Common pascal units documentation

Pasdoc

June 20, 2004

Contents

Chapter 1

Unit crc

1.1 Description

CRC generation unit

CRC generation routines, compatible with ISO 3309 and ITU-T-V42.

1.2 Overview

`UpdateCrc32`

1.3 Functions and Procedures

UpdateCrc32 function

Declaration `function UpdateCrc32(InitCrc:longword; b:  byte):longword;`

Description Routine to get the CRC-32 value. Normally to be compatible with the ISO 3309 standard, the first call to this routine should set `InitCRC` to `$FFFFFFFF`, and the final result of the CRC-32 should be `XOR`'ed with $FFFFFFFF.

Parameters **InitCRC** The value of the previous CRC

b The data byte to get the CRC-32 of

Returns The updated CRC-32 value

1.4 Author

Carl Eric Codere

Chapter 2

Unit dpautils

2.1 Description

Delphi/Kylix compatbility unit

This unit includes common definitions so that common code can be compiled under the Delphicompilers. It supports Delphi 6 and higher that are targeted for Win32 as well as WDOSX/DOS.

2.2 Author

Carl Eric Codere

Chapter 3

Unit fpautils

3.1 Description

Free Pascal compatibility unit

This unit includes common definitions so that common code can be compiled under the Free pascal compilers. It supports Freepascal 1.0.6 and higher (all targets).

3.2 Author

Carl Eric Codere

Chapter 4

Unit ietf

4.1 Description

ietf/web related support unit

This unit contains routines to validate strings, and characters according to different IETF standards (such as URL's, URI's and MIME types).

4.2 Overview

`urn_isvalid` Verifies the validity of a complete URN string

`urn_isvalidnid`

`urn_split` Splits an URN string in its separate components

4.3 Functions and Procedures

urn_isvalid function

Declaration `function urn_isvalid(s: shortstring): boolean;`

Description Verifies the validity of a complete URN string

This checks the conformance of the URN address. It is based on IETF RFC 2141.

Returns TRUE if this is a valid URN string

urn_isvalidnid function

Declaration `function urn_isvalidnid(nid: string): boolean;`

Description This routine checks that the specified NID (namespace) is either registered to IANA, or that it is an experimental NID, as described in IETF RFC 2611. More assignment information can be obtained from:

Returns TRUE if this is a registered or experimental NID string

urn_split function

Declaration `function urn_split(urn:string; var urnidstr,nidstr,nssstr: string): boolean;`

Description Splits an URN string in its separate components

It is based on IETF RFC 2141. nidstr Namespace identifier NID

Parameters

urn Complete URN string to separate

urnidstr Signature URN:

nssstr Namespace specific string NSS

Returns TRUE if the operation was successfull, or FALSE if the URN is malformed

4.4 Author

Carl Eric Codere

Chapter 5

Unit locale

5.1 Description

Localisation unit

This unit is used to convert different locale information. ISO Standards are used where appropriate. Credits where credits are due, information on the ISO and date formats where taken from

5.2 Overview

`GetCharEncoding`

`GetISODateString`

`GetISOTimeString`

`IsValidISODateString` Verifies if the date is in a valid ISO 8601 format

`IsValidISODateTimeString` Verifies if the date and time is in a valid ISO 8601 format

`IsValidISOTimeString` Verifies if the time is in a valid ISO 8601 format

`UNIXToDateTime`

5.3 Functions and Procedures

GetCharEncoding function

Declaration `function GetCharEncoding(alias: string; var name: string): integer;`

Description Using a registered ALIAS name for a specific character encoding, return the common or MIME name associated with this character set, and indicate the type of stream format used. The type of stream format used can be one of the `CHAR_ENCODING_XXXX` constants.

GetISODateString function

Declaration `function GetISODateString(Year, Month, Day: Word): shortstring;`

Description Returns the preferred date string as recommended by ISO 8601 (Gregorian Calendar). Returns an empty string if there is an error.

Parameters **year** Year of the date - valid values are from 0000 to 9999

month Month of the date - valid values are from 0 to 12

day Day of the month - valid values are from 1 to 31

GetISOTimeString function

Declaration `function GetISOTimeString(Hour, Minute, Second: Word; UTC: Boolean): shortstring;`

Description Returns the preferred time string as recommended by ISO 8601 (Gregorian Calendar). .

Returns Empty string if there is an error

IsValidISODateString function

Declaration `function IsValidISODateString(datestr: shortstring): boolean;`

Description Verifies if the date is in a valid ISO 8601 format

Parameters **datestr** Date string in valid ISO 8601 format

Returns TRUE if the date string is valid otherwise false

IsValidISODateTimeString function

Declaration `function IsValidISODateTimeString(str: shortstring): boolean;`

Description Verifies if the date and time is in a valid ISO 8601 format

Currently does not support the fractional second parameters, and only the format recommended by W3C when used with the time zone designator.

Parameters **str** Date-Time string in valid ISO 8601 format

Returns TRUE if the date-time string is valid otherwise false

IsValidISOTimeString function

Declaration `function IsValidISOTimeString(timestr: shortstring): boolean;`

Description Verifies if the time is in a valid ISO 8601 format

Currently does not support the fractional second parameters, and only the format recommended by W3C when used with the time zone designator.

Parameters **timestr** Time string in valid ISO 8601 format

Returns TRUE if the time string is valid otherwise false

UNIXToDateTime procedure

Declaration `procedure UNIXToDateTime(epoch: longword; var year, month, day, hour, minute, second: Word);`

Description Converts a UNIX styled time (the number of seconds since 1970) to a standard date and time representation.

5.4 Constants

CHAR_ENCODING_UTF8

Declaration `CHAR_ENCODING_UTF8 = 0;`

Description Character encoding value: UTF-8 storage format

CHAR_ENCODING_UNKNOWN

Declaration `CHAR_ENCODING_UNKNOWN = -1;`

Description Character encoding value: unknown format

CHAR_ENCODING_UTF32BE

Declaration `CHAR_ENCODING_UTF32BE = 1;`

Description Character encoding value: UTF-32 Big endian

CHAR_ENCODING_UTF32LE

Declaration `CHAR_ENCODING_UTF32LE = 2;`

Description Character encoding value: UTF-32 Little endian

CHAR_ENCODING_UTF16LE

Declaration `CHAR_ENCODING_UTF16LE = 3;`

Description Character encoding value: UTF-16 Little endian

CHAR_ENCODING_UTF16BE

Declaration `CHAR_ENCODING_UTF16BE = 4;`

Description Character encoding value: UTF-16 Big endian

CHAR_ENCODING_BYTE

Declaration `CHAR_ENCODING_BYTE = 5;`

Description Character encoding value: One byte per character storage format

CHAR_ENCODING_UTF16

Declaration `CHAR_ENCODING_UTF16 = 6;`

Description Character encoding value: UTF-16 unknown endian (determined by BOM)

CHAR_ENCODING_UTF32

Declaration `CHAR_ENCODING_UTF32 = 7;`

Description Character encoding value: UTF-32 unknown endian (determined by BOM)

5.5 Author

Carl Eric Codere

Chapter 6

Unit tpautils

6.1 Description

Turbo Pascal 7 Compatibility unit

This unit includes common definitions so that common code can be compiled under the Turbo/Borland pascal compilers. It supports both Turbo Pascal 7.0 and Borland Pascal 7.0 and higher.

6.2 Author

Carl Eric Codere

Chapter 7

Unit unicode

7.1 Description

unicode support unit

This unit contains routines to convert between the different unicode encoding schemes. All UNICODE/ISO 10646 strings are limited to 255 characters. Since all these encoding are variable length, except the UTF-32 (which is equivalent to UCS-4 according to ISO 10646:2003) and UCS-2 encoding, to parse through characters, every string should be converted to UTF-32 or UCS-2 before being used.

7.2 Overview

`ConvertFromUTF32` Convert an UTF-32 string to a single byte encoded string

`ConvertToUTF32` Convert a byte encoded string to an UTF-32 string

`ConvertUCS2ToUTF32` Convert an UCS-2 string to an UTF-32 string

`ConvertUTF16ToUTF32` Convert an UTF-16 string to an UTF-32 string

`ConvertUTF32ToUCS2` Convert an UTF-32 string to an UCS-2 string

`ConvertUTF32toUTF16` Convert an UTF-32 string to an UTF-16 string

`convertUTF32toUTF8` Convert an UTF-32 string to an UTF-8 string

`ConvertUTF8ToUTF32` Convert an UTF-8 string to an UTF-32 string

`lengthUTF16` Returns the current length of an UTF-16 string

`lengthutf8` Returns the current length of an UTF-8 string

`setlengthUTF16` Set the length of an UTF-16 string

`setlengthUTF8` Set the length of an UTF-8 string `utf16_sizeencoding` Returns the number of characters that are used to encode this character

`utf32_concat` Concatenates two UTF-32 strings, and gives a resulting UTF-32 string

`utf32_concatascii` Concatenates an UTF-32 string with an ASCII string, and gives a resulting UTF-32 string

`utf32_copy` Returns an utf-32 substring of an utf-32 string

`utf32_delete` Deletes a substring from a string

`utf32_equal` Checks if both UTF-32 strings are equal

`utf32_equalascii` Checks if an ASCII string is equal to an UTF-32 string

`utf32_issupported` Checks if conversion from/to this character set format to/from UTF-32 is supported

`utf32_isvalid` Checks if the UTF-32 character is valid

`utf32_iswhitespace` Determines if the specified character is a whitespace character

`utf32_length` Returns the current length of an UTF-32 string

`utf32_pos` Searches for an UTF-32 substring in an UTF-32 string

`utf32_posascii` Searches for an ASCII substring in an UTF-32 string

`utf32_setlength` Set the new dynamic length of an utf-32 string

`utf32_trimleft` Trims leading spaces and control characters from an UTF-32 string.

`utf32_trimright` Trims trailing spaces and control characters from an UTF-32 string.

`utf8_sizeencoding` Returns the number of characters that are used to encode this character

7.3 Functions and Procedures

ConvertFromUTF32 function

Declaration `function ConvertFromUTF32(source: utf32string; var dest: shortstring; desttype: string): integer;`

Description Convert an UTF-32 string to a single byte encoded string

This routine converts an UTF-32 string stored in native byte order (native endian) to a single-byte encoded string. The string is limited to 255 characters, and if the conversion cannot be successfully be completed, it gives out an error. The following `desttype` can be specified: ISO-8859-1, windows-1252, ISO-8859-2, ISO-8859-5, ISO-8859-16, macintosh, atari, cp437, cp850, ASCII.

Parameters **desttype** Indicates the single byte encoding scheme

Returns `UNICODE_ERR_OK`(7.5) if there was no error in the conversion

ConvertToUTF32 function

Declaration `function ConvertToUTF32(source: shortstring; var dest: utf32string; srctype: string): integer;`

Description Convert a byte encoded string to an UTF-32 string

This routine converts a single byte encoded string to an UTF-32 string stored in native byte order The string is limited to 255 characters, and if the conversion cannot be successfully be completed, it gives out an error. The following `srctype` can be specified: ISO-8859-1, windows-1252, ISO-8859-2, ISO-8859-5, ISO-8859-16, macintosh, atari, cp437, cp850, ASCII.

Parameters **srctype** Indicates the single byte encoding scheme

Returns `UNICODE_ERR_OK`(7.5) if there was no error in the conversion

ConvertUCS2ToUTF32 function

Declaration `function ConvertUCS2ToUTF32(src: array of ucs2; var dst: utf32string): integer;`

Description Convert an UCS-2 string to an UTF-32 string

This routine converts an UCS-2 string to an UTF-32 string that is stored in native byte order (big-endian). If some characters could not be converted an error will be reported.

Parameters **src** Either a single ucs-2 character or a complete ucs-2 string

dest Resulting UTF-32 coded string

Returns `UNICODE_ERR_OK`(7.5) if there was no error in the conversion

ConvertUTF16ToUTF32 function

Declaration `function ConvertUTF16ToUTF32(src: utf16string; var dst: utf32string): integer;`

Description Convert an UTF-16 string to an UTF-32 string

This routine converts an UTF-16 string to an UTF-32 string. Both strings must be stored in native byte order (native endian).

Returns `UNICODE_ERR_OK`(7.5) if there was no error in the conversion

ConvertUTF32ToUCS2 function

Declaration `function ConvertUTF32ToUCS2(src: array of utf32; var dst: ucs2string): integer;`

Description Convert an UTF-32 string to an UCS-2 string

This routine converts an UTF-32 string to an UCS-2 string that is stored in native byte order. If some characters could not be converted an error will be reported.

Parameters **src** Either a single utf-32 character or a complete utf-32 string

dest Resulting UCS-2 coded string

Returns `UNICODE_ERR_OK`(7.5) if there was no error in the conversion

ConvertUTF32toUTF16 function

Declaration `function ConvertUTF32toUTF16(src: array of utf32; var dest: utf16string): integer;`

Description Convert an UTF-32 string to an UTF-16 string

This routine converts an UTF-32 string to an UTF-16 string. Both strings must be stored in native byte order (native endian).

Parameters **src** Either a single utf-32 character or a complete utf-32 string

dest Resulting UTF-16 coded string

Returns `UNICODE_ERR_OK`(7.5) if there was no error in the conversion

convertUTF32toUTF8 function

Declaration `function convertUTF32toUTF8(s: array of utf32; var outstr: utf8string): integer;`

Description Convert an UTF-32 string to an UTF-8 string

Converts an UTF-32 string or character in native endian to an UTF-8 string.

Parameters **s** Either a single utf-32 character or a complete utf-32 string

outstr Resulting UTF-8 coded string

Returns `UNICODE_ERR_OK`(7.5) if there was no error in the conversion

ConvertUTF8ToUTF32 function

Declaration `function ConvertUTF8ToUTF32(src: utf8string; var dst: utf32string): integer;`

Description Convert an UTF-8 string to an UTF-32 string

This routine converts an UTF-8 string to an UTF-32 string that is stored in native byte order.

Returns `UNICODE_ERR_OK`(7.5) if there was no error in the conversion

lengthUTF16 function

Declaration `function lengthUTF16(s: array of utf16): integer;`

Description Returns the current length of an UTF-16 string

lengthutf8 function

Declaration `function lengthutf8(s: array of utf8): integer;`

Description Returns the current length of an UTF-8 string

setlengthUTF16 procedure

Declaration `procedure setlengthUTF16(var s: array of utf16; l: integer);`

Description Set the length of an UTF-16 string

setlengthUTF8 procedure

Declaration `procedure setlengthUTF8(var s: array of utf8; l: integer);`

Description Set the length of an UTF-8 string

utf16_sizeencoding function

Declaration `function utf16_sizeencoding(c: utf16): integer;`

Description Returns the number of characters that are used to encode this character

. Actually checks if this is a high-surrogate value, if not returns 1, indicating that the character is encoded a single `utf16` character, otherwise returns 2, indicating that 1 one other `utf16` character is required to encode this data.

utf32_concat procedure

Declaration `procedure utf32_concat(var resultstr: utf32string;s1: utf32string; s2: array of utf32);`

Description Concatenates two UTF-32 strings, and gives a resulting UTF-32 string

utf32_concatascii procedure

Declaration `procedure utf32_concatascii(var resultstr: utf32string;s1: utf32string; s2: shortstring);`

Description Concatenates an UTF-32 string with an ASCII string, and gives a resulting UTF-32 string

utf32_copy procedure

Declaration `procedure utf32_copy(var resultstr: utf32string; s: array of utf32; index: integer; count: integer);`

Description Returns an utf-32 substring of an utf-32 string

utf32_delete procedure

Declaration `procedure utf32_delete(var s: utf32string; index: integer; count: integer);`

Description Deletes a substring from a string

utf32_equal function

Declaration `function utf32_equal(const s1,s2: utf32string): boolean;`

Description Checks if both UTF-32 strings are equal

utf32_equalascii function

Declaration `function utf32_equalascii(s1 : array of utf32; s2: shortstring): boolean;`

Description Checks if an ASCII string is equal to an UTF-32 string

utf32_issupported function

Declaration `function utf32_issupported(s: string): boolean;`

Description Checks if conversion from/to this character set format to/from UTF-32 is supported

Parameters **s** This is an alias for a character set, as defined by IANA

Returns true if conversion to/from UTF-32 is supported with this character set, otherwise FALSE

utf32_isvalid function

Declaration `function utf32_isvalid(c: utf32): boolean;`

Description Checks if the UTF-32 character is valid

This routine verifies if the UTF-32 character is within the valid ranges of UTF-32 characters, as specified in the Unicode standard 4.0. BOM characters are NOT valid with this routine.

utf32_iswhitespace function

Declaration `function utf32_iswhitespace(c: utf32): boolean;`

Description Determines if the specified character is a whitespace character

utf32_length function

Declaration `function utf32_length(s: array of utf32): integer;`

Description Returns the current length of an UTF-32 string

utf32_pos function

Declaration `function utf32_pos(substr: utf32string;s : utf32string): integer;`

Description Searches for an UTF-32 substring in an UTF-32 string

utf32_posascii function

Declaration `function utf32_posascii(substr: shortstring; s: utf32string): integer;`

Description Searches for an ASCII substring in an UTF-32 string

utf32_setlength procedure

Declaration `procedure utf32_setlength(var s: array of utf32; l: integer);`

Description Set the new dynamic length of an utf-32 string

utf32_trimleft procedure

Declaration `procedure utf32_trimleft(var s: utf32string);`

Description Trims leading spaces and control characters from an UTF-32 string.

utf32_trimright procedure

Declaration `procedure utf32_trimright(var s: utf32string);`

Description Trims trailing spaces and control characters from an UTF-32 string.

utf8_sizeencoding function

Declaration `function utf8_sizeencoding(c: utf8): integer;`

Description Returns the number of characters that are used to encode this character

.

7.4 Types

utf8

Declaration `utf8 = char;`

Description UTF-8 base data type

utf16

Declaration `utf16 = word;`

Description UTF-16 base data type

utf32

Declaration `utf32 = longword;`

Description UTF-32 base data type

ucs2

Declaration `ucs2 = word;`

Description UCS-2 base data type

ucs2string

Declaration `ucs2string = array[0..255] of ucs2;`

Description UCS-2 string declaration. Index 0 contains the active length of the string in characters.

utf32string

Declaration `utf32string = array[0..255] of utf32;`

Description UTF-32 string declaration. Index 0 contains the active length of the string in characters.

utf8string

Declaration `utf8string = array[0..1024] of utf8;`

Description UTF-8 string declaration. Index 0 contains the active length of the string in BYTES

utf16string

Declaration `utf16string = array[0..255] of utf16;`

Description UTF-16 string declaration. Index 0 contains the active length of the string in BYTES

7.5 Constants

UNICODE_ERR_OK

Declaration `UNICODE_ERR_OK = 0;`

Description Return status: conversion successful

UNICODE_ERR_SOURCEILLEGAL

Declaration `UNICODE_ERR_SOURCEILLEGAL = -1;`

Description Return status: source sequence is illegal/malformed

UNICODE_ERR_LENGTH_EXCEED

Declaration `UNICODE_ERR_LENGTH_EXCEED = -2;`

Description Return status: Target space excedeed

UNICODE_ERR_INCOMPLETE_CONVERSION

Declaration `UNICODE_ERR_INCOMPLETE_CONVERSION = -3;`

Description Return status: Some charactrer could not be successfully converted to this format

UNICODE_ERR_NOTFOUND

Declaration `UNICODE_ERR_NOTFOUND = -4;`

Description Return status: The character set is not found

BOM_UTF8

Declaration `BOM_UTF8 = #$EF#$BB#$BF;`

Description Byte order mark: UTF-8 encoding signature

BOM_UTF32_BE

Declaration `BOM_UTF32_BE = #00#00#$FE#$FF;`

Description Byte order mark: UTF-32 big endian encoding signature

BOM_UTF32_LE

Declaration `BOM_UTF32_LE = #$FF#$FE#00#00;`

Description Byte order mark: UTF-32 little endian encoding signature

7.6 Author

Carl Eric Codere

Chapter 8

Unit utils

8.1 Description

General utilities common to all platforms.

8.2 Overview

`boolstr`	Convert a boolean value to an ASCII representation
`decstr`	Convert a value to an ASCII decimal representation
`EscapeToPascal`	
`FileExists`	Verifies the existence of a filename
`hexstr`	Convert a value to an ASCII hexadecimal representation
`Printf`	Format a string and print it out to the console
`StreamErrorProcedure`	Generic stream error procedure
`SwapLong`	Change the endian of a 32-bit value
`SwapWord`	Change the endian of a 16-bit value
`TrimLeft`	Remove all whitespace from the start of a string
`TrimRight`	Remove all whitespace from the end of a string
`UpString`	Convert a string to uppercase ASCII
`ValBinary`	
`ValDecimal`	
`ValHexadecimal`	
`ValOctal`	

8.3 Functions and Procedures

boolstr function

Declaration `function boolstr(val: boolean; cnt: byte): string;`

Description Convert a boolean value to an ASCII representation

To avoid left padding with spaces, set `cnt` to zero.

decstr function

Declaration `function decstr(val : longint;cnt : byte) : string;`

Description Convert a value to an ASCII decimal representation

To avoid left padding with zeros, set `cnt` to zero.

EscapeToPascal function

Declaration `function EscapeToPascal(const s:string; var code: integer): string;`

Description Converts a C style string (containing escape characters), to a pascal style string. Returns the converted string. If there is no error in the conversion, `code` will be equal to zero.

Parameters **s** String to convert

code Result of operation, 0 when there is no error

FileExists function

Declaration `Function FileExists(FName : string): Boolean;`

Description Verifies the existence of a filename

This routine verifies if the file named can be opened or if it actually exists. FName Name of the file to check Returns FALSE if the file cannot be opened or if it does not exist.

hexstr function

Declaration `function hexstr(val : longint;cnt : byte) : string;`

Description Convert a value to an ASCII hexadecimal representation

Printf function

Declaration `function Printf(const s : string; var Buf; size : word): string;`

Description Format a string and print it out to the console

This routine formats the string specified in s to the format specified and returns the resulting string. The following specifiers are allowed: %d : The buffer contents contains an integer %s : The buffer contents contains a string, terminated by a null character. %bh : The buffer contents contains a byte coded in BCD format, only the high byte will be kept. %bl : The buffer contents contains a byte coded in BCD format, only the low byte will be kept. s The string to format, with format specifiers buf The buffer containing the data size The size of the data in the buffer Returns The resulting formatted string

StreamErrorProcedure procedure

Declaration `procedure StreamErrorProcedure(Var S: TStream);`

Description Generic stream error procedure

Generic stream error procedure that can be used to set `streamerror`

SwapLong procedure

Declaration `Procedure SwapLong(var x :  longword);`

Description Change the endian of a 32-bit value

SwapWord procedure

Declaration `Procedure SwapWord(var x :  word);`

Description Change the endian of a 16-bit value

TrimLeft function

Declaration `function TrimLeft(const S: string):  string;`

Description Remove all whitespace from the start of a string

TrimRight function

Declaration `function TrimRight(const S: string):  string;`

Description Remove all whitespace from the end of a string

UpString function

Declaration `function UpString(s :  string):  string;`

Description Convert a string to uppercase ASCII

ValBinary function

Declaration `function ValBinary(const S:String; var code:  integer):longint;`

Description Convert a binary value represented by a string to its numerical value. If there is no error, `code` will be equal to zero.

ValDecimal function

Declaration `function ValDecimal(const S:String; var code:  integer):longint;`

Description Convert a decimal value represented by a string to its numerical value. If there is no error, `code` will be equal to zero.

ValHexadecimal function

Declaration `function ValHexadecimal(const S:String; var code:  integer):longint;`

Description Convert an hexadecimal value represented by a string to its numerical value. If there is no error, `code` will be equal to zero.

ValOctal function

Declaration `function ValOctal(const S:String;var code:  integer):longint;`

Description Convert an octal value represented by a string to its numerical value. If there is no error, `code` will be equal to zero.

8.4 Author

Carl Eric Codere

Chapter 9

Unit vpautils

9.1 Description

Virtual Pascal Compatibility unit

This unit includes common definitions so that common code can be compiled under the Virtual pascal compiler. It supports Virtual Pascal 2.1 and higher for the Win32, DOS and OS/2 targets.

9.2 Author

Carl Eric Codere