcsProfileInitEucCnGb2312

Initialize the profile holding EUC-CN/UCS conversion information

Definition

```
#include <ces.h>
SceCesEucCnUcsProfile* sceCesUcsProfileInitEucCnGb2312 (
    SceCesUcsProfileSheet* profSheet
)
#define sceCesUcsProfileInitEucCn sceCesUcsProfileInitEucCnGb2312
```

Arguments

profSheet Address of the UCS profile sheet

Return Values

The address specified in *profSheet* returns.

Description

In *profSheet*, the functions described here configure the profile holding EUC-CN/UCS conversion information, and returns its address as the *SceCesEucCnUcsProfile* type.

sceCesUcsProfileInitEucCnGb2312 () configures a profile that can handle EUC-CN mapping. Its abbreviated name is *sceCesUcsProfileInitEucCn* ().

Prepare the entity of the *SceCesUcsProfileSheet* type on the caller side and specify its address in *profSheet*.

ASCII is used as single-byte character set and GB2312 is used as double-byte character set.

As for control characters, the C1 character set (0x80 to 0x9f) is recognized in addition to the C0 character set (0x00 to 0x1f).

Notes

This function is not multi-thread safe.

Prepare profile entity by thread or perform exclusive control for multithreading.

See Also

SceCesUcsProfileSheet, *SceCesEucCnUcsProfile*, *sceCesMbcsUcsContextInit* ()

UCS Conversion Profile of UHC(Unified Hangul Code)

SceCesUhcUcsProfile

Profile holding UHC/UCS conversion information

Definition

```
#include <ces.h>
typedef struct SceCesUhcUcsProfile{
    /* omitted */
} SceCesUhcUcsProfile;
```

Description

This is the profile type holding UHC (Unified Hangul Code) character sets/UCS conversion information.

The address of this data type can be retrieved with one of the functions whose name begins by `sceCesUcsProfileInit`, listed under “See Also”.

Reference pages are not provided for the following 1-character processing functions using this data type.

Refer to the description pages of the 1-character processing functions provided for `SceCesMbcUcsProfile`, which has the similar argument structure, except for receiving this data type address as the profile argument.

Function for <code>SceCesUhcUcsProfile</code>	Function for <code>SceCesMbcUcsProfile</code>
<code>sceCesUhcToUtf32 ()</code>	<code>sceCesMbcToUtf32 ()</code>
<code>sceCesUhcToUtf32be ()</code>	<code>sceCesMbcToUtf32be ()</code>
<code>sceCesUhcToUtf32le ()</code>	<code>sceCesMbcToUtf32le ()</code>
<code>sceCesUhcToUtf16 ()</code>	<code>sceCesMbcToUtf16 ()</code>
<code>sceCesUhcToUtf16be ()</code>	<code>sceCesMbcToUtf16be ()</code>
<code>sceCesUhcToUtf16le ()</code>	<code>sceCesMbcToUtf16le ()</code>
<code>sceCesUhcToUtf8 ()</code>	<code>sceCesMbcToUtf8 ()</code>
<code>sceCesUhcToUcs2 ()</code>	<code>sceCesMbcToUcs2 ()</code>
<code>sceCesUtf32ToUhc ()</code>	<code>sceCesUtf32ToMbc ()</code>
<code>sceCesUtf32beToUhc ()</code>	<code>sceCesUtf32beToMbc ()</code>
<code>sceCesUtf32leToUhc ()</code>	<code>sceCesUtf32leToMbc ()</code>
<code>sceCesUtf16ToUhc ()</code>	<code>sceCesUtf16ToMbc ()</code>
<code>sceCesUtf16beToUhc ()</code>	<code>sceCesUtf16beToMbc ()</code>
<code>sceCesUtf16leToUhc ()</code>	<code>sceCesUtf16leToMbc ()</code>
<code>sceCesUtf8ToUhc ()</code>	<code>sceCesUtf8ToMbc ()</code>
<code>sceCesUcs2ToUhc ()</code>	<code>sceCesUcs2ToMbc ()</code>

See Also

```
sceCesUcsProfileInitEucKr (), sceCesUcsProfileInitUhc (),
sceCesGetMbcUcsProfile ()
```

SCE CONFIDENTIAL

sceCesUcsProfileInitUhc

Initialize the profile holding UHC/UCS conversion information

Definition

```
#include <ces.h>
SceCesUhcUcsProfile* sceCesUcsProfileInitUhc
    SceCesUcsProfileSheet* profSheet
)
```

Arguments

profSheet Address of the UCS profile sheet

Return Values

The address specified in *profSheet* returns as the *SceCesUhcUcsProfile** type.

Description

In *profSheet*, this function configures the profile holding UHC (Unified Hangul Code) character set/UCS conversion information, and returns its address as the *SceCesUhcUcsProfile* type.

Prepare the entity of the *SceCesUcsProfileSheet* type on the caller side and specify its address in *profSheet*.

UHC, an extended version of EUC-KR, is a character set that assigns complete Hangul 8822 characters to the code range that cannot be used with EUC. However, it is not compatible with the code range handled as the C1 (0x80 to 0x9f) character set in EUC-KR because these codes are used as the first byte of double-byte characters in UHC.

Notes

This function is not multi-thread safe.

Prepare profile entity by thread or perform exclusive control for multithreading.

See Also

SceCesUcsProfileSheet, *SceCesUhcUcsProfile*, *sceCesMbcsUcsContextInit()*

SCE CONFIDENTIAL

sceCesUhcUcsProfileSetSbcs

Set single-byte character sets handled by UHC

Definition

```
#include <ces.h>
int sceCesUhcUcsProfileSetSbcs (
    SceCesUhcUcsProfile* profile,
    const SceCesSbcsUcsProfile* sbcsUcsProf
)
```

Arguments

profile Address of UCS conversion profile of UHC
sbcsUcsProf Address of UCS conversion profile to be used for UHC single-byte character

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid

Description

This function sets the character set in the range represented in 7-bit (0x00-0x7f) in UHC for the specified UCS conversion profile.

Specify the address of `SceCesUhcUcsProfile` obtained through initialization with an initialization function whose name begins by `sceCesUcsProfileInitUhc` in *profile*.

Specify the UCS conversion profile of the character set returned by one of the following functions in *sbcsUcsProf*:

UCS profile reference function of specifiable character sets	Mapping differences
<code>sceCesRefersUcsProfileKsX1003()</code>	0x5c (Won sign) 0x7e (OVERLINE)
<code>sceCesRefersUcsProfileKsX1003Tilde0x7e()</code>	0x5c (Won sign) 0x7e (TILDE)
<code>sceCesRefersUcsProfileAscii()</code>	0x5c (REVERSE SOLIDUS) 0x7e (TILDE)

If SCE_CES_ERROR_INVALID_PROFILE returns, it means that a NULL pointer has been passed to *profile* or *sbcsUcsProf*, or that the profile that has been passed is not appropriate.

Notes

This function is not multi-thread safe.

Prepare profile entity by thread or perform exclusive control for multithreading.

See Also

`SceCesUhcUcsProfile`, `SceCesSbcsUcsProfile`, `sceCesUcsProfileInitUhc()`

UCS Conversion Profile of EUC-KR

SceCesEucKrUcsProfile

Profile holding EUC-KR/UCS conversion information

Definition

```
#include <ces.h>
typedef struct SceCesEucKrUcsProfile{
    /* omitted */
} SceCesEucKrUcsProfile;
```

Description

This is the *profile* type holding character sets used by EUC-KR/UCS conversion information. The address of this data type can be retrieved with one of the functions whose name begins by `sceCesUcsProfileInit`, listed under “See Also”.

Reference pages are not provided for the following 1-character processing functions using this data type.

Refer to the description pages of the 1-character processing functions provided for `SceCesMbcUcsProfile`, which has the similar argument structure, except for receiving this data type address as the profile argument.

Function for <code>SceCesEucKrUcsProfile</code>	Function for <code>SceCesMbcUcsProfile</code>
<code>sceCesEucKrToUtf32()</code>	<code>sceCesMbcToUtf32()</code>
<code>sceCesEucKrToUtf32be()</code>	<code>sceCesMbcToUtf32be()</code>
<code>sceCesEucKrToUtf32le()</code>	<code>sceCesMbcToUtf32le()</code>
<code>sceCesEucKrToUtf16()</code>	<code>sceCesMbcToUtf16()</code>
<code>sceCesEucKrToUtf16be()</code>	<code>sceCesMbcToUtf16be()</code>
<code>sceCesEucKrToUtf16le()</code>	<code>sceCesMbcToUtf16le()</code>
<code>sceCesEucKrToUtf8()</code>	<code>sceCesMbcToUtf8()</code>
<code>sceCesEucKrToUcs2()</code>	<code>sceCesMbcToUcs2()</code>
<code>sceCesUtf32ToEucKr()</code>	<code>sceCesUtf32ToMbc()</code>
<code>sceCesUtf32beToEucKr()</code>	<code>sceCesUtf32beToMbc()</code>
<code>sceCesUtf32leToEucKr()</code>	<code>sceCesUtf32leToMbc()</code>
<code>sceCesUtf16ToEucKr()</code>	<code>sceCesUtf16ToMbc()</code>
<code>sceCesUtf16beToEucKr()</code>	<code>sceCesUtf16beToMbc()</code>
<code>sceCesUtf16leToEucKr()</code>	<code>sceCesUtf16leToMbc()</code>
<code>sceCesUtf8ToEucKr()</code>	<code>sceCesUtf8ToMbc()</code>
<code>sceCesUcs2ToEucKr()</code>	<code>sceCesUcs2ToMbc()</code>

See Also

```
sceCesUcsProfileInitEucKr(), sceCesUcsProfileInitUhc(),
sceCesGetMbcUcsProfile()
```

SCE CONFIDENTIAL

sceCesUcsProfileInitEucKr

Initialize the profile holding EUC-KR/UCS conversion information

Definition

```
#include <ces.h>
SceCesEucKrUcsProfile* sceCesUcsProfileInitEucKr
    SceCesUcsProfileSheet* profSheet
)
```

Arguments

profSheet Address of the UCS profile sheet

Return Values

The address specified in *profSheet* returns as the *SceCesEucKrUcsProfile** type.

Description

In *profSheet*, this function configures the profile holding EUC-KR/UCS conversion information, and returns its address as the *SceCesEucKrUcsProfile* type.

Prepare the entity of the *SceCesUcsProfileSheet* type on the caller side and specify its address in *profSheet*.

ASCII is used as single-byte character set and KS X 1001:2002 character set is used as double-byte character set.

As for control characters, the C1 character set (0x80 to 0x9f) is recognized in addition to the C0 character set (0x00 to 0x1f).

Notes

This function is not multi-thread safe.

Prepare profile entity by thread or perform exclusive control for multithreading.

See Also

SceCesUcsProfileSheet, *SceCesEucKrUcsProfile*, *sceCesMbcUcsContextInit()*

One-Character Conversion Functions Using UCS Conversion Profiles

One-Character Conversion Functions Handling Single-Byte Character Codes and UCS

sceCesSbcToUtf32, sceCesSbcToUtf32be, sceCesSbcToUtf32le

Convert one character from single-byte character code to UTF-32

Definition

```
#include <ces.h>
int sceCesSbcToUtf32 (
    const SceCesSbcsUcsProfile *profile,
    uint8_t sbc,
    uint32_t *utf32
)
int sceCesSbcToUtf32be (
    const SceCesSbcsUcsProfile *profile,
    uint8_t sbc,
    uint32_t *utf32
)
int sceCesSbcToUtf32le (
    const SceCesSbcsUcsProfile *profile,
    uint8_t sbc,
    uint32_t *utf32
)
```

Arguments

<i>profile</i>	SBCS and UCS conversion profile address
<i>sbc</i>	Single-byte character code
<i>utf32</i>	Address of the variable for outputting UTF-32 character code

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid

Description

The function described here receives the UCS conversion profile of a single-byte character set and a single-byte code, and returns the code value representing that character in UTF-32.

If the calling function is `sceCesSbcToUtf32()`, the output value will be written in 32-bit units.

SCE CONFIDENTIAL

If you wish to expressly specify the endianness of the output value, use `sceCesSbcToUtf32be()` for big-endian and `sceCesSbcToUtf32le()` for little-endian.

Specify a profile indicating the type of single-byte character set and rules of conversion with UCS in *profile*. The profile can be referenced as the return value of a function whose name begins by `sceCesRefersUcsProfile`, and reference functions are available by single-byte character set. Specify in accordance with the character code specified in *sbc*.

Specify the single-byte character code to be converted to UTF-32 in *sbc*.

Specify the address for receiving the UTF-32 character code in *utf32*.

If `SCE_CES_ERROR_INVALID_PROFILE` returns, it means that the address of an invalid profile has been specified in *profile*.

If `SCE_CES_ERROR_INVALID_ENCODE` returns, it means that encoding has been determined to be invalid because although a 7-bit SBCS has been specified in *profile*, a value equal or greater than 0x80 has been passed to *sbc*.

If `SCE_CES_ERROR_UNASSIGNED_CODE` returns, it means that the character code to be converted to UCS is not defined in the profile.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been passed to *utf32*. This error is only reported when no other errors are detected, and means that the function would have been successful if the output buffer had been set appropriately.

When an error occurs, **utf32* will be initialized with 0.

Notes

This function is multi-thread safe.

See Also

`SceCesSbcsUcsProfile`, `sceCesGetMbcsUcsProfile()`

SCE CONFIDENTIAL

sceCesSbcToUtf16, sceCesSbcToUtf16be, sceCesSbcToUtf16le

Convert one character from single-byte character code to UTF-16 (BE/LE)

Definition

```
#include <ces.h>
int sceCesSbcToUtf16(
    const SceCesSbcsUcsProfile *profile,
    uint8_t sbc,
    uint16_t *utf16buf,
    uint32_t utf16max,
    uint32_t *utf16Len
)
int sceCesSbcToUtf16be(
    const SceCesSbcsUcsProfile *profile,
    uint8_t sbc,
    uint16_t *utf16buf,
    uint32_t utf16max,
    uint32_t *utf16Len
)
int sceCesSbcToUtf16le(
    const SceCesSbcsUcsProfile *profile,
    uint8_t sbc,
    uint16_t *utf16buf,
    uint32_t utf16max,
    uint32_t *utf16Len
)
```

Arguments

<i>profile</i>	SBCS and UCS conversion profile address
<i>sbc</i>	Single-byte character code
<i>utf16buf</i>	Address of the buffer for receiving UTF-16 character code
<i>utf16max</i>	Buffer size for retrieving UTF-16 character code (16-bit word count)
<i>utf16Len</i>	Address of the variable for receiving UTF-16 character code length (16-bit word count)

Return Values

Returns SCE_OK(0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

SCE CONFIDENTIAL

Description

The function described here receives the UCS conversion profile of a single-byte character set and a single-byte code, and returns the UTF-16 character code representing that character.

If the calling function is `sceCesSbcToUtf16()`, the output value will be written in 16-bit units.

If you wish to expressly specify the endianness of the output value, use `sceCesSbcToUtf16be()` for big-endian and `sceCesSbcToUtf16le()` for little-endian.

Specify a profile indicating the type of single-byte character set and rules of conversion with UCS in *profile*. The profile can be referenced as the return value of a function whose name begins by `sceCesRefersUcsProfile`, and reference functions are available by single-byte character set. Specify in accordance with the character code specified in *sbc*.

Specify the single-byte character code to be converted to UTF-16 in *sbc*.

Specify the address for outputting the UTF-16 16-bit code in *utf16buf*.

Specify the size (16-bit word count) in which the UTF-16 16-bit code can be output in *utf16max*.

Specify the address of the variable for receiving the length of the UTF-16 character code (16-bit word count) in *utf16Len*. If a NULL pointer has been specified, this argument will be ignored.

If `SCE_CES_ERROR_INVALID_PROFILE` returns, it means that the address of an invalid profile has been specified in *profile*.

If `SCE_CES_ERROR_INVALID_ENCODE` returns, it means that encoding has been determined to be invalid because although a 7-bit SBCS has been specified in *profile*, a value equal or greater than 0x80 has been passed to *sbc*.

If `SCE_CES_ERROR_UNASSIGNED_CODE` returns, it means that the character code to be converted to UCS is not defined in the profile.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been given to *utf16buf*, or that 0 has been given to *utf16max*.

The return of `SCE_CES_ERROR_DST_BUFFER_END` indicates that the UTF-16 code could not be stored because output buffer size was not sufficient.

Nothing will be written to **utf16buf* if an error occurs.

Notes

This function is multi-thread safe.

See Also

`SceCesSbcsUcsProfile`, `sceCesUtf16ToSbc()`, `sceCesUtf16beToSbc()`,
`sceCesUtf16leToSbc()`

sceCesSbcToUtf8

Convert one character from single-byte character code to UTF-8

Definition

```
#include <ces.h>
int sceCesSbcToUtf8 (
    const SceCesSbcsUcsProfile *profile,
    uint8_t sbc,
    uint8_t *utf8buf,
    uint32_t utf8max,
    uint32_t *utf8Len
)
```

Arguments

<i>profile</i>	SBCS and UCS conversion profile address
<i>sbc</i>	Single-byte character code
<i>utf8buf</i>	Address of the buffer for receiving UTF-8 character code
<i>utf8max</i>	Buffer size for retrieving UTF-8 character code (byte count)
<i>utf8Len</i>	Address of the variable for receiving UTF-8 character code length (byte count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

The function receives the UCS conversion profile of a single-byte character set and a single-byte code, and returns the UTF-8 character code representing that character.

Specify a profile indicating the type of single-byte character set and rules of conversion with UCS in *profile*. The profile can be referenced as the return value of a function whose name begins by *sceCesRefersUcsProfile*, and reference functions are available by single-byte character set.

Specify in accordance with the character code specified in *sbc*.

Specify the single-byte character code to be converted to UTF-16 in *sbc*.

Specify the address for outputting the UTF-8 8-bit code in *utf8buf*.

Specify the size (byte count) in which the UTF-8 8-bit code can be output in *utf8max*.

Specify the address of the variable for receiving the length of the UTF-8 character code (byte count) in *utf8Len*. If a NULL pointer has been specified, this argument will be ignored.

SCE CONFIDENTIAL

In case of normal termination, an UTF-8 code of 1 to 4 bytes will be written in *utf8buf*, and the length of the UTF-8 code (byte count) will return to **utf8Len*. The value stored in **utf8Len* will coincide with the number of bytes that has been written in case of normal function termination. However, it will not indicate the number of bytes that has been written, but rather the length of the code (byte count) represented in UTF-8. Code length will be stored also if nothing has been written due to an error.

If `SCE_CES_ERROR_INVALID_PROFILE` returns, it means that the address of an invalid profile has been specified in *profile*.

If `SCE_CES_ERROR_INVALID_ENCODE` returns, it means that encoding has been determined to be invalid because although a 7-bit SBCS has been specified in *profile*, a value equal or greater than 0x80 has been passed to *sbc*.

If `SCE_CES_ERROR_UNASSIGNED_CODE` returns, it means that the character code to be converted to UCS is not defined in the profile.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been given to *utf8buf*.

The return of `SCE_CES_ERROR_DST_BUFFER_END` indicates that the UTF-8 code could not be stored because output buffer size was not sufficient.

Nothing will be written to **utf8buf* if an error occurs.

Notes

This function is multi-thread safe.

See Also

`SceCesSbcsUcsProfile`, `sceCesUtf8ToSbc()`

sceCesSbcToUcs2

Convert one character from single-byte character code to UCS-2

Definition

```
#include <ces.h>
int sceCesSbcToUcs2 (
    const SceCesSbcsUcsProfile *profile,
    uint8_t sbc,
    uint16_t *ucs2
)
```

Arguments

<i>profile</i>	SBCS and UCS conversion profile address
<i>sbc</i>	Single-byte character code
<i>ucs2</i>	Address of the variable for receiving UCS-2 character code

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid

Description

The function described here receives the UCS conversion profile of a single-byte character set and a single-byte code, and returns the code value representing that character in UCS-2.

Specify a profile indicating the type of single-byte character set and rules of conversion with UCS in *profile*. The profile can be referenced as the return value of a function whose name begins by *sceCesRefersUcsProfile*.

Specify in accordance with the character code specified in *sbc*.

Specify the single-byte character code to be converted to UCS-2 in *sbc*.

Specify the address for receiving the UCS-2 character code in *ucs2*.

If SCE_CES_ERROR_INVALID_PROFILE returns, it means that the address of an invalid profile has been specified in *profile*.

If SCE_CES_ERROR_INVALID_ENCODE returns, it means that encoding has been determined to be invalid because although a 7-bit SBCS has been specified in *profile*, a value equal or greater than 0x80 has been passed to *sbc*.

If SCE_CES_ERROR_UNASSIGNED_CODE returns, it means that the character code to be converted to UCS is not defined in the profile.

If SCE_CES_ERROR_INVALID_DST_BUFFER returns, it means that a NULL pointer has been passed to *ucs2*. This error is only reported when no other errors are detected, and means that the function would have been successful if the output buffer had been set appropriately.

SCE CONFIDENTIAL

When an error occurs, **ucs2* will be initialized with 0.

Notes

This function is multi-thread safe.

See Also

`SceCesSbcsUcsProfile, sceCesUcs2ToSbc()`

000004892117

SCE CONFIDENTIAL

sceCesUtf32ToSbc

Convert one character from UTF-32 to single-byte character code

Definition

```
#include <ces.h>
int sceCesUtf32ToSbc(
    uint32_t utf32,
    const SceCesSbcsUcsProfile *profile,
    uint8_t *sbc
)
```

Arguments

<i>utf32</i>	UTF-32 character code
<i>profile</i>	SBCS and UCS conversion profile address
<i>sbc</i>	Address for receiving single-byte character codes

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal character code detected in source character code
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid

Description

This function returns a UTF-32 character code as a character code of a specified single-byte character set.

Specify the UTF-32 character code in *utf32*.

Specify a profile indicating the type of single-byte character set and rules of conversion with UCS in *profile*. The profile can be referenced as the return value of a function whose name begins by *sceCesRefersUcsProfile*.

Specify the address for receiving the single-byte character code in *sbc*.

If SCE_CES_ERROR_INVALID_ENCODE returns, it means that encoding has been determined to be invalid because a value equal or greater than 0x00110000, which is outside the UTF-32 valid range, has been specified in *utf32*.

If SCE_CES_ERROR_ILLEGAL_CODE is returned, it means the code has been determined to be illegal because a value in the range between U+D800 and U+DFFF that is reserved as surrogate area has been passed in *utf32*.

If SCE_CES_ERROR_INVALID_PROFILE returns, it means that the address of an invalid profile has been specified in *profile*.

SCE CONFIDENTIAL

If `SCE_CES_ERROR_UNASSIGNED_CODE` returns, it means that the character specified in *utf32* was not present in the single-byte character set specified in *profile*.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been passed to *sbc*. This error is only reported when no other errors are detected, and means that the function would have been successful if the output buffer had been set appropriately.

When an error occurs, **sbc* will be initialized with 0.

Notes

This function is multi-thread safe.

See Also

`SceCesSbcsUcsProfile, sceCesSbcToUtf32 ()`

000004892117

SCE CONFIDENTIAL

sceCesUtf32beToSbc, sceCesUtf32leToSbc

Convert one character from UTF-32 (BE/LE) to single-byte character code

Definition

```
#include <ces.h>
int sceCesUtf32beToSbc (
    const uint32_t *utf32addr,
    const SceCesSbcsUcsProfile *profile,
    uint8_t *sbc
)
int sceCesUtf32leToSbc (
    const uint32_t *utf32addr,
    const SceCesSbcsUcsProfile *profile,
    uint8_t *sbc
)
```

Arguments

utf32addr Address storing UTF-32 (BE/LE) character code
profile SBCS and UCS conversion profile address
sbc Address for receiving single-byte character codes

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal character code detected in source character code
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid

Description

The functions described here return a UTF-32 character code stored in memory as a character code in a specified single-byte character set.

Use `sceCesUtf32beToSbc()` if the character code is stored in UTF-32BE (big-endian) and `sceCesUtf32leToSbc()` if it is stored in UTF-32LE (little endian).

Specify the address storing the UTF-32 character code in *utf32addr*.

Specify a profile indicating the type of single-byte character set and rules of conversion with UCS in *profile*. The profile can be referenced as the return value of a function whose name begins by `sceCesRefersUcsProfile`.

Specify the address for receiving the single-byte character code in *sbc*.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer has been passed in *utf32addr*.

SCE CONFIDENTIAL

If `SCE_CES_ERROR_INVALID_ENCODE` returns, it means that encoding has been determined to be invalid because a value equal or greater than 0x00110000, which is outside the UTF-32 valid range, has been specified in **utf32addr*.

If `SCE_CES_ERROR_ILLEGAL_CODE` is returned, it means the code has been determined to be illegal because a value in the range between U+D800 and U+DFFF that is reserved as surrogate area has been passed in **utf32addr*.

If `SCE_CES_ERROR_INVALID_PROFILE` returns, it means that the address of an invalid profile has been specified in *profile*.

If `SCE_CES_ERROR_UNASSIGNED_CODE` returns, it means that the character stored in *utf32addr* was not present in the single-byte character set specified in *profile*.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been passed to *sbc*. This error is only reported when no other errors are detected, and means that the function would have been successful if the output buffer had been set appropriately.

When an error occurs, **sbc* will be initialized with 0.

Notes

This function is multi-thread safe.

See Also

`SceCesSbcsUcsProfile`, `sceCesSbcToUtf32be()`, `sceCesSbcToUtf32le()`

SCE CONFIDENTIAL

sceCesUtf16ToSbc, sceCesUtf16beToSbc, sceCesUtf16leToSbc

Convert one character from UTF-16 to single-byte character code

Definition

```
#include <ces.h>
int sceCesUtf16ToSbc (
    const uint16_t *utf16addr,
    uint32_t utf16max,
    uint32_t *utf16Len,
    const SceCesSbcsUcsProfile *profile,
    uint8_t *sbc
)
int sceCesUtf16beToSbc (
    const uint16_t *utf16addr,
    uint32_t utf16max,
    uint32_t *utf16Len,
    const SceCesSbcsUcsProfile *profile,
    uint8_t *sbc
)
int sceCesUtf16leToSbc (
    const uint16_t *utf16addr,
    uint32_t utf16max,
    uint32_t *utf16Len,
    const SceCesSbcsUcsProfile *profile,
    uint8_t *sbc
)
```

Arguments

<i>utf16addr</i>	Address storing UTF-16 character code
<i>utf16max</i>	Size (16-bit word count) of the buffer storing UTF-16 character code
<i>utf16Len</i>	Address of the variable for receiving UTF-16 character code length (16-bit word count)
<i>profile</i>	SBCS and UCS conversion profile address
<i>sbc</i>	Address for receiving single-byte character codes

Return Values

Returns SCE_OK(0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal character code detected in source character code
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid

©SCEI

SCE CONFIDENTIAL

Description

The functions described here return a character passed in UTF-16 as a character code in a specified single-byte character set.

If the calling function is `sceCesUtf16ToSbc()`, the UTF-16 code will be read in 16-bit word units. If you wish to read out by expressly specifying endianness, use `sceCesUtf16beToSbc()` for big-endian and `sceCesUtf16leToSbc()` for little-endian.

Specify the address where the UTF-16 character code is stored in `utf16addr`.

Specify the length of the buffer (16-bit word count) for which recognition of UTF-16 character code is allowed in `utf16max`.

Specify the address of the variable for receiving the length of the character code (16-bit word count) stored in UTF-16 in `utf16Len`. If a NULL pointer has been specified, this argument will be ignored.

Specify a profile indicating the type of single-byte character set and rules of conversion with UCS in `profile`. The profile can be referenced as the return value of a function whose name begins by `sceCesRefersUcsProfile`.

Specify the address for receiving the single-byte character code in `sbc`.

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed in `utf16addr`, or that 0 has been passed in `utf16max`. 0 will be stored in `utf16Len`.

If `SCE_CES_ERROR_SRC_BUFFER_END` returns, it means that the character string specified in `utf16addr` has been interrupted in the midst of a code representing one character due to limitation by `utf16max`. In this case, the length of the character code (16-bit word count) whose recognition has been attempted will return to `*utf16Len` as a value greater than `utf16max`.

If `SCE_CES_ERROR_ILLEGAL_CODE` is returned, it means the code has been determined to be illegal because codes in the range between U+D800 and U+DFFF that is reserved as surrogate area were not used in pairs. 1 will be stored in `*utf16Len` as the length (16-bit word count) of the code determined to be illegal.

If `SCE_CES_ERROR_INVALID_PROFILE` returns, it means that the address of an invalid profile has been specified in `profile`.

If `SCE_CES_ERROR_UNASSIGNED_CODE` returns, it means that the character stored in `utf16addr` was not present in the single-byte character set specified in `profile`.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been passed to `sbc`. This error is only reported when no other errors are detected, and means that the function would have been successful if the output buffer had been set appropriately.

When an error occurs, `*sbc` will be initialized with 0.

Notes

This function is multi-thread safe.

See Also

`SceCesSbcsUcsProfile`, `sceCesSbcToUtf16()`, `sceCesSbcToUtf16be()`,
`sceCesSbcToUtf16le()`

SCE CONFIDENTIAL

sceCesUtf8ToSbc

Convert one character from UTF-8 to single-byte character code

Definition

```
#include <ces.h>
int sceCesUtf8ToSbc (
    const uint8_t *utf8addr,
    uint32_t utf8max,
    uint32_t *utf8Len,
    const SceCesSbcsUcsProfile *profile,
    uint8_t *sbc
)
```

Arguments

<i>utf8addr</i>	Address storing UTF-8 character code
<i>utf8max</i>	Size (byte count) of the buffer storing UTF-8 character code
<i>utf8Len</i>	Address of the variable for receiving UTF-8 character code length (byte count)
<i>profile</i>	SBCS and UCS conversion profile address
<i>sbc</i>	Address for receiving single-byte character codes

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal character code detected in source character code
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid

Description

This function returns a character passed in UTF-8 as a character code in a specified single-byte character set.

Specify the address where the UTF-8 character code is stored in *utf8addr*.

Specify the length of the buffer (byte count) for which recognition of UTF-8 character code is allowed in *utf8max*.

Specify the address of the variable for receiving the length (byte count) of the stored UTF-8 character code in *utf8Len*. If a NULL pointer has been specified, this argument will be ignored.

SCE CONFIDENTIAL

Specify a profile indicating the type of single-byte character set and rules of conversion with UCS in *profile*. The profile can be referenced as the return value of a function whose name begins by *sceCesRefersUcsProfile*.

Specify the address for receiving the single-byte character code in *sbc*.

In case of normal termination, the value stored in **utf8Len* will be from 1 to 4.

If *SCE_CES_ERROR_INVALID_SRC_BUFFER* returns, it means that a NULL pointer has been passed in *utf8addr*, or that 0 has been passed in *utf8max*. 0 will be stored in *utf8Len*.

If *SCE_CES_ERROR_SRC_BUFFER_END* returns, it means that the character string specified in *utf8addr* has been interrupted in the midst of a code representing one character due to limitation by *utf8max*. In case of this error, the character code length determined from the first UTF-8 byte will return to **utf8Len*. Note that a higher value than that specified in *utf8max* will return.

If *SCE_CES_ERROR_INVALID_ENCODE* returns, it means that encoding has been determined to be invalid because the address specified in *utf8addr* stored a byte string that could not be recognized as UTF-8. The number of bytes successfully recognized as UTF-8 will return to **utf8Len* as a value between 0 and 5.

If *SCE_CES_ERROR_ILLEGAL_CODE* returns, it means that the code has been determined to be illegal because the address specified in *utf8addr* contains a codes in the U+D800 to U+DFFF range reserved as a surrogate area, or encoding with an unnecessarily large number of bytes (such as representing U+0000 as the 2-byte sequence C0,80).

If *SCE_CES_ERROR_INVALID_PROFILE* returns, it means that the address of an invalid profile has been specified in *profile*.

If *SCE_CES_ERROR_UNASSIGNED_CODE* returns, it means that the character stored in *utf8addr* was not present in the single-byte character set specified in *profile*.

If *SCE_CES_ERROR_INVALID_DST_BUFFER* returns, it means that a NULL pointer has been passed to *sbc*. This error is only reported when no other errors are detected, and means that the function would have been successful if the output buffer had been set appropriately.

When an error occurs, **sbc* will be initialized with 0.

Notes

This function is multi-thread safe.

See Also

SceCesSbcsUcsProfile, *sceCesSbcToUtf8()*

SCE CONFIDENTIAL

sceCesUcs2ToSbc

Convert one character from UCS-2 to single-byte character code

Definition

```
#include <ces.h>
int sceCesUcs2ToSbc (
    uint32_t ucs2,
    const SceCesSbcsUcsProfile *profile,
    uint8_t *sbc
)
```

Arguments

<i>ucs2</i>	UCS-2 character code
<i>profile</i>	SBCS and UCS conversion profile address
<i>sbc</i>	Address for receiving single-byte character codes

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal character code detected in source character code
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid

Description

This function returns a character passed in UCS-2 as a character code in a specified single-byte character set.

Specify the UCS-2 character code in *ucs2*.

Specify a profile indicating the type of single-byte character set and rules of conversion with UCS in *profile*. The profile can be referenced as the return value of a function whose name begins by *sceCesRefersUcsProfile*.

Specify the address for receiving the single-byte character code in *sbc*.

If SCE_CES_ERROR_ILLEGAL_CODE is returned, it means the code has been determined to be illegal because a value in the range between U+D800 and U+DFFF that is reserved as surrogate area has been passed in *ucs2*.

If SCE_CES_ERROR_INVALID_PROFILE returns, it means that the address of an invalid profile has been specified in *profile*.

If SCE_CES_ERROR_UNASSIGNED_CODE returns, it means that the character specified in *ucs2* was not present in the single-byte character set specified in *profile*.

If SCE_CES_ERROR_INVALID_DST_BUFFER returns, it means that a NULL pointer has been passed to *sbc*. This error is only reported when no other errors are detected, and means that the function would have been successful if the output buffer had been set appropriately.

When an error occurs, **sbc* will be initialized with 0.

SCE CONFIDENTIAL

Notes

This function is multi-thread safe.

See Also

SceCesSbcsUcsProfile, sceCesSbcToUcs2 ()

000004892117

One-Character Conversion Functions Handling Multi-Byte Character Codes (Including SBC) and UCS

sceCesMbcToUtf32, sceCesMbcToUtf32be, sceCesMbcToUtf32le

Convert one character from multi-byte character code to UTF-32

Definition

```
#include <ces.h>
int sceCesMbcToUtf32 (
    const SceCesMbcsUcsProfile *profile,
    const uint8_t *mbcAddr,
    uint32_t mbcMax,
    uint32_t *mbcLen,
    uint32_t *utf32
)
int sceCesMbcToUtf32be (
    const SceCesMbcsUcsProfile *profile,
    const uint8_t *mbcAddr,
    uint32_t mbcMax,
    uint32_t *mbcLen,
    uint32_t *utf32
)
int sceCesMbcToUtf32le (
    const SceCesMbcsUcsProfile *profile,
    const uint8_t *mbcAddr,
    uint32_t mbcMax,
    uint32_t *mbcLen,
    uint32_t *utf32
)
```

Arguments

<i>profile</i>	MBCS and UCS conversion profile address
<i>mbcAddr</i>	Address storing the multi-byte character code
<i>mbcMax</i>	Maximum length of the buffer storing the multi-byte character code (byte count)
<i>mbcLen</i>	Address of the variable for receiving the length (byte count) of the successfully recognized multi-byte character code
<i>utf32</i>	Address of the variable for outputting UTF-32 character code

Return Values

Returns `SCE_OK (0)` as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
<code>SCE_CES_ERROR_INVALID_PROFILE</code>	<code>0x805C0004</code>	Specified profile is invalid
<code>SCE_CES_ERROR_INVALID_SRC_BUFFER</code>	<code>0x805C0010</code>	Specified source buffer is invalid
<code>SCE_CES_ERROR_SRC_BUFFER_END</code>	<code>0x805C0011</code>	Specified source buffer is insufficient
<code>SCE_CES_ERROR_INVALID_ENCODE</code>	<code>0x805C0014</code>	Source encoding determined to be invalid
<code>SCE_CES_ERROR_UNASSIGNED_CODE</code>	<code>0x805C0020</code>	Code points in output destination encoding scheme are not defined
<code>SCE_CES_ERROR_INVALID_DST_BUFFER</code>	<code>0x805C0030</code>	Output destination buffer is invalid

Description

The functions described here read out an MBCS character code from a specified address, and return its size and the character code in UTF-32.

If the calling function is `sceCesMbcToUtf32()`, the output value will be written in 32-bit units.

If you wish to expressly specify the endianness of the output value, use `sceCesMbcToUtf32be()` for big-endian and `sceCesMbcToUtf32le()` for little-endian.

In *profile*, specify `const SceCesMbcUcsProfile*`. This can be obtained with the return value of the macro function `sceCesGetMbcUcsProfile()`, using the address of the UCS conversion profile for the character sets of each CES as the argument.

Obtain the address of the UCS conversion profile for the character sets of each CES with the following procedure.

The UCS conversion profile of multi-byte character sets can be obtained as the return value of profile initialization functions whose name begins by `sceCesUcsProfileInit`.

The type of the UCS conversion profile of single-byte character sets can be obtained as the return value of profile reference functions whose name begins by `sceCesRefersUcsProfile`.

Specify the address storing the character code of the character set indicated by the profile in *mbcAddr*.

Specify the maximum length (byte count) of the buffer storing the character code of the character set indicated by the *profile* in *mbcMax*.

Specify the address of the variable for receiving the length (byte count) of the character code of the character set indicated by the recognized profile in *mbcLen*. The length of the recognized byte string will be returned even if an error occurs. If a NULL pointer has been specified, this argument will be ignored.

Specify the address for receiving the UTF-32 character code in *utf32*.

If `SCE_CES_ERROR_INVALID_PROFILE` returns, it means that the address of an invalid profile has been specified in *profile*.

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed in *mbcAddr*, or that 0 has been passed in *mbcMax*. 0 will be stored in **mbcLen*.

If `SCE_CES_ERROR_SRC_BUFFER_END` returns, it means that the character specified in *mbcAddr* has been interrupted in the midst of a code representing one character due to limitation by *mbcMax*. In this case, the length of character code determined in accordance with the code that has been successfully read will return to **mbcLen*. Note that the returned value will be larger than *mbcMax*.

SCE CONFIDENTIAL

If `SCE_CES_ERROR_INVALID_ENCODE` returns, it means that the encoding has been determined to be invalid because it includes byte values that cannot be recognized, such as in the case where the byte string passed to *mbcAddr* includes values within the MBCS's holding area.

If `SCE_CES_ERROR_UNASSIGNED_CODE` returns, it means that the character code to be converted to UCS is not defined in the profile.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been passed to *utf32*. This error is only reported when no other errors are detected, and means that the function would have been successful if the output buffer had been set appropriately.

When an error occurs, **utf32* will be initialized with 0.

Notes

This function is multi-thread safe.

See Also

`SceCesMbcUcsProfile`, `sceCesUtf32ToMbc()`, `sceCesUtf32beToMbc()`,
`sceCesUtf32leToMbc()`

SCE CONFIDENTIAL

sceCesMbcToUtf16, sceCesMbcToUtf16be, sceCesMbcToUtf16le

Convert one character from multi-byte character code to UTF-16 (BE/LE)

Definition

```
#include <ces.h>
int sceCesMbcToUtf16(
    const SceCesMbcUcsProfile *profile,
    const uint8_t *mbcAddr,
    uint32_t mbcMax,
    uint32_t *mbcLen,
    uint16_t *utf16buf,
    uint32_t utf16max,
    uint32_t *utf16Len
)
int sceCesMbcToUtf16be(
    const SceCesMbcUcsProfile *profile,
    const uint8_t *mbcAddr,
    uint32_t mbcMax,
    uint32_t *mbcLen,
    uint16_t *utf16buf,
    uint32_t utf16max,
    uint32_t *utf16Len
)
int sceCesMbcToUtf16le(
    const SceCesMbcUcsProfile *profile,
    const uint8_t *mbcAddr,
    uint32_t mbcMax,
    uint32_t *mbcLen,
    uint16_t *utf16buf,
    uint32_t utf16max,
    uint32_t *utf16Len
)
```

Arguments

<i>profile</i>	MBCS and UCS conversion profile address
<i>mbcAddr</i>	Address storing the multi-byte character code
<i>mbcMax</i>	Maximum length of the buffer storing the multi-byte character code (byte count)
<i>mbcLen</i>	Address of the variable for receiving the length (byte count) of the successfully recognized multi-byte character code
<i>utf16buf</i>	Address of the buffer for receiving UTF-16 character code
<i>utf16max</i>	Buffer size for retrieving UTF-16 character code (16-bit word count)
<i>utf16Len</i>	UTF-16 character code length (16-bit word count)

Return Values

Returns `SCE_OK (0)` as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
<code>SCE_CES_ERROR_INVALID_PROFILE</code>	<code>0x805C0004</code>	Specified profile is invalid
<code>SCE_CES_ERROR_INVALID_SRC_BUFFER</code>	<code>0x805C0010</code>	Specified source buffer is invalid
<code>SCE_CES_ERROR_SRC_BUFFER_END</code>	<code>0x805C0011</code>	Specified source buffer is insufficient
<code>SCE_CES_ERROR_INVALID_ENCODE</code>	<code>0x805C0014</code>	Source encoding determined to be invalid
<code>SCE_CES_ERROR_UNASSIGNED_CODE</code>	<code>0x805C0020</code>	Code points in output destination encoding scheme are not defined
<code>SCE_CES_ERROR_INVALID_DST_BUFFER</code>	<code>0x805C0030</code>	Output destination buffer is invalid
<code>SCE_CES_ERROR_DST_BUFFER_END</code>	<code>0x805C0031</code>	Output destination buffer is insufficient

Description

The functions described here read out an MBCS character code from a specified address, and return its size, the character code in UTF-16 and code length.

If the calling function is `sceCesMbcToUtf16()`, the output value will be written in 16-bit units.

If you wish to expressly specify the endianness of the output value, use `sceCesMbcToUtf16be()` for big-endian and `sceCesMbcToUtf16le()` for little-endian.

In *profile*, specify `const SceCesMbcUcsProfile*`. This can be obtained with the return value of the macro function `sceCesGetMbcUcsProfile()`, using the address of the UCS conversion profile for the character sets of each CES as the argument.

Obtain the address of the UCS conversion profile for the character sets of each CES with the following procedure.

The UCS conversion profile of multi-byte character sets can be obtained as the return value of profile initialization functions whose name begins by `sceCesUcsProfileInit`.

The type of the UCS conversion profile of single-byte character sets can be obtained as the return value of profile reference functions whose name begins by `sceCesRefersUcsProfile`.

Specify the address storing the character code of the character set indicated by the profile in *mbcAddr*.

Specify the maximum length (byte count) of the buffer storing the character code of the character set indicated by the *profile* in *mbcMax*.

Specify the address of the variable for receiving the length (byte count) of the character code of the character set indicated by the recognized profile in *mbcLen*. The length of the recognized byte string will be returned even if an error occurs. If a NULL pointer has been specified, this argument will be ignored.

Specify the address for outputting the UTF-16 16-bit code in *utf16buf*.

Specify the size (16-bit word count) in which the UTF-16 16-bit code can be output in *utf16max*.

Specify the address of the variable for receiving the length of the UTF-16 character code (16-bit word count) in *utf16Len*. If a NULL pointer has been specified, this argument will be ignored.

If `SCE_CES_ERROR_INVALID_PROFILE` returns, it means that the address of an invalid profile has been specified in *profile*.

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed in *mbcAddr*, or that 0 has been passed in *mbcMax*. 0 will be stored in **mbcLen*.

SCE CONFIDENTIAL

If `SCE_CES_ERROR_SRC_BUFFER_END` returns, it means that the character specified in `mbcAddr` has been interrupted in the midst of a code representing one character due to limitation by `mbcMax`. In this case, the length of character code determined in accordance with the code that has been successfully read will return to `*mbcLen`. Note that the returned value will be larger than `mbcMax`.

If `SCE_CES_ERROR_INVALID_ENCODE` returns, it means that the encoding has been determined to be invalid because it includes byte values that cannot be recognized, such as in the case where the byte string passed to `mbcAddr` includes values within the MBCS's holding area.

If `SCE_CES_ERROR_UNASSIGNED_CODE` returns, it means that the character code to be converted to UCS is not defined in the profile.

In the cases of the above errors, there will be no output to `utf16buf` and 0 will be stored in `utf16Len`.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been specified in `utf16buf`.

The return of `SCE_CES_ERROR_DST_BUFFER_END` indicates that the UTF-16 code could not be stored because output buffer size was not sufficient.

These two error cases caused by the output buffer are only reported when no other errors are detected, and it means that the function would have been successfully completed if the output buffer had been appropriately specified. In this case, there will be no output to `utf16buf`, but the length of the code (16-bit word count) that was to be output will return to `*utf16Len`.

Notes

This function is multi-thread safe.

See Also

`SceCesMbcToUcsProfile`, `sceCesUtf16ToMbc()`, `sceCesUtf16beToMbc()`,
`sceCesUtf16leToMbc()`

SCE CONFIDENTIAL

sceCesMbcToUtf8

Convert one character from multi-byte character code to UTF-8

Definition

```
#include <ces.h>
int sceCesMbcToUtf8 (
    const SceCesMbcUcsProfile *profile,
    const uint8_t *mbcAddr,
    uint32_t mbcMax,
    uint32_t *mbcLen,
    uint8_t *utf8buf,
    uint32_t utf8max,
    uint32_t *utf8Len
)
```

Arguments

<i>profile</i>	MBCS and UCS conversion profile address
<i>mbcAddr</i>	Address storing the multi-byte character code
<i>mbcMax</i>	Maximum length of the buffer storing the multi-byte character code (byte count)
<i>mbcLen</i>	Address of the variable for receiving the length (byte count) of the successfully recognized multi-byte character code
<i>utf8buf</i>	Address of the buffer for receiving UTF-8 character code
<i>utf8max</i>	Buffer size for retrieving UTF-8 character code (byte count)
<i>utf8Len</i>	UTF-8 character code length (byte count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

The function described here reads out an MBCS character code from a specified address, and returns its size, the character code in UTF-8 and code length.

In *profile*, specify `const SceCesMbcUcsProfile*`. This can be obtained with the return value of the macro function `sceCesGetMbcUcsProfile()`, using the address of the UCS conversion profile for the character sets of each CES as the argument.

Obtain the address of the UCS conversion profile for the character sets of each CES with the following procedure.

The UCS conversion profile of multi-byte character sets can be obtained as the return value of profile initialization functions whose name begins by *sceCesUcsProfileInit*.

The type of the UCS conversion profile of single-byte character sets can be obtained as the return value of profile reference functions whose name begins by *sceCesRefersUcsProfile*.

Specify the address storing the character code of the character set indicated by the profile in *mbcAddr*.

Specify the maximum length (byte count) of the buffer storing the character code of the character set indicated by the *profile* in *mbcMax*.

Specify the address of the variable for receiving the length (byte count) of the character code of the character set indicated by the recognized profile in *mbcLen*. The length of the recognized byte string will be returned even if an error occurs. If a NULL pointer has been specified, this argument will be ignored.

Specify the address for outputting the UTF-8 8-bit code in *utf8buf*.

Specify the size (byte count) in which the UTF-8 8-bit code can be output in *utf8max*.

Specify the address of the variable for receiving the length of the UTF-8 character code (byte count) in *utf8Len*. If a NULL pointer has been specified, this argument will be ignored.

If *SCE_CES_ERROR_INVALID_PROFILE* returns, it means that the address of an invalid profile has been specified in *profile*.

If *SCE_CES_ERROR_INVALID_SRC_BUFFER* returns, it means that a NULL pointer has been passed in *mbcAddr*, or that 0 has been passed in *mbcMax*. 0 will be stored in **mbcLen*.

If *SCE_CES_ERROR_SRC_BUFFER_END* returns, it means that the character specified in *mbcAddr* has been interrupted in the midst of a code representing one character due to limitation by *mbcMax*. In this case, the length of character code determined in accordance with the code that has been successfully read will return to **mbcLen*. Note that the returned value will be larger than *mbcMax*.

If *SCE_CES_ERROR_INVALID_ENCODE* returns, it means that the encoding has been determined to be invalid because it includes byte values that cannot be recognized, such as in the case where the byte string passed to *mbcAddr* includes values within the MBCS's holding area.

If *SCE_CES_ERROR_UNASSIGNED_CODE* returns, it means that the character code to be converted to UCS is not defined in the profile.

In the cases of the above errors, there will be no output to *utf8buf* and 0 will be stored in *utf8Len*.

If *SCE_CES_ERROR_INVALID_DST_BUFFER* returns, it means that a NULL pointer has been specified in *utf8buf*.

The return of *SCE_CES_ERROR_DST_BUFFER_END* indicates that the UTF-8 code could not be stored because output buffer size was not sufficient.

These two error cases caused by the output buffer are only reported when no other errors are detected, and it means that the function would have been successfully completed if the output buffer had been appropriately specified. In this case, there will be no output to *utf8buf*, but the length (byte count) of the UTF-8 code that was to be output will return to **utf8Len*.

Notes

This function is multi-thread safe.

See Also

SceCesMbcUcsProfile, *sceCesUtf8ToMbc* ()

sceCesMbcToUcs2

Convert one character from multi-byte character code to UCS-2

Definition

```
#include <ces.h>
int sceCesMbcToUcs2 (
    const SceCesMbcUcsProfile *profile,
    const uint8_t *mbcAddr,
    uint32_t mbcMax,
    uint32_t *mbcLen,
    uint16_t *ucs2
)
```

Arguments

<i>profile</i>	MBCS and UCS conversion profile address
<i>mbcAddr</i>	Address storing the multi-byte character code
<i>mbcMax</i>	Maximum length of the buffer storing the multi-byte character code (byte count)
<i>mbcLen</i>	Address of the variable for receiving the length (byte count) of the successfully recognized multi-byte character code
<i>ucs2</i>	Address of the variable for receiving UCS-2 character code

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined
SCE_CES_ERROR_OUT_OF_CODE_RANGE	0x805C0024	Character outside the representable code range of the output destination encoding scheme is detected
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid

Description

This function reads out an MBCS character code from a specified address, and returns its size and the character code in UCS-2.

In *profile*, specify `const SceCesMbcUcsProfile*`. This can be obtained with the return value of the macro function `sceCesGetMbcUcsProfile()`, using the address of the UCS conversion profile for the character sets of each CES as the argument.

Obtain the address of the UCS conversion profile for the character sets of each CES with the following procedure.

The UCS conversion profile of multi-byte character sets can be obtained as the return value of profile initialization functions whose name begins by *sceCesUcsProfileInit*.

The type of the UCS conversion profile of single-byte character sets can be obtained as the return value of profile reference functions whose name begins by *sceCesRefersUcsProfile*.

Specify the address storing the character code of the character set indicated by the profile in *mbcAddr*.

Specify the maximum length (byte count) of the buffer storing the character code of the character set indicated by the *profile* in *mbcMax*.

Specify the address of the variable for receiving the length (byte count) of the character code of the character set indicated by the recognized profile in *mbcLen*. The length of the recognized byte string will be returned even if an error occurs. If a NULL pointer has been specified, this argument will be ignored.

Specify the address for receiving the UCS-2 character code in *ucs2*.

If *SCE_CES_ERROR_INVALID_PROFILE* returns, it means that the address of an invalid profile has been specified in *profile*.

If *SCE_CES_ERROR_INVALID_SRC_BUFFER* returns, it means that a NULL pointer has been passed in *mbcAddr*, or that 0 has been passed in *mbcMax*. 0 will be stored in **mbcLen*.

If *SCE_CES_ERROR_SRC_BUFFER_END* returns, it means that the character specified in *mbcAddr* has been interrupted in the midst of a code representing one character due to limitation by *mbcMax*. In this case, the length of character code determined in accordance with the code that has been successfully read will return to **mbcLen*. Note that the returned value will be larger than *mbcMax*.

If *SCE_CES_ERROR_INVALID_ENCODE* returns, it means that the encoding has been determined to be invalid because it includes byte values that cannot be recognized, such as in the case where the byte string passed to *mbcAddr* includes values within the MBCS's holding area.

If *SCE_CES_ERROR_UNASSIGNED_CODE* returns, it means that the character code to be converted to UCS is not defined in the profile.

If *SCE_CES_ERROR_OUT_OF_CODE_RANGE* returns, it means that output has been determined to be impossible because a character code equal or greater than U+00010000, which is outside the range representable in UCS-2, was encoded in *mbcAddr*.

If *SCE_CES_ERROR_INVALID_DST_BUFFER* returns, it means that a NULL pointer has been passed to *ucs2*. This error is only reported when no other errors are detected, and means that the function would have been successful if the output buffer had been set appropriately.

When an error occurs, **ucs2* will be initialized with 0.

Notes

This function is multi-thread safe.

See Also

SceCesMbcUcsProfile, *sceCesUcs2ToMbc* ()

SCE CONFIDENTIAL

sceCesUtf32ToMbc

Convert one character from UTF-32 to multi-byte character code

Definition

```
#include <ces.h>
int sceCesUtf32ToMbc (
    uint32_t utf32,
    const SceCesMbcUcsProfile *profile,
    uint8_t *mbcBuf,
    uint32_t mbcMax,
    uint32_t *mbcLen
)
```

Arguments

<i>utf32</i>	UTF-32 character code
<i>profile</i>	MBCS and UCS conversion profile address
<i>mbcBuf</i>	Address of the buffer for receiving the character code in multi-byte format
<i>mbcMax</i>	Maximum length (byte count) of the buffer for receiving the character code in multi-byte format
<i>mbcLen</i>	Address of the variable for receiving multi-byte character code length (byte count)

Return Values

Returns `SCE_OK (0)` as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
<code>SCE_CES_ERROR_INVALID_PROFILE</code>	<code>0x805C0004</code>	Specified profile is invalid
<code>SCE_CES_ERROR_INVALID_ENCODE</code>	<code>0x805C0014</code>	Source encoding determined to be invalid
<code>SCE_CES_ERROR_ILLEGAL_CODE</code>	<code>0x805C0015</code>	Illegal character code detected in source character code
<code>SCE_CES_ERROR_UNASSIGNED_CODE</code>	<code>0x805C0020</code>	Code points in output destination encoding scheme are not defined
<code>SCE_CES_ERROR_INVALID_DST_BUFFER</code>	<code>0x805C0030</code>	Output destination buffer is invalid
<code>SCE_CES_ERROR_DST_BUFFER_END</code>	<code>0x805C0031</code>	Output destination buffer is insufficient

Description

This function returns a character specified in UTF-32 as an MBCS character code and the length of the character code.

Specify the UTF-32 character code in *utf32*.

In *profile*, specify `const SceCesMbcUcsProfile*`. This can be obtained with the return value of the macro function `sceCesGetMbcUcsProfile()`, using the address of the UCS conversion profile for the character sets of each CES as the argument.

Obtain the address of the UCS conversion profile for the character sets of each CES with the following procedure.

SCE CONFIDENTIAL

The UCS conversion profile of multi-byte character sets can be obtained as the return value of profile initialization functions whose name begins by `sceCesUcsProfileInit`.

The type of the UCS conversion profile of single-byte character sets can be obtained as the return value of profile reference functions whose name begins by `sceCesRefersUcsProfile`.

Specify the address for outputting the character code in the character set indicated by the profile in `mbcBuf`.

Specify the outputtable size (byte count) of the character code in the character set indicated by the profile in `mbcMax`.

Specify the address of the variable for receiving the length (byte count) of the character code in the character set indicated by the profile in `mbcLen`. If a NULL pointer has been specified, this argument will be ignored.

If `SCE_CES_ERROR_INVALID_ENCODE` returns, it means that encoding has been determined to be invalid because a value equal or greater than 0x00110000, which is outside the UTF-32 valid range, has been specified in `utf32`.

If `SCE_CES_ERROR_ILLEGAL_CODE` is returned, it means the code has been determined to be illegal because a value in the range between U+D800 and U+DFFF that is reserved as surrogate area has been passed in `utf32`.

If `SCE_CES_ERROR_INVALID_PROFILE` returns, it means that the address of an invalid profile has been specified in `profile`.

If `SCE_CES_ERROR_UNASSIGNED_CODE` returns, it means that the character specified in `utf32` was not present in the character set specified in `profile`.

In the case of the above errors, nothing will be written to `mbcBuf` and 0 will be stored in `*mbcLen`.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been specified in `mbcBuf`.

The return of `SCE_CES_ERROR_DST_BUFFER_END` indicates that the MBCS code could not be stored because output buffer size was not sufficient.

These two error cases caused by the output buffer are only reported when no other errors are detected, and it means that the function would have been successfully completed if the output buffer had been appropriately specified. In this case, there will be no output to `mbcBuf`, but the length of the code (byte count) that was to be output will return to `*mbcLen`.

Notes

This function is multi-thread safe.

See Also

`SceCesMbcUcsProfile`, `sceCesMbcToUtf32()`

SCE CONFIDENTIAL

sceCesUtf32beToMbc, sceCesUtf32leToMbc

Convert one character from UTF-32 (BE/LE) to multi-byte character code

Definition

```
#include <ces.h>
int sceCesUtf32beToMbc (
    const uint32_t *utf32addr,
    const SceCesMbcUcsProfile *profile,
    uint8_t *mbcBuf,
    uint32_t mbcMax,
    uint32_t *mbcLen
)
int sceCesUtf32leToMbc (
    const uint32_t *utf32addr,
    const SceCesMbcUcsProfile *profile,
    uint8_t *mbcBuf,
    uint32_t mbcMax,
    uint32_t *mbcLen
)
```

Arguments

<i>utf32addr</i>	Address storing UTF-32 character code
<i>profile</i>	MBCS and UCS conversion profile address
<i>mbcBuf</i>	Address of the buffer for receiving the character code in multi-byte format
<i>mbcMax</i>	Maximum length (byte count) of the buffer for receiving the character code in multi-byte format
<i>mbcLen</i>	Address of the variable for receiving multi-byte character code length (byte count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal character code detected in source character code
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

The functions described here return a character indicated by a UTF-32 character code stored in memory as an MBCS character code and the length of the character code.

Use `sceCesUtf32beToMbc()` if the character code is stored in UTF-32BE (big-endian) and `sceCesUtf32leToMbc()` if it is stored in UTF-32LE (little endian).

©SCEI

Specify the address storing the UTF-32 character code in *utf32addr*.

In *profile*, specify `const SceCesMbcUcsProfile*`. This can be obtained with the return value of the macro function `sceCesGetMbcUcsProfile()`, using the address of the UCS conversion profile for the character sets of each CES as the argument.

Obtain the address of the UCS conversion profile for the character sets of each CES with the following procedure.

The UCS conversion profile of multi-byte character sets can be obtained as the return value of profile initialization functions whose name begins by `sceCesUcsProfileInit`.

The type of the UCS conversion profile of single-byte character sets can be obtained as the return value of profile reference functions whose name begins by `sceCesRefersUcsProfile`.

Specify the address for outputting the character code in the character set indicated by the profile in *mbcBuf*.

Specify the outputtable size (byte count) of the character code in the character set indicated by the profile in *mbcMax*.

Specify the address of the variable for receiving the length (byte count) of the character code in the character set indicated by the profile in *mbcLen*. If a NULL pointer has been specified, this argument will be ignored.

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed in *utf32addr*.

If `SCE_CES_ERROR_INVALID_ENCODE` returns, it means that encoding has been determined to be invalid because a value equal or greater than 0x00110000, which is outside the UTF-32 valid range, has been specified in **utf32addr*.

If `SCE_CES_ERROR_ILLEGAL_CODE` is returned, it means the code has been determined to be illegal because a value in the range between U+D800 and U+DFFF that is reserved as surrogate area has been passed in **utf32addr*.

If `SCE_CES_ERROR_INVALID_PROFILE` returns, it means that the address of an invalid profile has been specified in *profile*.

If `SCE_CES_ERROR_UNASSIGNED_CODE` returns, it means that the character stored in *utf32addr* was not present in the character set specified in *profile*.

In the cases of the above errors, there will be no output to *mbcBuf* and 0 will be stored in **mbcLen*.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been specified in *mbcBuf*.

The return of `SCE_CES_ERROR_DST_BUFFER_END` indicates that the multi-byte character code could not be stored because output buffer size was not sufficient.

These two error cases caused by the output buffer are only reported when no other errors are detected, and it means that the function would have been successfully completed if the output buffer had been appropriately specified. In this case, there will be no output to *mbcBuf*, but the length of the code (byte count) that was to be output will return to **mbcLen*.

Notes

This function is multi-thread safe.

See Also

`SceCesMbcUcsProfile`, `sceCesMbcToUtf32be()`, `sceCesMbcToUtf32le()`

SCE CONFIDENTIAL

sceCesUtf16ToMbc, sceCesUtf16beToMbc, sceCesUtf16leToMbc

Convert one character from UTF-16 to multi-byte character code

Definition

```
#include <ces.h>
int sceCesUtf16ToMbc(
    const uint16_t *utf16addr,
    uint32_t utf16max,
    uint32_t *utf16Len,
    const SceCesMbcUcsProfile *profile,
    uint8_t *mbcBuf,
    uint32_t mbcMax,
    uint32_t *mbcLen
)
int sceCesUtf16beToMbc(
    const uint16_t *utf16addr,
    uint32_t utf16max,
    uint32_t *utf16Len,
    const SceCesMbcUcsProfile *profile,
    uint8_t *mbcBuf,
    uint32_t mbcMax,
    uint32_t *mbcLen
)
int sceCesUtf16leToMbc(
    const uint16_t *utf16addr,
    uint32_t utf16max,
    uint32_t *utf16Len,
    const SceCesMbcUcsProfile *profile,
    uint8_t *mbcBuf,
    uint32_t mbcMax,
    uint32_t *mbcLen
)
```

Arguments

<i>utf16addr</i>	Address storing UTF-16 character code
<i>utf16max</i>	Size (16-bit word count) of the buffer storing UTF-16 character code
<i>utf16Len</i>	Address of the variable for receiving UTF-16 character code length (16-bit word count)
<i>profile</i>	MBCS and UCS conversion profile address
<i>mbcBuf</i>	Address of the buffer for receiving the character code in multi-byte format
<i>mbcMax</i>	Maximum length (byte count) of the buffer for receiving the character code in multi-byte format
<i>mbcLen</i>	Address of the variable for receiving multi-byte character code length (byte count)

Return Values

Returns `SCE_OK (0)` as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
<code>SCE_CES_ERROR_INVALID_SRC_BUFFER</code>	<code>0x805C0010</code>	Specified source buffer is invalid
<code>SCE_CES_ERROR_SRC_BUFFER_END</code>	<code>0x805C0011</code>	Specified source buffer is insufficient
<code>SCE_CES_ERROR_ILLEGAL_CODE</code>	<code>0x805C0015</code>	Illegal character code detected in source character code
<code>SCE_CES_ERROR_INVALID_PROFILE</code>	<code>0x805C0004</code>	Specified profile is invalid
<code>SCE_CES_ERROR_UNASSIGNED_CODE</code>	<code>0x805C0020</code>	Code points in output destination encoding scheme are not defined
<code>SCE_CES_ERROR_INVALID_DST_BUFFER</code>	<code>0x805C0030</code>	Output destination buffer is invalid
<code>SCE_CES_ERROR_DST_BUFFER_END</code>	<code>0x805C0031</code>	Output destination buffer is insufficient

Description

The functions described here return a character specified in UTF-16 as an MBCS character code and the length of the character code.

If the calling function is `sceCesUtf16ToMbc()`, the UTF-16 code will be read in 16-bit word units. If you wish to read out by expressly specifying endianness, use `sceCesUtf16beToMbc()` for big-endian and `sceCesUtf16leToMbc()` for little-endian.

Specify the address where the UTF-16 character code is stored in *utf16addr*.

Specify the length of the buffer (16-bit word count) for which recognition of UTF-16 character code is allowed in *utf16max*.

Specify the address of the variable for receiving the length of the character code (16-bit word count) stored in UTF-16 in *utf16Len*. If a NULL pointer has been specified, this argument will be ignored.

In *profile*, specify `const SceCesMbcUcsProfile*`. This can be obtained with the return value of the macro function `sceCesGetMbcUcsProfile()`, using the address of the UCS conversion profile for the character sets of each CES as the argument.

Obtain the address of the UCS conversion profile for the character sets of each CES with the following procedure.

The UCS conversion profile of multi-byte character sets can be obtained as the return value of profile initialization functions whose name begins by `sceCesUcsProfileInit`.

The type of the UCS conversion profile of single-byte character sets can be obtained as the return value of profile reference functions whose name begins by `sceCesRefersUcsProfile`.

Specify the address for outputting the character code in the character set indicated by the profile in *mbcBuf*.

Specify the outputtable size (byte count) of the character code in the character set indicated by the profile in *mbcMax*.

Specify the address of the variable for receiving the length (byte count) of the character code in the character set indicated by the profile in *mbcLen*. If a NULL pointer has been specified, this argument will be ignored.

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed in *utf16addr*, or that 0 has been passed in *utf16max*. 0 will be stored in **utf16Len*.

SCE CONFIDENTIAL

If `SCE_CES_ERROR_SRC_BUFFER_END` returns, it means that the character string specified in `utf16addr` has been interrupted in the midst of a code representing one character due to limitation by `utf16max`. In this case, the length of the character code (16-bit word count) whose recognition has been attempted will return to `*utf16Len` as a value greater than `utf16max`.

If `SCE_CES_ERROR_ILLEGAL_CODE` is returned, it means the code has been determined to be illegal because codes in the range between U+D800 and U+DFFF that is reserved as surrogate area were not used in pairs. 1 will be stored in `*utf16Len` as the length of the code (16-bit word count) that was found to be illegal.

If `SCE_CES_ERROR_INVALID_PROFILE` returns, it means that the address of an invalid profile has been specified in `profile`.

If `SCE_CES_ERROR_UNASSIGNED_CODE` returns, it means that the character stored in `utf16addr` was not present in the character set specified in `profile`.

In the cases of the above errors, there will be no output to `mbcBuf` and 0 will be stored in `*mbcLen`.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been specified in `mbcBuf`.

The return of `SCE_CES_ERROR_DST_BUFFER_END` indicates that the multi-byte character code could not be stored because output buffer size was not sufficient.

These two error cases caused by the output buffer are only reported when no other errors are detected, and it means that the function would have been successfully completed if the output buffer had been appropriately specified. In this case, there will be no output to `mbcBuf`, but the length of the code (byte count) that was to be output will return to `*mbcLen`.

Notes

This function is multi-thread safe.

See Also

`SceCesMbcUcsProfile`, `sceCesMbcToUtf16()`, `sceCesMbcToUtf16be()`,
`sceCesMbcToUtf16le()`

SCE CONFIDENTIAL

sceCesUtf8ToMbc

Convert one character from UTF-8 to multi-byte character code

Definition

```
#include <ces.h>
int sceCesUtf8ToMbc (
    const uint8_t *utf8addr,
    uint32_t utf8max,
    uint32_t *utf8Len,
    const SceCesMbcUcsProfile *profile,
    uint8_t *mbcBuf,
    uint32_t mbcMax,
    uint32_t *mbcLen
)
```

Arguments

<i>utf8addr</i>	Address storing UTF-8 character code
<i>utf8max</i>	Size (byte count) of the buffer storing UTF-8 character code
<i>utf8Len</i>	Address of the variable for receiving UTF-8 character code length (byte count)
<i>profile</i>	MBCS and UCS conversion profile address
<i>mbcBuf</i>	Address of the buffer for receiving the character code in multi-byte format
<i>mbcMax</i>	Maximum length (byte count) of the buffer for receiving the character code in multi-byte format
<i>mbcLen</i>	Address of the variable for receiving multi-byte character code length (byte count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal character code detected in source character code
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

This function returns a character specified in UTF-8 as an MBCS character code and the length of the character code.

Specify the address where the UTF-8 character code is stored in *utf8addr*.

©SCEI

Specify the length of the buffer (byte count) for which recognition of UTF-8 character code is allowed in *utf8max*.

Specify the address of the variable for receiving the length (byte count) of the stored UTF-8 character code in *utf8Len*. If a NULL pointer has been specified, this argument will be ignored.

In *profile*, specify `const SceCesMbcUcsProfile*`. This can be obtained with the return value of the macro function `sceCesGetMbcUcsProfile()`, using the address of the UCS conversion profile for the character sets of each CES as the argument.

Obtain the address of the UCS conversion profile for the character sets of each CES with the following procedure.

The UCS conversion profile of multi-byte character sets can be obtained as the return value of profile initialization functions whose name begins by `sceCesUcsProfileInit`.

The type of the UCS conversion profile of single-byte character sets can be obtained as the return value of profile reference functions whose name begins by `sceCesRefersUcsProfile`.

Specify the address for outputting the character code in the character set indicated by the profile in *mbcBuf*.

Specify the outputtable size (byte count) of the character code in the character set indicated by the profile in *mbcMax*.

Specify the address of the variable for receiving the length (byte count) of the character code in the character set indicated by the profile in *mbcLen*. If a NULL pointer has been specified, this argument will be ignored.

If `SCE_CES_ERROR_INVALID_SRC_BUFFER` returns, it means that a NULL pointer has been passed in *utf8addr*, or that 0 has been passed in *utf8max*. 0 will be stored in **utf8Len*.

If `SCE_CES_ERROR_SRC_BUFFER_END` returns, it means that the character string specified in *utf8addr* has been interrupted in the midst of a code representing one character due to limitation by *utf8max*. In case of this error, the character code length determined from the first UTF-8 byte will return to **utf8Len*. Note that a higher value than that specified in *utf8max* will return.

If `SCE_CES_ERROR_INVALID_ENCODE` returns, it means that encoding has been determined to be invalid because the address specified in *utf8addr* stored a byte string that could not be recognized as UTF-8. The number of bytes successfully recognized as UTF-8 will return to **utf8Len* as a value between 0 and 5.

If `SCE_CES_ERROR_ILLEGAL_CODE` returns, it means that the code has been found invalid because the address specified in *utf8addr* stores codes in the U+D800 to U+DFFF range reserved as a surrogate area, or encoding with an unnecessarily large number of bytes (such as representing U+0000 as the 2-byte sequence C0,80).

If `SCE_CES_ERROR_INVALID_PROFILE` returns, it means that the address of an invalid profile has been specified in *profile*.

If `SCE_CES_ERROR_UNASSIGNED_CODE` returns, it means that the character stored in *utf8addr* was not present in the character set specified in *profile*.

In the cases of the above errors, there will be no output to *mbcBuf* and 0 will be stored in **mbcLen*.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been specified in *mbcBuf*.

The return of `SCE_CES_ERROR_DST_BUFFER_END` indicates that the multi-byte character code could not be stored because output buffer size was not sufficient.

These two error cases caused by the output buffer are only reported when no other errors are detected, and it means that the function would have been successfully completed if the output buffer had been appropriately specified. In this case, there will be no output to *mbcBuf*, but the length of the code (byte count) that was to be output will return to **mbcLen*.

SCE CONFIDENTIAL

Notes

This function is multi-thread safe.

See Also

SceCesMbcToUtf8 (), sceCesMbcToUtf8 ()

000004892117

sceCesUcs2ToMbc

Convert one character from UCS-2 to multi-byte character code

Definition

```
#include <ces.h>
int sceCesUcs2ToMbc (
    uint32_t ucs2,
    const SceCesMbcUcsProfile *profile,
    uint8_t *mbcBuf,
    uint32_t mbcMax,
    uint32_t *mbcLen
)
```

Arguments

<i>ucs2</i>	UCS-2 character code
<i>profile</i>	MBCS and UCS conversion profile address
<i>mbcBuf</i>	Address of the buffer for receiving the character code in multi-byte format
<i>mbcMax</i>	Maximum length (byte count) of the buffer for receiving the character code in multi-byte format
<i>mbcLen</i>	Address of the variable for receiving multi-byte character code length (byte count)

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal character code detected in source character code
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

This function returns a character specified in UCS-2 as an MBCS character code and the length of the character code.

Specify the UCS-2 character code in *ucs2*.

In *profile*, specify `const SceCesMbcUcsProfile*`. This can be obtained with the return value of the macro function `sceCesGetMbcUcsProfile()`, using the address of the UCS conversion profile for the character sets of each CES as the argument.

Obtain the address of the UCS conversion profile for the character sets of each CES with the following procedure.

The UCS conversion profile of multi-byte character sets can be obtained as the return value of profile initialization functions whose name begins by `sceCesUcsProfileInit`.

SCE CONFIDENTIAL

The type of the UCS conversion profile of single-byte character sets can be obtained as the return value of profile reference functions whose name begins by *sceCesRefersUcsProfile*.

Specify the address for outputting the character code in the character set indicated by the profile in *mbcBuf*.

Specify the outputtable size (byte count) of the character code in the character set indicated by the profile in *mbcMax*.

Specify the address of the variable for receiving the length (byte count) of the character code in the character set indicated by the profile in *mbcLen*. If a NULL pointer has been specified, this argument will be ignored.

If *SCE_CES_ERROR_ILLEGAL_CODE* is returned, it means the code has been determined to be illegal because a value in the range between U+D800 and U+DFFF that is reserved as surrogate area has been passed in *ucs2*.

If *SCE_CES_ERROR_INVALID_PROFILE* returns, it means that the address of an invalid profile has been specified in *profile*.

If *SCE_CES_ERROR_UNASSIGNED_CODE* returns, it means that the character specified in *ucs2* was not present in the character set specified in *profile*.

In the cases of the above errors, there will be no output to *mbcBuf* and 0 will be stored in **mbcLen*.

If *SCE_CES_ERROR_INVALID_DST_BUFFER* returns, it means that a NULL pointer has been specified in *mbcBuf*.

The return of *SCE_CES_ERROR_DST_BUFFER_END* indicates that the multi-byte character code could not be stored because output buffer size was not sufficient.

These two error cases caused by the output buffer are only reported when no other errors are detected, and it means that the function would have been successfully completed if the output buffer had been appropriately specified. In this case, there will be no output to *mbcBuf*, but the length of the code (byte count) that was to be output will return to **mbcLen*.

Notes

This function is multi-thread safe.

See Also

SceCesMbcToUcsProfile, *sceCesMbcToUcs2* ()

Functions for Handling Processing Specific to JIS Character Sets

Conversion Profiles of JIS Character Sets and UCS

SceCesJiscsUcsProfile, sceCesGetJiscsUcsProfile

Profile holding JIS character sets and UCS conversion information

Definition

```
#include <ces.h>
typedef struct SceCesJiscsUcsProfile{
    /* omitted */
} SceCesJiscsUcsProfile;

#define sceCesGetJiscsUcsProfile( profile ) /* omitted */
```

Description

SceCesJiscsUcsProfile is the type of the profile holding the conversion information of JIS character sets and UCS.

The type can be referenced as the return value of the **sceCesGetJiscsUcsProfile()** macro function through the UCS conversion profile for JIS-related CES.

SceCesSJisUcsProfile and **SceCesEucJpUcsProfile** are the JIS-related UCS conversion profiles that can be given to *profile* of **sceCesGetJiscsUcsProfile()**. Specify the address of either one of the profiles.

The address of this data type obtained with the return value is required when calling **sceCesJiscsToUcs()** and **sceCesUcsToJiscs()**.

See Also

SceCesSJisUcsProfile, **SceCesEucJpUcsProfile**, **sceCesJiscsToUcs()**, **sceCesUcsToJiscs()**

Conversion Functions of JIS Character Sets and UCS

sceCesJiscsToUcs

Convert one character from a JIS character to a UCS character

Definition

```
#include <ces.h>
int sceCesJiscsToUcs (
    const SceCesJiscsUcsProfile *profile,
    uint32_t jcode,
    uint32_t *ucode
)
```

Arguments

<i>profile</i>	Address of UCS conversion profile for JIS character sets
<i>jcode</i>	Code value indicating the JIS character
<i>ucode</i>	Address for receiving the UCS character code value

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal character code detected in source character code
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid

Description

This function converts a character of JIS character sets to a UCS character code value and returns the code value.

Specify the address of UCS conversion profile for JIS character sets to *profile*. Conversion is performed depending on the mapping information to UCS held by the referenced profile; therefore select the profile according to the purpose.

Specify the code value of JIS character sets to *jcode*. Several specification methods are available. A macro function for creating constants is provided, so specify the code value as follows.

```
// JIS X 0201 character range (single-byte character)
jcode = 0xa1;
jcode = SCE_CES_JISCS_CODE_JIS8_X0201 ( 0xa1 );
```

```
// Specify with JIS plane, row and cell (ku,ten)/(men,ku,ten)
jcode = SCE_CES_JISCS_CODE_POINT_X0208( 1, 1 );
jcode = SCE_CES_JISCS_CODE_POINT_X0212( 2, 22 );
jcode = SCE_CES_JISCS_CODE_POINT_X0213( 1, 16, 1 );

// Specify with JIS code (jiscode)/(men,jiscode)
jcode = 0x2121; //JIS X 0208 recognition
jcode = SCE_CES_JISCS_CODE_JIS7_X0208( 0x2121 );
jcode = SCE_CES_JISCS_CODE_JIS7_X0212( 0x2236 );
jcode = SCE_CES_JISCS_CODE_JIS7_X0213( 1, 0x2121 );
jcode = SCE_CES_JISCS_CODE_JIS7_X0213( 2, 0x2121 );
```

It is possible to pass the S-JIS code value when *profile* is the address that references *SceCesEucJpUcsProfile*.

```
// Specify with S-JIS code value
jcode = 0x8140;
jcode = SCE_CES_JISCS_CODE_SJIS( 0x8140 );
```

Correctly set the value that is in the range of 16-bit at the maximum and whose precedent byte is placed at the upper part.

It is possible to pass the EUC-JP code value when *profile* is the address that references *SceCesEucJpUcsProfile*.

```
// Specify with EUC-JP code value
jcode = 0xA1A1;
jcode = SCE_CES_JISCS_CODE_EUCJP( 0xA1A1 );
```

Correctly set the value that is in the range of 24bit at the maximum and whose precedent byte is placed at the upper part.

To *ucode*, specify the address of the variable that receives the UCS character code point value (32bit value).

31	24	23	16	15	8	7	0
Group	Plane		Row		Cell		
0-127(0)	0-255 (0-16)		0-255		0-255		

The code point value to be received equals to the UTF-32 encoded value; therefore no function is provided for encoding the value to Unicode encoding scheme. Use the code point value as the UTF-32 value, and convert it from UTF-32 to UTF-8 or UTF-16 as necessary.

If `SCE_CES_ERROR_INVALID_PROFILE` returns, it means that the invalid profile address is specified in *profile*.

If `SCE_CES_ERROR_INVALID_ENCODE` returns, it means that the value specified in *jcode* has been determined to be invalid because a value that cannot be recognized as an encoded value of one character of S-JIS or EUC-JP has been set in the argument.

If `SCE_CES_ERROR_ILLEGAL_CODE` returns, it is determined that an error has occurred because an inappropriate code value has been set in *jcode*.

If `SCE_CES_ERROR_UNASSIGNED_CODE` returns, it means that the character specified in *jcode* does not exist in the character set specified in *profile*.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been passed to *sjisCode*. This error is only reported when no other errors are detected, and means that the function would have been successful if the output buffer had been set appropriately.

**ucode* will be initialized with 0 when an error occurs.

Notes

This function is multi-thread safe.

SCE CONFIDENTIAL

See Also

`SceCesJiscsUcsProfile(), sceCesUcsToJiscs()`

000004892117

sceCesUcsToJiscs

Convert one character from a UCS character to a JIS character

Definition

```
#include <ces.h>
int sceCesUcsToJiscs (
    uint32_t ucode,
    const SceCesJiscsUcsProfile *profile,
    uint32_t *jcode
)
```

Arguments

<i>ucode</i>	UCS character code value
<i>profile</i>	Address of UCS conversion profile for JIS character sets
<i>jcode</i>	Address for receiving code value indicating the JIS character

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal character code detected in source character code
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid

Description

This function converts a UCS character to a character of JIS character sets and returns the code value.

Specify the code point value of the UCS character to *ucode*. (UTF-32 encoded value can be set.)

31	24	23	16	15	8	7	0
Group 0-127(0)	Plane 0-255 (0-16)		Row 0-255			Cell 0-255	

Specify the address of UCS conversion profile for JIS character sets to *profile*. Conversion is performed depending on the mapping information to UCS held by the referenced profile; therefore select the profile according to the purpose.

To *jcode*, specify the address of the variable that receives the code value indicating the JIS character. The bit structure of the code value to be received is as follows; a single-byte code value or a code point represented with the numbers of JIS plane, row and cell will return.

31	24	23	16	15	8	7	0
JIS X 0201 0x00	0x00					SBC value 0x00-0xFF	
JIS X 0208 0x02	----	0x00	Row* 1-94 (95-120)		Cell 1-94		
JIS X 0213 0x03	Plane 1, 2	Row 1-94		Cell 1-94			
JIS_X_0212 0x04	----	0x00	Row 1-94		Cell 1-94		

*Numbers in parenthesis represent the range that is valid when a profile handling external characters is specified.

Supplement: Rows that contain character definitions in JIS X 0208: 1-8, 16-84 (13, 89-92, 115-119)
 Rows that contain character definitions in the second plane of JIS X 0213: 1, 3, 4, 5, 8, 12-15, 78-94
 Rows that contain character definitions in JIS X 0212: 2, 6, 7, 9, 10, 11, 16-77

If SCE_CES_ERROR_ILLEGAL_CODE returns, it means that an inappropriate value has been passed to *ucode*.

If SCE_CES_ERROR_INVALID_PROFILE returns, it means that the address of invalid profile is specified in *profile*.

If SCE_CES_ERROR_UNASSIGNED_CODE returns, it means that the character specified in *ucode* does not exist in the character set specified in *profile*.

If SCE_CES_ERROR_INVALID_DST_BUFFER returns, it means that a NULL pointer has been passed to *jcode*. This error is only reported when no other errors are detected, and means that the function would have been successful if the output buffer had been set appropriately.

**jcode* will be initialized with 0 when an error occurs.

Notes

This function is multi-thread safe.

See Also

SceCesJiscsUcsProfile(), sceCesJiscsToUcs()

Functions for Handling JIS Characters

sceCesJisGetLevel

Retrieve the implementation level of JIS characters

Definition

```
#include <ces.h>
int sceCesJisGetLevel(
    uint8_t men,
    uint8_t ku,
    uint8_t ten,
    uint32_t *level
)
```

Arguments

<i>men</i>	Plane number
<i>ku</i>	Row number
<i>ten</i>	Cell number
<i>level</i>	Address of variable for receiving the implementation level

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified profile is invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Invalid code is specified in source
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	No character is defined to the code point

Description

This function is used to check which of the implementation levels 1 to 4 supports the character specified with JIS plane, row and cell. Moreover, if a free area with no character defined is specified, this function returns an error. The character set currently supported is JIS X 0213:2004, which includes JIS X 0208:1997 (a character set that supports the Level 1 and Level 2).

Specify the JIS plane number (1, 2) to *men*. All the characters in the JIS X 0208 range are 1.

Specify the JIS row number (1 to 94) to *ku*.

Specify the JIS cell number (1 to 94) to *ten*.

To *level*, specify the address of the variable that receives the implementation level (JIS Level 1 to Level 4) to which the character specified with plane, row and cell belongs. Any one of the values 1 to 4 will be stored if succeeded.

If SCE_CES_ERROR_ILLEGAL_CODE returns, it means that the code values of plane, row and cell specified in *men*, *ku* and *ten* have been determined to be illegal.

If SCE_CES_ERROR_UNASSIGNED_CODE returns, it is determined that no character is defined to the code point indicated with the plane, row and cell numbers specified in *men*, *ku* and *ten*. In this case, , 3 or 4 is returned to **level*.

SCE CONFIDENTIAL

If `SCE_CES_ERROR_INVALID_PARAMETER` returns, it is determined that an error has occurred because a NULL pointer has been passed to *level*.

**level* will be initialized with 0 when an error occurs, except for the cases described above.

Notes

This function is multi-thread safe.

See Also

`sceCesSJisCodeToJisX0208()`, `sceCesSJisCodeToJisX0213()`

000004892117

Functions for Handling Shift_JIS Codes

sceCesSJisGetCode

Retrieve Shift_JIS code value of one character

Definition

```
#include <ces.h>
int sceCesSJisGetCode(
    const uint8_t *sjisAddr,
    uint32_t sjisMax,
    uint32_t *sjisLen,
    uint16_t *sjisCode
)
```

Arguments

<i>sjisAddr</i>	Address that stores the Shift_JIS encoding
<i>sjisMax</i>	Maximum length (byte count) of the buffer that stores the Shift_JIS encoding
<i>sjisLen</i>	Address of variable for receiving the length (byte count) of the successfully recognized Shift_JIS character
<i>sjisCode</i>	Address of variable for receiving code value (16-bit) of the Shift_JIS character

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid

Description

This function retrieves the length (byte count) of one Shift_JIS encoding character and its Shift_JIS code value (16-bit) from the byte string stored in the specified address.

Specify the address that stores the Shift_JIS encoding to *sjisAddr*.

Specify the maximum length (byte count) of the buffer that stores the Shift_JIS encoding to *sjisMax*.

To *sjisLen*, specify the address of variable for receiving the encoding length (byte count) of the successfully recognized Shift_JIS character. If a NULL pointer is specified, this argument will be ignored.

To *sjisCode*, specify the address for receiving the code value (16-bit value) of the successfully recognized Shift_JIS character.

If SCE_CES_ERROR_INVALID_SRC_BUFFER returns, it means that a NULL pointer is passed to *sjisAddr*, or 0 is passed to *sjisMax*. In this case, 0 is stored in **sjisLen*.

SCE CONFIDENTIAL

If `SCE_CES_ERROR_SRC_BUFFER_END` returns, it means that the character specified in *sjisAddr* is interrupted at some midpoint in the code that represents one character because of the limitation of *sjisMax*. In this case, the character code length (byte count) that should have been recognized returns to **sjisLen*, and the value is greater than the value of *sjisMax*. If

`SCE_CES_ERROR_INVALID_ENCODE` returns, it means that encoding has been determined to be invalid because the byte string passed to *sjisAddr* includes unrecognizable byte values such as a value within the Shift_JIS's holding area.

If `SCE_CES_ERROR_INVALID_DST_BUFFER` returns, it means that a NULL pointer has been passed to *sjisCode*. This error is only reported when no other errors are detected, and means that the function would have been successful if the output buffer had been set appropriately.

**sjisCode* will be initialized with 0 when an error occurs.

Notes

This function is multi-thread safe.

See Also

`sceCesSJisPutCode()`, `sceCesSJisCodeToJisX0208()`, `sceCesSJisCodeToJisX0213()`

sceCesSJisPutCode

Encoding output of Shift_JIS code of one character

Definition

```
#include <ces.h>
int sceCesSJisPutCode (
    uint16_t sjisCode,
    uint8_t *sjisBuf,
    uint32_t sjisMax,
    uint32_t *sjisLen
)
```

Arguments

<i>sjisCode</i>	Shift_JIS character code value
<i>sjisBuf</i>	Address of the buffer to which the Shift_JIS encoding byte string is written
<i>sjisMax</i>	Maximum length (byte count) of the buffer to which the Shift_JIS encoding byte string is written
<i>sjisLen</i>	Address of variable for receiving the length (byte count) of the Shift_JIS encoding

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient

Description

This function outputs the character specified with a Shift_JIS character code value in the form of Shift_JIS encoding byte string and returns the length of the output.

To *sjisCode*, specify the Shift_JIS character code value.

To *sjisBuf*, specify the address to which the Shift_JIS encoding byte string is output.

To *sjisMax*, specify the size (byte count) in which the Shift_JIS encoding byte string can be output.

To *sjisLen*, specify the address of variable for receiving the length (byte count) of Shift_JIS encoding byte string. If a NULL pointer is specified, this argument will be ignored.

If SCE_CES_ERROR_INVALID_ENCODE returns, it is determined to be invalid because the code value specified to *sjisCode* will be an encoding byte string that uses an area defined as the Shift_JIS's holding area. In this case, nothing is written to *sjisBuf*, and 0 is stored in **sjisLen*.

If SCE_CES_ERROR_INVALID_DST_BUFFER returns, it means that a NULL pointer has been specified to *sjisBuf*.

If SCE_CES_ERROR_DST_BUFFER_END returns, it means that the Shift_JIS encoding byte string could not be stored because of the size shortage of output buffer.

SCE CONFIDENTIAL

If an error resulting from the output buffer occurs, nothing is output to *sjisBuf*, but the length (byte count) of the code that should have been output returns to **sjisLen*.

Notes

This function is multi-thread safe.

See Also

`sceCesSJisGetCode()`, `sceCesJisX0208ToSJisCode()`, `sceCesJisX0213ToSJisCode()`

000004892117

SCE CONFIDENTIAL

sceCesSJisCodeToJisX0208

Conversion from Shift_JIS code to JIS X 0208 row and cell

Definition

```
#include <ces.h>
int sceCesSJisCodeToJisX0208 (
    uint16_t sjisCode,
    uint8_t *ku,
    uint8_t *ten
)
```

Arguments

<i>sjisCode</i>	Shift_JIS code value
<i>ku</i>	Address of variable for receiving the row number
<i>ten</i>	Address of variable for receiving the cell number

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_OUT_OF_CODE_RANGE	0x805C0024	Character outside the representable code range of the output destination encoding scheme is detected

Description

This function converts a Shift_JIS code value to JIS X 0208 row and cell codes and then returns the code values.

Specify the Shift_JIS code value whose row and cell positions you wish to obtain to *sjisCode* in 16-bit value.

Specify the address of variable for receiving the JIS X 0208 row number to *ku*. If a NULL pointer is specified, this argument will be ignored.

Specify the address of variable for receiving the JIS X 0208 cell number to *ten*. If a NULL pointer is specified, this argument will be ignored.

This function will be terminated successfully even if the specified code value indicates a code point within a free area to which no character is defined in JIS X 0208. However, if the specified code value is a code value that can be represented with single-byte such as JIS X 0201 and ASCII, which are outside the JIS X 0208 range, this function will fail.

The values between 1 and 94 are stored in **ku* and **ten* in the case of normal termination.

If SCE_CES_ERROR_INVALID_ENCODE returns, it is determined to be invalid because the code value specified in *sjisCode* represents an encoding that cannot be used as Shift_JIS.

If SCE_CES_ERROR_OUT_OF_CODE_RANGE returns, it means that a single-byte code value or a code value within a range that has been generally extended for user-defined characters for Japanese is specified in *sjisCode*. In the case of the latter, the row and cell numbers of between 95th and 120th row, which are outside the JIS regulations, return to **ku* and **ten*.

SCE CONFIDENTIAL

**ku* and **ten* will be initialized with 0 when an error occurs, except for the cases described above.

Notes

This function is multi-thread safe.

See Also

`sceCesSJisGetCode()`, `sceCesSJisCodeToJisX0213()`, `sceCesJisGetLevel()`

000004892117

sceCesSJisCodeToJisX0213

Conversion from Shift_JIS code to JIS X 0213 plane, row and cell

Definition

```
#include <ces.h>
int sceCesSJisCodeToJisX0213 (
    uint16_t sjisCode,
    uint8_t *men,
    uint8_t *ku,
    uint8_t *ten
)
```

Arguments

<i>sjisCode</i>	Shift_JIS code value
<i>men</i>	Address of variable for receiving the plane number
<i>ku</i>	Address of variable for receiving the row number
<i>ten</i>	Address of variable for receiving the cell number

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_OUT_OF_CODE_RANGE	0x805C0024	Character outside the representable code range of the output destination encoding scheme is detected

Description

This function converts a Shift_JIS code value to JIS X 0213 plane, row and cell codes and then returns the code values.

(This function requires Shift_JIS code values of Shift_JIS2004 and cannot handle the user-defined characters for Japanese, etc. that are not incorporated in JIS X 0213. The area that has been used for *kanji* of external characters extension is recognized as the area for Level 3 and Level 4 *kanji* that are newly assigned to JIS X 0213.)

Specify the Shift_JIS code value whose plane, row and cell positions you wish to obtain to *sjisCode* in 16-bit value.

Specify the address of variable for receiving the JIS X 0213 plane number (1, 2) to *men*.

Specify the address of variable for receiving the JIS X 0213 row number (1 to 94) to *ku*.

Specify the address of variable for receiving the JIS X 0213 cell number (1 to 94) to *ten*.

This function will be terminated successfully even if the specified Shift_JIS code value indicates a code point within a free area to which no character is defined for JIS X 0213. However, if the specified code value is a code value that can be represented with single-byte such as JIS X 0201 and ASCII, which are outside the JIS X 0213 range, this function will fail.

If SCE_CES_ERROR_INVALID_ENCODE returns, it is determined to be invalid because the code value specified in *sjisCode* indicates an encoding that uses an area defined as the Shift_JIS's holding area.

SCE CONFIDENTIAL

If `SCE_CES_ERROR_OUT_OF_CODE_RANGE` returns, it is determined to be out of the conversion range because a single-byte code value is specified in *sjisCode*.

**men*, **ku* and **ten* will be initialized with 0 when an error occurs.

Notes

This function is multi-thread safe.

See Also

`sceCesSJisGetCode()`, `sceCesSJisCodeToJisX0208()`, `sceCesJisGetLevel()`

000004892117

SCE CONFIDENTIAL

sceCesJisX0208ToSJisCode

Conversion from JIS X 0208 row and cell to Shift_JIS code

Definition

```
#include <ces.h>
int sceCesJisX0208ToSJisCode (
    uint8_t ku,
    uint8_t ten,
    uint16_t *sjisCode
)
```

Arguments

<i>ku</i>	Row number
<i>ten</i>	Cell number
<i>sjisCode</i>	Address of variable for receiving the Shift_JIS code value

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Invalid code is specified in source

Description

This function receives a JIS X 0208 character in the form of row and cell codes and returns a code value after converting the received codes to a Shift_JIS code value.

Specify the JIS X 0208 row number (1 to 94) to *ku*.

Specify the JIS X 0208 cell number (1 to 94) to *ten*.

Specify the address of variable for receiving the Shift_JIS code value (16-bit value) to *sjisCode*. If a NULL pointer is specified, this argument will be ignored.

If SCE_CES_ERROR_ILLEGAL_CODE returns, row and cell code values specified in *ku* and *ten* are determined to be illegal. In the case that the code value specified in *ku* is any one of the numbers between 95 and 120 and this error occurs, a code value taking the external characters extension into account will be stored in **sjisCode*.

**sjisCode* will be initialized with 0 when an error occurs, except for the cases described above.

Notes

This function is multi-thread safe.

See Also

sceCesSJisCodeToJisX0208(), sceCesSJisPutCode()

SCE CONFIDENTIAL

sceCesJisX0213ToSJisCode

Conversion from JIS X 0213 plane, row and cell to Shift_JIS code

Definition

```
#include <ces.h>
int sceCesJisX0213ToSJisCode (
    uint8_t men,
    uint8_t ku,
    uint8_t ten,
    uint16_t *sjisCode
)
```

Arguments

<i>men</i>	Plane number
<i>ku</i>	Row number
<i>ten</i>	Cell number
<i>sjisCode</i>	Address of variable for receiving the Shift_JIS code value

Return Values

Returns SCE_OK (0) as the value of the function for success.

Returns one of the following error codes (negative value) for errors.

Value	Hexadecimal	Description
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Invalid code is specified in source
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined

Description

This function receives a JIS X 0213 character in the form of plane, row and cell codes and returns a code value after converting the received codes to a Shift_JIS code value.

(This function stores Shift_JIS code values of Shift_JIS2004, and thus there is a possibility that this function cannot handle the Shift_JIS code processing for JIS X 0208 and user-defined characters for Japanese. The area that has been used for *kanji* of external characters extension is used as the area for Level 3 and Level 4 *kanji* that are newly assigned to JIS X 0213.)

Specify the JIS X 0213 plane number (1, 2) to *men*.

Specify the JIS X 0213 row number (1 to 94) to *ku*.

Specify the JIS X 0213 cell number (1 to 94) to *ten*.

Specify the address of variable for receiving the Shift_JIS code value (16-bit value) to *sjisCode*. If a NULL pointer is specified, this argument will be ignored.

If SCE_CES_ERROR_ILLEGAL_CODE returns, plane, row and cell code values specified in *men*, *ku* and *ten* are determined to be illegal.

If SCE_CES_ERROR_UNASSIGNED_CODE returns, it is determined that an error has occurred because the code point indicated with the plane, row and cell numbers specified in *men*, *ku* and *ten* is not supported in Shift_JIS2004 (undefined area in the second plane).

**sjisCode* will be initialized with 0 when an error occurs.

SCE CONFIDENTIAL

Notes

This function is multi-thread safe.

See Also

`sceCesSJisCodeToJisX0213()`, `sceCesSJisPutCode()`

000004892117

Constants

000004892117

SCE CONFIDENTIAL

SceCesEndianParam

enum constant indicating CES endian processing method

Definition

```
#include <ces.h>
typedef enum SceCesEndianParam {
    (omitted; see below)
} SceCesEndianParam;
```

Description

Constant indicating CES endianness

Value	Description
SCE_CES_ENDIAN_BE	Big-endian
SCE_CES_ENDIAN_LE	Little-endian
SCE_CES_ENDIAN_SYS	Same endianness as the system selected
SCE_CES_ENDIAN_STAY	Current status is maintained

See Also

`sceCesSetUtf16StrEndian()`, `sceCesSetUtf32StrEndian()`

Return Codes

List of the return codes returned by the functions of the CES library

Definition

Value	Hexadecimal	Description
SCE_CES_OK	0x00000000	Success
SCE_CES_ERROR_INVALID_PARAMETER	0x805C0001	Specified argument value is invalid
SCE_CES_ERROR_INVALID_PROFILE	0x805C0004	Specified profile is invalid
SCE_CES_ERROR_INVALID_SRC_BUFFER	0x805C0010	Specified source buffer is invalid
SCE_CES_ERROR_SRC_BUFFER_END	0x805C0011	Specified source buffer is insufficient
SCE_CES_ERROR_INVALID_ENCODE	0x805C0014	Source encoding determined to be invalid
SCE_CES_ERROR_ILLEGAL_CODE	0x805C0015	Illegal character code detected in source character code
SCE_CES_ERROR_UNASSIGNED_CODE	0x805C0020	Code points in output destination encoding scheme are not defined
SCE_CES_ERROR_OUT_OF_CODE_RANGE	0x805C0024	Character outside the representable code range of the output destination encoding scheme is detected
SCE_CES_ERROR_INVALID_DST_BUFFER	0x805C0030	Output destination buffer is invalid
SCE_CES_ERROR_DST_BUFFER_END	0x805C0031	Output destination buffer is insufficient