Common pascal units documentation

Pasdoc

November 23, 2011

Contents

Chapter 1

Unit collects

1.1 Description

Collection units

This unit contains collection and utility objects, being quite similar to those included in the objects and classes units. The only difference being that they compile on all compiler targets.

1.2 Overview

`TStack Object` Stack object

`TExtendedCollection Object` Base collection object

`TExtendedSortedCollection Object` Base sorted collection object

`TExtendedStringCollection Object` String pointer collection object

`TExtendedSortedStringCollection Object` Sorted string pointer collection object

`TLongintCollection Object` Collection of numeric longint values.

`THashItem Object` Basic hash item that stores date for use in THashTable.

`TStringHashItem Object` Simple String hash item that stores strings for use in THashTable.

`THashTable Object` Hash table that is used to store and retrieve key-value pairs.

1.3 Classes, Interfaces, Objects and Records

TStack Object

Hierarchy

TStack > TObject

Description

Stack object

Implement an object that is used as a LIFO stack containing pointers as data.

Methods

init

Declaration `public constructor init;`

done

Declaration `public destructor done;`

isEmpty

Declaration `public function isEmpty: boolean;`

Description Returns true if the stack is empty.

peek

Declaration `public function peek: pointer;`

Description Peek at the element on top of the stack

pop

Declaration `public function pop: pointer;`

Description Pop the last element pushed from the stack

push

Declaration `public procedure push(p: pointer);`

Description Push the specified data on the stack.

TExtendedCollection Object

Hierarchy

TExtendedCollection > TObject

Description

Base collection object

Fields

Count `public Count: Integer;`

Delta `public Delta: Integer;`

Items `public Items: PItemList;`

Limit `public Limit: Integer;`

Methods

Init

Declaration `public CONSTRUCTOR Init (ALimit, ADelta: Integer);`

Done

Declaration `public DESTRUCTOR Done; Virtual;`

At

Declaration `public FUNCTION At (Index: Integer): Pointer;`

FirstThat

Declaration `public FUNCTION FirstThat (Test: Pointer): Pointer;`

IndexOf

Declaration `public FUNCTION IndexOf (Item: Pointer): Integer; Virtual;`

LastThat

Declaration `public FUNCTION LastThat (Test: Pointer): Pointer;`

AtDelete

Declaration `public PROCEDURE AtDelete (Index: Integer);`

AtFree

Declaration `public PROCEDURE AtFree (Index: Integer);`

AtInsert

Declaration `public PROCEDURE AtInsert (Index: Integer; Item: Pointer);`

AtPut

Declaration `public PROCEDURE AtPut (Index:  Integer; Item:  Pointer);`

Delete

Declaration `public PROCEDURE Delete (Item:  Pointer);`

DeleteAll

Declaration `public PROCEDURE DeleteAll;`

Error

Declaration `public PROCEDURE Error (Code, Info:  Integer); Virtual;`

ForEach

Declaration `public PROCEDURE ForEach (Action:  Pointer);`

Free

Declaration `public PROCEDURE Free (Item:  Pointer);`

FreeAll

Declaration `public PROCEDURE FreeAll;`

FreeItem

Declaration `public PROCEDURE FreeItem (Item:  Pointer); Virtual;`

Insert

Declaration `public PROCEDURE Insert (Item:  Pointer); Virtual;`

Pack

Declaration `public PROCEDURE Pack;`

SetLimit

Declaration `public PROCEDURE SetLimit (ALimit:  Integer); Virtual;`

TExtendedSortedCollection Object

Hierarchy

TExtendedSortedCollection > `TExtendedCollection`(1.3) > TObject

Description

Base sorted collection object

Fields

Duplicates `public Duplicates: Boolean;`

Methods

Init

Declaration `public CONSTRUCTOR Init (ALimit, ADelta: Integer);`

Compare

Declaration `public FUNCTION Compare (Key1, Key2: Pointer): Integer; Virtual;`

IndexOf

Declaration `public FUNCTION IndexOf (Item: Pointer): Integer; Virtual;`

KeyOf

Declaration `public FUNCTION KeyOf (Item: Pointer): Pointer; Virtual;`

Search

Declaration
```
public FUNCTION Search (Key: Pointer; Var Index: Integer): Boolean;
Virtual;
```

Insert

Declaration `public PROCEDURE Insert (Item: Pointer); Virtual;`

TExtendedStringCollection Object

Hierarchy

TExtendedStringCollection > `TExtendedCollection`(1.3) > TObject

Description

String pointer collection object

This collection accepts pointers to shortstrings as input. The data is not sorted.

Methods

FreeItem

Declaration `public PROCEDURE FreeItem (Item:  Pointer); Virtual;`

TExtendedSortedStringCollection Object

Hierarchy

TExtendedSortedStringCollection > `TExtendedSortedCollection`(1.3) > `TExtendedCollection`(1.3) > TObject

Description

Sorted string pointer collection object

This collection accepts pointers to shortstrings as input. The data is sorted as it is added in.

Methods

Init

Declaration `public CONSTRUCTOR Init (ALimit, ADelta:  Integer);`

Compare

Declaration `public FUNCTION Compare (Key1, Key2:  Pointer):  Integer; Virtual;`

FreeItem

Declaration `public PROCEDURE FreeItem (Item:  Pointer); Virtual;`

TLongintCollection Object

Hierarchy

TLongintCollection > `TExtendedCollection`(1.3) > TObject

Description

Collection of numeric longint values.

This collection is a collection of longint values instead of pointers, so instead of pointers you pass it directly the value to store in the collection.

Methods

FreeItem

Declaration `public procedure FreeItem(Item: pointer); virtual;`

THashItem Object

Hierarchy

THashItem > TObject

Description

Basic hash item that stores date for use in THashTable.

Methods

Init

Declaration `public Constructor Init;`

Done

Declaration `public Destructor Done; virtual;`

toString

Declaration `public Function toString: string; virtual;`

Description Returns the string representation of this item

TStringHashItem Object

Hierarchy

TStringHashItem > `THashItem`(1.3) > TObject

Description

Simple String hash item that stores strings for use in THashTable.

Methods

Init

Declaration `public Constructor Init(s: String);`

Done

Declaration `public Destructor Done; virtual;`

toString

Declaration `public Function toString:  string; virtual;`

THashTable Object

Hierarchy

THashTable > `TExtendedSortedCollection`(1.3) > `TExtendedCollection`(1.3) > TObject

Description

Hash table that is used to store and retrieve key-value pairs.

The key is a string, while the value is any old styled object data.

Methods

Init

Declaration `public CONSTRUCTOR Init (ALimit, ADelta:  Integer);`

done

Declaration `public Destructor done; virtual;`

containsKey

Declaration `public function containsKey(const key:  string):  Boolean;`

Description Determines whether a key is in this hash table

get

Declaration `public Function get(const key:  string):  PObject;`

Description Returns the value to which the specified key is mapped in this hashtable.

.

Parameters **key** A key in the hashtable

Returns the value to which the key is mapped in this hashtable; nil if the key is not mapped to any value in this hashtable

getByFastIndex

Declaration `public Function getByFastIndex(fastkey: integer): PObject;`

Description Returns the value to which the specified fast key is mapped in this hashtable.

.

Parameters **key** A key in the hashtable

Returns the value to which the key is mapped in this hashtable; nil if the key is not mapped to any value in this hashtable

getKeyName

Declaration `public function getKeyName(index: integer): string;`

Description Returns the key name associated with the specified index, the index starting at 0 to count -1. The returns an empty string if out of bounds. The index is an internal value and is not the same as the fastKey index.

remove

Declaration `public Function remove(key: string): Boolean;`

Description Removes the key (and its corresponding value) from this hashtable. This method does nothing if the key is not in the hashtable.

size

Declaration `public Function size: integer;`

Description Returns the number of elements in this hashtable

clear

Declaration `public procedure clear;`

Description Removes all keys and elements from this HashTable

put

Declaration `public procedure put(key: string;fastkey: integer; obj: PObject);`

Description Maps the specified key to the specified value in this hashtable. Neither the key nor the value can be nil.

1.4 Types

PStack

Declaration `PStack = ^TStack;`

ProcedureType

Declaration `ProcedureType = Function(Item: Pointer): Boolean;`

ForEachProc

Declaration `ForEachProc = Procedure(Item: Pointer);`

PExtendedCollection

Declaration `PExtendedCollection = ^TExtendedCollection;`

PExtendedSortedCollection

Declaration `PExtendedSortedCollection = ^TExtendedSortedCollection;`

PExtendedStringCollection

Declaration `PExtendedStringCollection = ^TExtendedStringCollection;`

PExtendedSortedStringCollection

Declaration `PExtendedSortedStringCollection = ^TExtendedSortedStringCollection;`

PObject

Declaration `PObject = ^TObject;`

PHashTable

Declaration `PHashTable = ^THashTable;`

PHashItem

Declaration `PHashItem = ^THashItem;`

PStringHashItem

Declaration `PStringHashItem = ^TStringHashItem;`

1.5 Constants

MaxCollectionSize

Declaration `MaxCollectionSize = 8192;`

coIndexError

Declaration `coIndexError = -1;`

coOverflow

Declaration `coOverflow = -2;`

KEY_NO_FASTKEY

Declaration `KEY_NO_FASTKEY = high(word);`

1.6 Author

Carl Eric Codere

Chapter 2

Unit dateutil

2.1 Description

Date and time utility routines

This unit is quite similar to the unit dateutils provided with Delphi 6 and Delphi 7. Only a subset of the API found in those units is implemented in this unit, furthermore it contains a new set of extended API's included in the dateexth.inc file.
There are subtle differences with the Delphi implementation: 1. All string related parameters and function results use ISO 8601 formatted date and time strings. 2. The internal format of TDateTime (called TJuliandDate) is not the same as on the Delphi compilers (Internally TJuliandDate is stored as a Julian date) 3. The milliseconds field is only an approximation, and should not be considered as accurate. 4. Becasue everything is coded with floats, the seconds field has a precision of +/- 2 seconds.
All dates are assumed to be in Gregorian calendar date format (This is a proleptic Gregorian calendar unit).

2.2 Overview

`TDateInfo record`

`tfiletime packed record`

`CurrentYear` Returns the current year

`Date` Returns the current date, with the time value equal to midnight.

`DateOf` Strips the time portion from a TJuliandDate value.

`DateTimeToStr` Converts a TJuliandDate value to a string in extended ISO 8601 date and time representation.

`DateTimeToStrExt` Converts a TJuliandDate value to a string in extended ISO 8601 date and time representation.

`DateToStr` Converts a TJuliandDate value to a string in extended ISO 8601 date representation.

`DayOf` Returns the day of the month represented by a TJuliandDate value.

`DaysBetween` Returns the number of days between two specified TJuliandDate values.

`DecodeDate` Returns Year, Month, and Day values for a TJuliandDate value.

`DecodeDateTime` Returns Year, Month, Day, Hour, Minute, Second, and Millisecond values for a TJuliand-Date.

`DecodeTime` Breaks a TJuliandDate value into hours, minutes, seconds, and milliseconds.

`GetCurrentDate` Returns the current date set in the operating system

`GetCurrentTime` Returns the current time set in the operating system

`GetTime` Returns the current time.

`HourOf` Returns the hour of the day represented by a TJuliandDate value.

`IncDay` Returns a date shifted by a specified number of days.

`IncHour` Returns a date/time value shifted by a specified number of hours.

`IncMilliSecond` Returns a date/time value shifted by a specified number of milliseconds.

`IncMinute` Returns a date/time value shifted by a specified number of minutes.

`IncSecond` Returns a date/time value shifted by a specified number of seconds.

`IncWeek` Returns a date shifted by a specified number of weeks.

`IsPM` Indicates whether the time portion of a specified TJuliandDate value occurs after noon.

`IsValidDate` Indicates whether a specified year, month, and day represent a valid date.

`IsValidDateTime` Indicates whether a specified year, month, day, hour, minute, second, and millisecond represent a valid date and time.

`IsValidTime` Indicates whether a specified hour, minute, second, and millisecond represent a valid date and time.

`MinuteOf` Returns the minute of the hour represented by a TJuliandDate value.

`MonthOf` Returns the month of the year represented by a TJuliandDate value.

`Now` Returns the current date and time.

`SameDate` Indicates whether two TJuliandDate values represent the same year, month, and day.

`SameDateTime` Indicates whether two TJuliandDate values represent the same year, month, day, hour, minute, second, and millisecond.

`SameTime` Indicates whether two TJuliandDate values represent the same time of day, ignoring the date portion.

`SecondOf` Returns the second of the minute represented by a TJuliandDate value.

`Time` Returns the current time.

`TimeOf` Strips the date portion from a TJuliandDate value

`TimeToStr` Converts a TJuliandDate value to a string in extended ISO 8601 time representation.

`Today` Returns a TJuliandDate value that represents the current date.

`TryEncodeDate` Returns a TJuliandDate value that represents a specified Year, Month, and Day.

`TryEncodeDateAndTimeToStr`

`TryEncodeDateTime` Returns a TJuliandDate that represents a specified year, month, day, hour, minute, second, and millisecond.

`TryEncodeTime` Returns a TJuliandDate value for a specified Hour, Min, Sec, and MSec.

`TryFileTimeToDateTimeExt` Converts a Win32 FILETIME to a TJuliandDate

`TryStrToDate` Converts a string of date in ISO 8601 to a TJuliandDate value, with a Boolean success code.

`TryStrToDateTime` Converts a string date-time representation to a TJuliandDate value with a Boolean success code.

`TryStrToDateTimeExt` Converts a string to a TJuliandDate value with a Boolean success code.

`TryStrToTime` Converts a string time to a TJuliandDate value with an error default

`TryUNIXToDateTimeExt` Converts a UNIX time_t to a TJuliandDate

`YearOf` Returns the year represented by a TJuliandDate value.

2.3 Classes, Interfaces, Objects and Records

TDateInfo record

Description

Useful structure that contains additional information on a date and time

Fields

DateTime `DateTime: TJuliandDate;`

Actual date and time value

UTC `UTC: boolean;`

Is this value local or according to UTC?

tfiletime packed record

Description

Win32 FILETIME timestamp

Fields

LowDateTime `LowDateTime: longword;`

HighDateTime `HighDateTime: longword;`

2.4 Functions and Procedures

CurrentYear

Declaration `function CurrentYear: word;`

Description Returns the current year

Date

Declaration `function Date: TJuliandDate;`

Description Returns the current date, with the time value equal to midnight.

DateOf

Declaration `function DateOf(const AValue: TJuliandDate): TJuliandDate;`

Description Strips the time portion from a TJuliandDate value.

DateTimeToStr

Declaration `function DateTimeToStr(DateTime: TJuliandDate): string;`

Description Converts a TJuliandDate value to a string in extended ISO 8601 date and time representation.

Returns the extended format representation of a date and time as recommended by ISO 8601 (Gregorian Calendar). The extended format ISO 8601 representation is of the form: YYYY-MM-DDThh:mm:ss

DateTimeToStrExt

Declaration `function DateTimeToStrExt(DateTime: TJuliandDate; utc: boolean): string;`

Description Converts a TJuliandDate value to a string in extended ISO 8601 date and time representation.

Returns the extended format representation of a date and time as recommended by ISO 8601 (Gregorian Calendar). The extended format ISO 8601 representation is of the form: YYYY-MM-DDThh:mm:ss[Z]. Sets the timezone designator if required.

DateToStr

Declaration `function DateToStr(date: TJuliandDate): string;`

Description Converts a TJuliandDate value to a string in extended ISO 8601 date representation.

Returns the extended format representation of a date as recommended by ISO 8601 (Gregorian Calendar). The extended format ISO 8601 representation is of the form: YYYY-MM-DD

DayOf

Declaration `function DayOf(const AValue: TJuliandDate): Word;`

Description Returns the day of the month represented by a TJuliandDate value.

DaysBetween

Declaration `function DaysBetween(const ANow, AThen: TJuliandDate): integer;`

Description Returns the number of days between two specified TJuliandDate values.

DecodeDate

Declaration `procedure DecodeDate(Date: TJuliandDate; var Year, Month, Day: Word);`

Description Returns Year, Month, and Day values for a TJuliandDate value.

DecodeDateTime

Declaration `procedure DecodeDateTime(const AValue: TJuliandDate; var Year, Month, Day, Hour, Minute, Second, MilliSecond: Word);`

Description Returns Year, Month, Day, Hour, Minute, Second, and Millisecond values for a TJuliandDate.

DecodeTime

Declaration `procedure DecodeTime(Time: TJuliandDate; var Hour, Min, Sec, MSec: Word);`

Description Breaks a TJuliandDate value into hours, minutes, seconds, and milliseconds.

GetCurrentDate

Declaration `procedure GetCurrentDate(var Year,Month,Day,DayOfWeek: integer);`

Description Returns the current date set in the operating system

GetCurrentTime

Declaration `procedure GetCurrentTime(var Hour,Minute,Second,Sec100: integer);`

Description Returns the current time set in the operating system

Ranges of the values returned are Hour 0..23, Minute 0..59, Second 0..60, Sec100 0..99.

GetTime

Declaration `function GetTime: TJuliandDate;`

Description Returns the current time.

HourOf

Declaration `function HourOf(const AValue: TJuliandDate): Word;`

Description Returns the hour of the day represented by a TJuliandDate value.

IncDay

Declaration `function IncDay(const AValue: TJuliandDate; const ANumberOfDays: Integer): TJuliandDate;`

Description Returns a date shifted by a specified number of days.

IncHour

Declaration `function IncHour(const AValue: TJuliandDate; const ANumberOfHours: longint): TJuliandDate;`

Description Returns a date/time value shifted by a specified number of hours.

IncMilliSecond

Declaration `function IncMilliSecond(const AValue: TJuliandDate; const ANumberOfMilliSeconds: big_integer_t): TJuliandDate;`

Description Returns a date/time value shifted by a specified number of milliseconds.

IncMinute

Declaration `function IncMinute(const AValue: TJuliandDate; const ANumberOfMinutes: big_integer_t): TJuliandDate;`

Description Returns a date/time value shifted by a specified number of minutes.

IncSecond

Declaration `function IncSecond(const AValue: TJuliandDate; const ANumberOfSeconds: big_integer_t): TJuliandDate;`

Description Returns a date/time value shifted by a specified number of seconds.

IncWeek

Declaration `function IncWeek(const AValue: TJuliandDate; const ANumberOfWeeks: Integer): TJuliandDate;`

Description Returns a date shifted by a specified number of weeks.

IsPM

Declaration `function IsPM(const AValue: TJuliandDate): Boolean;`

Description Indicates whether the time portion of a specified TJuliandDate value occurs after noon.

IsValidDate

Declaration `function IsValidDate(const AYear, AMonth, ADay: Word): Boolean;`

Description Indicates whether a specified year, month, and day represent a valid date.

IsValidDateTime

Declaration `function IsValidDateTime(const AYear, AMonth, ADay, AHour, AMinute, ASecond, AMilliSecond: Word): Boolean;`

Description Indicates whether a specified year, month, day, hour, minute, second, and millisecond represent a valid date and time.

IsValidTime

Declaration `function IsValidTime(const AHour, AMinute, ASecond, AMilliSecond: Word): Boolean;`

Description Indicates whether a specified hour, minute, second, and millisecond represent a valid date and time.

MinuteOf

Declaration `function MinuteOf(const AValue: TJuliandDate): Word;`

Description Returns the minute of the hour represented by a TJuliandDate value.

MonthOf

Declaration `function MonthOf(const AValue: TJuliandDate): Word;`

Description Returns the month of the year represented by a TJuliandDate value.

Now

Declaration `function Now: TJuliandDate;`

Description Returns the current date and time.

SameDate

Declaration `function SameDate(const A, B: TJuliandDate): Boolean;`

Description Indicates whether two TJuliandDate values represent the same year, month, and day.

SameDateTime

Declaration `function SameDateTime(const A, B: TJuliandDate): Boolean;`

Description Indicates whether two TJuliandDate values represent the same year, month, day, hour, minute, second, and millisecond.

SameTime

Declaration `function SameTime(const A, B: TJuliandDate): Boolean;`

Description Indicates whether two TJuliandDate values represent the same time of day, ignoring the date portion.

SecondOf

Declaration `function SecondOf(const AValue: TJuliandDate): Word;`

Description Returns the second of the minute represented by a TJuliandDate value.

Time

Declaration `function Time: TJuliandDate;`

Description Returns the current time.

TimeOf

Declaration `function TimeOf(const AValue: TJuliandDate): TJuliandDate;`

Description Strips the date portion from a TJuliandDate value

TimeToStr

Declaration `function TimeToStr(Time: TJuliandDate): string;`

Description Converts a TJuliandDate value to a string in extended ISO 8601 time representation.

Returns the extended format representation of a time as recommended by ISO 8601 (Gregorian Calendar). The extended format ISO 8601 representation is of the form: hh:mm:ss.

Today

Declaration `function Today: TJuliandDate;`

Description Returns a TJuliandDate value that represents the current date.

TryEncodeDate

Declaration `function TryEncodeDate(Year, Month, Day: Word; var Date: TJuliandDate): Boolean;`

Description Returns a TJuliandDate value that represents a specified Year, Month, and Day.

TryEncodeDateAndTimeToStr

Declaration `function TryEncodeDateAndTimeToStr(const Year, Month, Day, Hour, Minute, Second, MilliSecond: word; UTC: boolean; var AValue: string):boolean;`

Description This routine encodes a complete date and time to its string representation. The encoded string conforms to the ISO 8601 complete representation extended format (YYYY-MM-DDTHH:MM:SS[Z]).

The year value is required, while all other fields are optional. The other fields can be set to EMPTY_DATETIME_FIELD to indicate that they are empty. It also adds the UTC marker if required and if it is set and time information is present.

TryEncodeDateTime

Declaration `function TryEncodeDateTime(const AYear, AMonth, ADay, AHour, AMinute, ASecond, AMilliSecond: Word; var AValue: TJuliandDate): Boolean;`

Description Returns a TJuliandDate that represents a specified year, month, day, hour, minute, second, and millisecond.

TryEncodeTime

Declaration `function TryEncodeTime(Hour, Min, Sec, MSec: Word; var Time: TJuliandDate): Boolean;`

Description Returns a TJuliandDate value for a specified Hour, Min, Sec, and MSec.

TryFileTimeToDateTimeExt

Declaration `function TryFileTimeToDateTimeExt(ftime: tfiletime; var DateTime: TJuliandDate; var UTC: boolean): boolean;`

Description Converts a Win32 FILETIME to a TJuliandDate

This routine converts a 64-bit FILETIME format time into a TJuliandDate format. The value is always according to UTC, so the UTC value shall always return TRUE. Because floating point values are used, the precision of the returned timestamp will be +/- 2 seconds.

TryStrToDate

Declaration `function TryStrToDate(const S: string; var Value: TJuliandDate): Boolean;`

Description Converts a string of date in ISO 8601 to a TJuliandDate value, with a Boolean success code.

In the case where the date does not contain the full representation of a date (for examples, YYYY or YYYY-MM), then the missing values will be set to 1 to be legal.

Most legal dates for Dates of ISO 8601 are supported by this routine.

TryStrToDateTime

Declaration `function TryStrToDateTime(const S: string; var Value: TJuliandDate): Boolean;`

Description Converts a string date-time representation to a TJuliandDate value with a Boolean success code.

Supported formats: 1) Format of Complete Representation for calendar dates (as specified in ISO 8601), which should include the Time designator character. 2) Format: 'YYYY-MM-DD HH:mm:ss [GMT—UTC—UT]' 3) Openoffice 1.1.x HTML date format: 'YYYYMMDD;HHmmssuu' 4) Adobe PDF 'D:YYYYMMDDHHMMSSOHH'mm" format

The returned value will be according to UTC if timezone information is uncluded, otherwise, it will be left as is. To determine if the value was actually converted to UTC, use TryStrToDateTimeExt.

In the case where the date does not contain the full representation of a date (for examples, YYYY or YYYY-MM), then the missing values will be set to 1 to be legal.

TryStrToDateTimeExt

Declaration `function TryStrToDateTimeExt(const S: string; var Value: TJuliandDate; var UTC: boolean) : Boolean;`

Description Converts a string to a TJuliandDate value with a Boolean success code.

This routine also gives information if the value was successfully converted to UTC time or not (if no timezone information was available in the string then the utc value will be false).

Supported formats: 1) Format of Complete Representation for calendar dates (as specified in ISO 8601), which should include the Time designator character. 3) Format: 'YYYY-MM-DD HH:mm:ss [GMT—UTC—UT]' 4) Openoffice 1.1.x HTML date format: 'YYYYMMDD;HHmmssuu' 5) Adobe PDF 'D:YYYYMMDDHHMMSSOHH'mm" format

The returned value will be according to UTC if timezone information is specified.

In the case where the date does not contain the full representation of a date (for examples, YYYY or YYYY-MM), then the missing values will be set to 1 to be legal.

TryStrToTime

Declaration `function TryStrToTime(const S: string; var Value: TJuliandDate): Boolean;`

Description Converts a string time to a TJuliandDate value with an error default

Supported formats: 1) ISO 8601 time format (complete representation) with optional timezone designators. 2) Format: 'HH:mm:ss [GMT—UTC—UT]' 3) Openoffice 1.1.x HTML time format: 'HHmmssuu' 4) Adobe PDF 'D:YYYYMMDDHHMMSSOHH'mm" format

The returned value will be according to UTC if timezone information is specified. The Date field is truncated and is equal to zero upon return.

TryUNIXToDateTimeExt

Declaration `function TryUNIXToDateTimeExt(unixtime: big_integer_t; var DateTime: TJuliandDate; var UTC: boolean): boolean;`

Description Converts a UNIX time_t to a TJuliandDate

This routine converts a UNIX time_t format time into a TJuliandDate format. The time_t is always according to UTC, so the UTC value shall always return TRUE.

On Turbo Pascal and Virtual Pascal this routine shall fail with all values greater that represent years greater than 2038.

YearOf

Declaration `function YearOf(const AValue: TJuliandDate): Word;`

Description Returns the year represented by a TJuliandDate value.

2.5 Types

TJuliandDate

Declaration `TJuliandDate = real;`

Description This is the Julian Day number

float

Declaration `float = real;`

platformword

Declaration `platformword = system.word;`

2.6 Constants

DayMonday

Declaration `DayMonday = 1;`

DayTuesday

Declaration `DayTuesday = 2;`

DayWednesday

Declaration `DayWednesday = 3;`

DayThursday

Declaration `DayThursday = 4;`

DayFriday

Declaration `DayFriday = 5;`

DaySaturday

Declaration `DaySaturday = 6;`

DaySunday

Declaration `DaySunday = 7;`

2.7 Author

Carl Eric Codere

Chapter 3

Unit extdos

3.1 Description

Extended Operating system routines

Routines that extend the capabilities of the pascal DOS unit. It supports more information extraction from the operating system.
Everything string returned and input is/should be encoded in UTF-8 format.
Currently this unit is only supported on the Win32 platform.

3.2 Overview

`TFileAssociation record` Information on file associations for the shell

`TFileStats record`

`TSearchRecExt record` Returned by `FindFirstEx`(3.4) and `FindNextEx`(3.4)

`DirectoryExists` Verifies the existence of a directory

`FileExists` Verifies the existence of a filename

`FindCloseEx` Closes the search and frees the resources previously allocated by a call to `FindFirstEx`(3.4).

`FindFirstEx` Searches the specified directory for the first entry matching the specified file name and set of attributes.

`FindNextEx` Returns the next entry that matches the name and name specified in a previous call to `FindFirstEx`(3.4).

`GetCurrentDirectory` Returns the current active directory

`GetFileATime` Returns the last access date and time of a file

`GetFileAttributes` Returns the attributes of a file

`GetFileCTime` Returns the creation date and time of a file

`GetFileMTime` Returns the last modification date and time of a file

`GetFileOwner` Returns the file owner account name

`GetFilesize` Returns the size of a file

`GetFilestats` Returns information on a file

`GetGlobalConfigDirectory` Returns the path of the current shared application data configuration directory

`GetLoginConfigDirectory` Returns the path of the current user's application data configuration directory

`GetLoginHomeDirectory` Returns the path of the user's home directory

`GetUserFullName` Returns a full name from an account name

`SetCurrentDirectory` Sets the current active directory

`SetFileATime` Change the last access time of a file

`SetFileCTime` Change the creation time of a file

`SetFileMTime` Change the modification time of a file

3.3 Classes, Interfaces, Objects and Records

TFileAssociation record

Description

Information on file associations for the shell

Fields

appname `appname: utf8string;`
Application name associated with this resource

exename `exename: utf8string;`
Application executable and parameters to use on this type of resource

TFileStats record

Description

Statistics for a resource on disk. as returned by `getfilestats`(3.4)

Fields

name `name: utf8string;`

Name of the resource on disk

size `size: big_integer_t;`

Size of the resource on disk

owner `owner: utf8string;`

Owner (User name) of the resource on disk

ctime `ctime: TJuliandDate;`

Creation time of the resource

mtime `mtime: TJuliandDate;`

Last modification time of the resource

atime `atime: TJuliandDate;`

Last access time of the resource

nlink `nlink: integer;`

Number of links to resource

attributes `attributes: tresourceattributes;`

Attributes for this file

association `association: tfileassociation;`

association for this file (operating system)

streamcount `streamcount: integer;`

number of parallel streams for this resource

accesses `accesses: integer;`

number of file accesses since file's creation

utc `utc: boolean;`

indicates if the times are in UTC format, this is always true, unless the filesystem does not support this information.

dev `dev: array[0..127] of char;`

Device where this file resides, this value is represented as an hexadecimal null terminated string and is operating system dependent.

ino `ino: array[0..127] of char;`

Unique file serial number, this may change from one boot to the next.

comment `comment: utf8string;`

Comment associated with this file type (as stored by the operating system)

dirstr `dirstr: utf8string;`

Directory where this file is located

TSearchRecExt record

Description

Returned by `FindFirstEx`(3.4) and `FindNextEx`(3.4)

Fields

Stats `Stats: TFileStats;`

File statistics

FindHandle `FindHandle : THandle;`

Operating system specific data

W32FindData `W32FindData : TWin32FindDataW;`

IncludeAttr `IncludeAttr : longint;`

SearchAttr `SearchAttr: TResourceAttributes;`

3.4 Functions and Procedures

DirectoryExists

Declaration `function DirectoryExists(DName : utf8string): Boolean;`

Description Verifies the existence of a directory

This routine verifies if the directory named can be opened or if it actually exists.

Parameters **DName** Name of the directory to check

Returns FALSE if the directory cannot be opened or if it does not exist.

FileExists

Declaration `Function FileExists(const FName : utf8string): Boolean;`

Description Verifies the existence of a filename

This routine verifies if the file named can be opened or if it actually exists.

Parameters **FName** Name of the file to check

Returns FALSE if the file cannot be opened or if it does not exist.

FindCloseEx

Declaration `procedure FindCloseEx(var SearchRec: TSearchRecExt);`

Description Closes the search and frees the resources previously allocated by a call to `FindFirstEx`(3.4).

Returns 0 on success, otherwise an error code

FindFirstEx

Declaration `function FindFirstEx(path: putf8char; attr: tresourceattributes; var SearchRec:TSearchRecExt): integer;`

Description Searches the specified directory for the first entry matching the specified file name and set of attributes.

Returns 0 on success, otherwise an error code

FindNextEx

Declaration `function FindNextEx(var SearchRec: TSearchRecExt): integer;`

Description Returns the next entry that matches the name and name specified in a previous call to `FindFirstEx`(3.4).

Returns 0 on success, otherwise an error code

GetCurrentDirectory

Declaration `function GetCurrentDirectory(var DirStr: utf8string): boolean;`

Description Returns the current active directory

Parameters **DirStr** The current directory for the process

Returns true on success, otherwise false

GetFileATime

Declaration `function GetFileATime(fname: putf8char; var atime: TJuliandDate): integer;`

Description Returns the last access date and time of a file

This returns the last access time of a file in UTC coordinates. Returns 0 if there was no error, otherwise returns an operating system (if positive) or module (if negative) specific error code.

Parameters **fname** The filename to access (UTF-8 encoded)

atime The file access date in UTC/GMT format

Returns 0 on success, otherwise an error code

GetFileAttributes

Declaration `function GetFileAttributes(fname: putf8char): tresourceattributes;`

Description Returns the attributes of a file

.

Returns If error returns big_integer_t(-1), otherwise the size of the file is returned.

GetFileCTime

Declaration `function GetFileCTime(fname: putf8char; var ctime: TJuliandDate): integer;`

Description Returns the creation date and time of a file

This returns the creation time of a file in UTC coordinates. Returns 0 if there was no error, otherwise returns an operating system (if positive) or module (if negative) specific error code.

Parameters **fname** The filename to access (UTF-8 encoded)
atime The file creation date in UTC/GMT format

Returns 0 on success, otherwise an error code

GetFileMTime

Declaration `function GetFileMTime(fname: putf8char; var mtime: TJuliandDate): integer;`

Description Returns the last modification date and time of a file

This returns the last modification time of a file in UTC coordinates. Returns 0 if there was no error, otherwise returns an operating system (if positive) or module (if negative) specific error code.

Parameters **fname** The filename to access (UTF-8 encoded)
atime The file modification date in UTC/GMT format

Returns 0 on success, otherwise an error code

GetFileOwner

Declaration `function GetFileOwner(fname: putf8char): utf8string;`

Description Returns the file owner account name

This routine returns the owner/creator of this resource as a login user name. This is usually the user name when the user logged on to the system. If this routine is not supported, or if there was an error, this routine returns an empty string.

Parameters **fname** The filename to access (UTF-8 encoded)

Returns The account name on success, otherwise an empty string

GetFilesize

Declaration `function GetFilesize(fname: putf8char): big_integer_t;`

Description Returns the size of a file

.

Returns If error returns big_integer_t(-1), otherwise the size of the file is returned.

GetFilestats

Declaration `function GetFilestats(fname: putf8char; var stats: TFileStats): integer;`

Description Returns information on a file

.

Returns information on a directory or file given by the complete file specification to the file.

Returns 0 if no error, otherwise, an error code

GetGlobalConfigDirectory

Declaration `function GetGlobalConfigDirectory: utf8string;`

Description Returns the path of the current shared application data configuration directory

This routine returns the current path where shared application configuration data should be stored for the logged-on user. If the path does not exist, it is created first.

.

Returns The full path to the requested information, or an empty string if an error occured

GetLoginConfigDirectory

Declaration `function GetLoginConfigDirectory: utf8string;`

Description Returns the path of the current user's application data configuration directory

This routine returns the current path where private application configuration data should be stored for the logged-on user. If the path does not exist, it is created first.

.

Returns The full path to the requested information, or an empty string if an error occured

GetLoginHomeDirectory

Declaration `function GetLoginHomeDirectory: utf8string;`

Description Returns the path of the user's home directory

This routine returns the path of the currently logged in user's home directory.

GetUserFullName

Declaration `function GetUserFullName(account: utf8string): utf8string;`

Description Returns a full name from an account name

From a login name, returns the full name of the user of this account.

Parameters **fname** The account name

Returns The full name of the user, or an empty string upon error or if unknown or unsupported.

SetCurrentDirectory

Declaration `function SetCurrentDirectory(const DirStr: utf8string): boolean;`

Description Sets the current active directory

Parameters **DirStr** The current directory to select for the process

Returns true on success, otherwise false

SetFileATime

Declaration `function SetFileATime(fname: putf8char; newatime: TJuliandDate): integer;`

Description Change the last access time of a file

Changes the access time of a file. The time should be in UTC coordinates.

If the time is not supported (for example a year of 1900 on UNIX system), then an error will be reported and no operation will be performed.

Parameters **fname** The filename to access (UTF-8 encoded)

atime The new access time

Returns 0 on success, otherwise an error code

SetFileCTime

Declaration `function SetFileCTime(fname: putf8char; newctime: TJuliandDate): integer;`

Description Change the creation time of a file

Changes the creation time of a file. The time should be in UTC coordinates.

If the time is not supported (for example a year of 1900 on UNIX system), then an error will be reported and no operation will be performed.

Parameters **fname** The filename to access (UTF-8 encoded)

mtime The new modification time

Returns 0 on success, otherwise an error code

SetFileMTime

Declaration `function SetFileMTime(fname: putf8char; newmtime: TJuliandDate): integer;`

Description Change the modification time of a file

Changes the modification time of a file. The time should be in UTC coordinates.

If the time is not supported (for example a year of 1900 on UNIX system), then an error will be reported and no operation will be performed.

Parameters **fname** The filename to access (UTF-8 encoded)

mtime The new modification time

Returns 0 on success, otherwise an error code

3.5 Types

tresourceattribute

Declaration `tresourceattribute = (...);`

Description Possible attributes of a resource

`attr_any`

`attr_readonly` Resource is read-only globally

`attr_hidden` Resource is hidden

`attr_system` Resource is a system resource

`attr_archive` Resource is an archive

`attr_link` Resource is a link

`attr_directory` Resource is a directory

`attr_temporary` resource is a temporary resource

`attr_encrypted` resource is encrypted by the operating system

`attr_no_indexing` resource should not be indexed

`attr_device` resource is a device

`attr_extended` resource contains extended attributes/reparse point

`attr_compressed` resource is compressed by the filesystem

`attr_offline` Resource is actually offline

`attr_sparse` Resource is a sparse file

tresourceattributes

Declaration `tresourceattributes = set of tresourceattribute;`

3.6 Constants

EXTDOS_STATUS_OK

Declaration `EXTDOS_STATUS_OK = 0;`

Description Return code: No error in operation

EXTDOS_STATUS_UNSUPPORTED

Declaration `EXTDOS_STATUS_UNSUPPORTED = -1;`

Description Return code: This routine is unsupported on this operating system.

EXTDOS_STATUS_DATE_CONVERT_ERROR

Declaration `EXTDOS_STATUS_DATE_CONVERT_ERROR = -2;`

Description Return code: Conversion operation from native date to TJuliandDate was invalid.

EXTDOS_STATUS_DATE_UNSUPPORTED

Declaration `EXTDOS_STATUS_DATE_UNSUPPORTED = -3;`

Description Return code: Filesystem does not support this date

MIN_FILE_YEAR

Declaration `MIN_FILE_YEAR = 1601;`

Description Minimal filesystem year for dates for this operating system **

3.7 Author

Carl Eric Codre

Chapter 4

Unit fileio

4.1 Description

File I/O unit

This a replacement File I/O unit containing routines to access files on disk. They are a better replacement of the standard system I/O routines since they support larger file sizes, as well as debugging features.
If DEBUG is defined, when the application quits, files that are still opened will be displayed on the console. It is important to note, that only the files opened with this API can be checked this way.

4.2 Overview

`FileAssign` Assign a filename to a file

`FileBlockRead` Read data from a file

`FileBlockWrite` Write data to a file

`FileClose` Close a previously opened file

`FileGetPos` Return the current file pointer position

`FileGetSize` Get the size of a file in bytes

`FileIOResult` Return the result of the last I/O operation

`FileReset` Open a file for reading or writing

`FileRewrite` Open/Overwrite a file

`FileSeek` Change the file pointer position of an opened file

`FileTruncate` Truncate a file at the current file position

4.3 Functions and Procedures

FileAssign

Declaration `procedure FileAssign(var F: file; const Name: string);`

Description Assign a filename to a file

This is uased to assign a filename with a file. This assignment will then permit to operate on the file. The assignment is completely compatible with the standard `AssignFile` system unit routine.

FileBlockRead

Declaration `function FileBlockRead(var F: file; var Buf; Count: integer): integer;`

Description Read data from a file

Reads data bytes from the specified opened file. Returns the number of bytes actually read.

FileBlockWrite

Declaration `function FileBlockWrite(var F: file; var Buf; Count: integer): integer;`

Description Write data to a file

Write data bytes to the specified opened file. Returns the number of bytes actually written.

FileClose

Declaration `procedure FileClose(var F: file);`

Description Close a previously opened file

This closes a file that was previously opened by a call to FileOpen or FileReset.

FileGetPos

Declaration `function FileGetPos(var F: file): big_integer_t;`

Description Return the current file pointer position

Returns the current file pointer position within the file. The start of the file is at position zero.

FileGetSize

Declaration `function FileGetSize(var F: file): big_integer_t;`

Description Get the size of a file in bytes

Returns the current size of the file in bytes. The file must be assigned and opened to access this routine.

FileIOResult

Declaration `function FileIOResult:  integer;`

Description Return the result of the last I/O operation

Returns the I/O Result of the last operation. If this routine is not called and the unit is compiled with the DEBUG define, and if there was an error in the last operation, then the next File I/O operation will cause a runtime error with the I/O error code.

FileReset

Declaration `procedure FileReset(var F: file; mode:  integer);`

Description Open a file for reading or writing

This opens a file using a specified file mode. The filemonde constants are defined in other units (fmXXXX constants). The file should have previously been assigned using `FileAssign`.

FileRewrite

Declaration `procedure FileRewrite(var F: file; mode:  integer);`

Description Open/Overwrite a file

This creates a file or overwrites a file (if it does not exist). Currently the mode constant is not used, since the file is always opened in read/write mode.

The file should have previously been assigned using `FileAssign`.

FileSeek

Declaration `procedure FileSeek(var F: file; N: big_integer_t);`

Description Change the file pointer position of an opened file

Seeks to a specific file position in a file (starting from zero which is the start of the file).

FileTruncate

Declaration `procedure FileTruncate(var F: file);`

Description Truncate a file at the current file position

Truncates a file at the current file position. File must be assigned and opened.

4.4 Author

Carl Eric Codere

Chapter 5

Unit fs

5.1 Description

This unit is used to validate and convert filenames according to a specific filesystem convention.

5.2 Overview

TDOSFileSystem Object

TWin32FileSystem Object

TPOSIXFileSystem Object

TOS2FileSystem Object

TAmigaFFSFileSystem Object

TISO9660Level1FileSystem Object

TISO9660Level2Filesystem Object

TUDFFilesystem Object

TJolietFileSystem Object

THFSPlusFileSystem Object

5.3 Classes, Interfaces, Objects and Records

TDOSFileSystem Object

Hierarchy

TDOSFileSystem > TObject

Description

FAT12/FAT16 MS-DOS compatible filesystem validation object

Methods

isValidFilename

Declaration `public function isValidFilename(s: ucs4string): boolean;`

Description Verifies if the filename specified is compatible with this type of filesystem

TWin32FileSystem Object

Hierarchy

TWin32FileSystem > TObject

Description

NTFS compatible filesystem validation object

Methods

isValidFilename

Declaration `public function isValidFilename(s: ucs4string): boolean;`

Description Verifies if the filename specified is compatible with this type of filesystem

TPOSIXFileSystem Object

Hierarchy

TPOSIXFileSystem > TObject

Description

Base POSIX-1995 filesystem validation object.

Methods

GetMaxFilenameLength

Declaration `public function GetMaxFilenameLength: integer;`

isValidFilename

Declaration `public function isValidFilename(s: ucs4string): boolean;`

Description Verifies if the filename specified is compatible with this type of filesystem

TOS2FileSystem Object

Hierarchy

TOS2FileSystem > TObject

Description

HPFS compatible filesystem validation object

Methods

isValidFilename

Declaration `public function isValidFilename(s: ucs4string): boolean;`

Description Verifies if the filename specified is compatible with this type of filesystem

TAmigaFFSFileSystem Object

Hierarchy

TAmigaFFSFileSystem > TObject

Description

FFS compatible filesystem validation object

Methods

isValidFilename

Declaration `public function isValidFilename(s: ucs4string): boolean;`

Description Verifies if the filename specified is compatible with this type of filesystem

TISO9660Level1FileSystem Object

Hierarchy

TISO9660Level1FileSystem > TObject

Description

ISO 9660 Level 1 compatible filesystem validation object

Methods

isValidFilename

Declaration `public function isValidFilename(s: ucs4string): boolean;`

Description Verifies if the filename specified is compatible with this type of filesystem

TISO9660Level2Filesystem Object

Hierarchy

TISO9660Level2Filesystem > TObject

Description

ISO 9660 Level 2 compatible filesystem validation object

Methods

isValidFilename

Declaration `public function isValidFilename(s: ucs4string): boolean;`

Description Verifies if the filename specified is compatible with this type of filesystem

TUDFFilesystem Object

Hierarchy

TUDFFilesystem > TObject

Description

UDF compatible filesystem validation object

Methods

isValidFilename

Declaration `public function isValidFilename(s: ucs4string): boolean;`

Description Verifies if the filename specified is compatible with this type of filesystem

TJolietFileSystem Object

Hierarchy

TJolietFileSystem > TObject

Description

CD-ROM Joliet extensions compatible filesystem validation object

Methods

isValidFilename

Declaration `public function isValidFilename(s: ucs4string): boolean;`

Description Verifies if the filename specified is compatible with this type of filesystem

THFSPlusFileSystem Object

Hierarchy

THFSPlusFileSystem > TObject

Description

MacOS HFS+ compatible filesystem validation object

Methods

isValidFilename

Declaration `public function isValidFilename(s: ucs4string): boolean;`

Description Verifies if the filename specified is compatible with this type of filesystem

Chapter 6

Unit ietf

6.1 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).

6.2 Overview

`file_pathsplit`

`http_pathsplit`

`langtag_isvalid`

`langtag_split`

`mime_isvalidcontenttype`

`uri_split` Extract information from an URI string

`urn_isvalid` Verifies the validity of a complete URN string

`urn_isvalidnid`

`urn_pathsplit`

`urn_split` Splits an URN string in its separate components

6.3 Functions and Procedures

file_pathsplit

Declaration `function file_pathsplit(path: string; var directory, name: string): boolean;`

Description Splits a path string returned by uri_split into its individual components for the file URI.

http_pathsplit

Declaration `function http_pathsplit(path: string; var directory, name: string): boolean;`

Description Splits a path string returned by uri_split into its individual components for the http URI.

langtag_isvalid

Declaration `function langtag_isvalid(const s: string): boolean;`

langtag_split

Declaration `function langtag_split(const s: string; var primary,sub: string): boolean;`

mime_isvalidcontenttype

Declaration `function mime_isvalidcontenttype(const s: shortstring): boolean;`

Description abstract(Validates the syntax of a MIME type)

This routine is used to validate if MIME content type signature consists of a constructed valid syntax. It does not validate if the MIME type is assigned or if it actually exists or not.

Parameters **s** MIME type signature to verify

Returns TRUE if the signature has valid syntax

uri_split

Declaration `function uri_split(url: string; var scheme, authority,path, query: string): boolean;`

Description Extract information from an URI string

Given an URI complete absolute specification string, extract and return the scheme, authority, path and query components of the URI. The exact definition of these terms is specified in IETF RFC 2396.

Parameters **url** URI to check

Returns FALSE if the URI is not valid, otherwise returns TRUE

urn_isvalid

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

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: http://www.iana.org/assignments/urn-namespaces

Returns TRUE if this is a registered or experimental NID string

urn_pathsplit

Declaration `function urn_pathsplit(path: string; var namespace, nss: string): boolean;`

Description Splits a path string returned by uri_split into its individual components for URN.

urn_split

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.

Parameters

urn Complete URN string to separate

urnidstr Signature URN:

nidstr Namespace identifier NID

nssstr Namespace specific string NSS

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

6.4 Author

Carl Eric Codere

Chapter 7

Unit iso3166

7.1 Description

Country code unit

This unit is used to check the country codes as well as return information on the country, according to ISO 3166.
The lists were converted from the semicolon delimited version available here: http://www.iso.org/iso/en/prods-services/iso3166ma/
The version used is based on version of 2004-04-26.

7.2 Overview

`getcountryname_en` Returns the country name in english according to its country code.

`getcountryname_fr` Returns the country name in french according to its country code.

`isvalidcountrycode` Verifies if the 2 letter country code is valid

7.3 Functions and Procedures

getcountryname_en

Declaration `function getcountryname_en(s: shortstring): shortstring;`

Description Returns the country name in english according to its country code.

The country code is case-insensitive.

The returned string is encoded according to ISO-8859-1.

getcountryname_fr

Declaration `function getcountryname_fr(s: shortstring): shortstring;`

Description Returns the country name in french according to its country code.

The country code is case-insensitive.

The returned string is encoded according to ISO-8859-1.

isvalidcountrycode

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

Description Verifies if the 2 letter country code is valid

This routine checks if the two letter country code is valid (as defined in ISO3166-1). The country code is not case sensitive.

Parameters **s** The three digit country code

Returns TRUE if the country code is valid, otherwise returns FALSE

7.4 Author

Carl Eric Codere

Chapter 8

Unit iso639

8.1 Description

Language code unit

This unit is used to check the language codes as well as return information on the country, according to ISO 639-1 and ISO 639-2.
The database was taken from the following site: http://www.loc.gov/standards/iso639-2/ISO-639-2_values_8bits.txt
The database used is from 2004-10-19.

8.2 Overview

`getlangcode_en` Returns the 2 character code related to the english name of the language.

`getlangcode_fr` Returns the 2 character code related to the french name of the language.

`getlangname_en` Returns the language name in english for the specified language code.

`getlangname_fr` Returns the language name in french for the specified language code.

`isvalidlangcode` Verifies if the 2 or 3 letter language code is valid

8.3 Functions and Procedures

getlangcode_en

Declaration `function getlangcode_en(name: shortstring): shortstring;`

Description Returns the 2 character code related to the english name of the language.

The search is not case sensitive (according to ISO 639-1). If there is no 2 character language code for this language, or if the language name is not found, the routine returns an empty string.

The language name string should be encoded according to ISO-8859-1.

Parameters **name** The name of the language

Returns The 2 character language code

getlangcode_fr

Declaration `function getlangcode_fr(name: shortstring): shortstring;`

Description Returns the 2 character code related to the french name of the language.

The search is not case sensitive (according to ISO 639-1). If there is no 2 character language code for this language, or if the language name is not found, the routine returns an empty string.

The language name string should be encoded according to ISO-8859-1.

Parameters **name** The name of the language

Returns The 2 character language code

getlangname_en

Declaration `function getlangname_en(s: shortstring): shortstring;`

Description Returns the language name in english for the specified language code.

The language code IS case insensitive and can be either 2 or 3 characters in length (according to ISO 639-1 and ISO 639-2 respectively)

The returned string is encoded according to ISO-8859-1. If there are alternate names for the language, only the first alternate name is returned.

Parameters **s** The two or three digit language code

getlangname_fr

Declaration `function getlangname_fr(s: shortstring): shortstring;`

Description Returns the language name in french for the specified language code.

The language code IS case insensitive and can be either 2 or 3 characters in length (according to ISO 639-1 and ISO 639-2 respectively)

The returned string is encoded according to ISO-8859-1. If there are alternate names for the language, only the first alternate name is returned.

Parameters **s** The two or three digit language code

isvalidlangcode

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

Description Verifies if the 2 or 3 letter language code is valid

This routine checks if the two or three letter language code is valid (as defined in ISO 639, part 1 and part 2 respectively). The language code IS case sensitive and should be in lower case.

Parameters **s** The two or three digit language code

Returns TRUE if the language code is valid, otherwise returns FALSE

8.4 Author

Carl Eric Codere

Chapter 9

Unit locale

9.1 Description

Localisation unit

This unit is used to convert different locale information. ISO Standards are used where appropriate.
The exact representations that are supported are the following: Calendar Date: Complete Representation - basic Caldedar Date: Complete Representation - extended Calendar Date: Representations with reduced precision Time of the day: Local time of the day: Complete representation - basic Time of the day: Local time of the day: Complete representation - extended Time of the day: UTC Time : Complete representation - basic Time of the day: UTC Time: Complete representation - extended Time of the day: Local and UTC Time: extended format
Credits where credits are due, information on the ISO and date formats where taken from http://www.cl.cam.ac.uk/ mgk25/isc time.html

9.2 Overview

`DateTimeStrFormat` Convert a string representation of a date to a TJuliandDate according to a specified template

`EvaluateDateTime`

`GetCharEncoding`

`GetISODateString`

`GetISODateStringBasic`

`GetISODateTimeString`

`GetISOTimeString`

`GetISOTimeStringBasic`

`ISOLanguageCodeToMicrosoftCode`

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

`IsValidISODateStringExt` Verifies if the date is in a valid ISO 8601 format and returns the decoded values

`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

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

`MicrosoftCodePageToMIMECharset`

`MicrosoftLangageCodeToISOCode`

`MIMECharsetToMicrosoftCodePage`

`UNIXToDateTime`

9.3 Functions and Procedures

DateTimeStrFormat

Declaration `function DateTimeStrFormat(const DateTimeFormat : string; const DateTimeStr : string) : TJuliandDate;`

Description Convert a string representation of a date to a TJuliandDate according to a specified template

Date and time formats are specified by date and time pattern strings. Within date and time pattern strings, unquoted letters from 'A' to 'Z' and from 'a' to 'z' are interpreted as pattern letters representing the components of a date or time string. Text can be quoted using single quotes (') or double quotes (") to avoid interpretation. """ represents a single quote. All other characters are not interpreted; they're simply copied into the output string during formatting or matched against the input string during parsing. The format is a subset of the SimpleDateFormat Java Class.

The following pattern letters are defined (all other characters from 'A' to 'Z' and from 'a' to 'z' are reserved):

Letter Date or Time Component Presentation Examples y Year Year 1996; 96 M Month in year Number 07 w Week in year Number 27 D Day in year Number 189 d Day in month Number 10 a Am/pm marker Text PM H Hour in day (0-23) Number 0 h Hour in am/pm (1-12) Number 12 m Minute in hour Number 30 s Second in minute Number 55 S Millisecond Number 978

Year: For formatting, if the number of pattern letters is 2, the year is truncated to 2 digits; otherwise it is interpreted as a number.

For parsing, if the number of pattern letters is more than 2, the year is interpreted literally, regardless of the number of digits.

For parsing with the abbreviated year pattern ("y" or "yy"), the year is assumed to be in the 20th century.

Month: If the number of pattern letters is 3 or more, the month is interpreted as text; otherwise, it is interpreted as a number.

EvaluateDateTime

Declaration `function EvaluateDateTime(const DateTimeFormat : string; dt: TJuliandDate): string;`

GetCharEncoding

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

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

Description Returns the extended format representation of a date as recommended by ISO 8601 (Gregorian Calendar).

Returns an empty string if there is an error. The extended representation separates each member (year,month,day) with a dash character (-).

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

GetISODateStringBasic

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

Description Returns the basic format representation of a date 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

GetISODateTimeString

Declaration `function GetISODateTimeString(Year, Month, Day, Hour, Minute, Second: Word; UTC: Boolean): shortstring;`

Description Returns the extended format representation of a date and time as recommended by ISO 8601 (Gregorian Calendar). The extended format ISO 8601 representation is of the form: YYYY-MM-DDTHH:mm:ss[Timezone offset]

Returns an empty string if there is an error.

GetISOTimeString

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

Description Returns the extended format representation of a time as recommended by ISO 8601 (Gregorian Calendar).

Returns an empty string if there is an error. The extended representation separates each member (hour,minute,second) with a colon (:).

GetISOTimeStringBasic

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

Description Returns the basic format representation of a time as recommended by ISO 8601 (Gregorian Calendar).

Returns an empty string if there is an error. The extended representation separates each member (hour,minute,second) with a colon (:).

ISOLanguageCodeToMicrosoftCode

Declaration `function ISOLanguageCodeToMicrosoftCode(lang: string): integer;`

Description Using a ISO 639-2 language code identifier, return the resulting Microsoft language identifier (as defined by Microsoft and OS/2)

IsValidISODateString

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

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

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

strict If set, the format must exactly be YYYYMMDD or YYYY-MM-DD. If not set, less precision is allowed

Returns TRUE if the date string is valid otherwise false

IsValidISODateStringExt

Declaration `function IsValidISODateStringExt(datestr: shortstring; strict: boolean; var Year, Month,Day: word): boolean;`

Description Verifies if the date is in a valid ISO 8601 format and returns the decoded values

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

strict If set, the format must exactly be YYYYMMDD or YYYY-MM-DD. If not set, less precision is allowed

Year The year value from, always valid

Month The month value from 0 to 12, 0 indicating that this parameter was not present

Day The day value from 0 to 31, 0 indicating that this parameter was not present

Returns TRUE if the date string is valid otherwise false

IsValidISODateTimeString

Declaration `function IsValidISODateTimeString(str: shortstring; strict: boolean): 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. Also validates an entry if it only contains the date component (it is automatically detected).

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

strict If set to TRUE then the complete representation must be present, either in basic or extended format to consider the date and time valid

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

IsValidISOTimeString

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

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

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

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

strict The value must contain all the required parameters with UTC, either in basic or extended format to be valid

Returns TRUE if the time string is valid otherwise false

IsValidISOTimeStringExt

Declaration `function IsValidISOTimeStringExt(timestr: shortstring; strict: boolean; var hour,min,sec: word; var offhour,offmin: smallint): boolean;`

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

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

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

strict The value must contain all the required parameters with UTC, either in basic or extended format to be valid

Returns TRUE if the time string is valid otherwise false

MicrosoftCodePageToMIMECharset

Declaration `function MicrosoftCodePageToMIMECharset(cp: word): string;`

Description Using a code page identifier (as defined by Microsoft and OS/2) return the resulting IANA encoding alias string

Parameters **cp** Codepage

Returns IANA String representation of the character encoding

MicrosoftLangageCodeToISOCode

Declaration `function MicrosoftLangageCodeToISOCode(langcode: integer): string;`

Description Using a Microsoft language identifier (as defined by Microsoft and OS/2) return the resulting ISO 639-2 language code identifier.

MIMECharsetToMicrosoftCodePage

Declaration `function MIMECharsetToMicrosoftCodePage(charset: string): word;`

Description Using a IANA charactet set identifier, return the equivalent the code page identifier (as defined by Microsoft and OS/2)

Parameters **charset** IANA String representation of the character encoding

Returns cp Codepage

UNIXToDateTime

Declaration `procedure UNIXToDateTime(epoch: big_integer_t; 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.

9.4 Constants

YEAR_CHAR

Declaration `YEAR_CHAR = 'y';`

DAY_IN_MONTH

Declaration `DAY_IN_MONTH = 'd';`

DAY_IN_WEEK

Declaration `DAY_IN_WEEK = 'E';`

AM_PM_INDICATOR

Declaration `AM_PM_INDICATOR = 'a';`

HOUR_IN_DAY_24

Declaration `HOUR_IN_DAY_24 = 'H';`

HOUR_IN_DAY_12

Declaration `HOUR_IN_DAY_12 = 'h';`

MINUTE_IN_HOUR

Declaration `MINUTE_IN_HOUR = 'm';`

SECOND_IN_MINUTE

Declaration `SECOND_IN_MINUTE = 's';`

MILLISECOND_CHAR

Declaration `MILLISECOND_CHAR = 'S';`

MONTH_IN_YEAR_CHAR

Declaration `MONTH_IN_YEAR_CHAR = 'M';`

PatternChars

Declaration
```
PatternChars:  set of char = [ YEAR_CHAR, DAY_IN_MONTH, DAY_IN_WEEK,
AM_PM_INDICATOR, HOUR_IN_DAY_24, HOUR_IN_DAY_12, MINUTE_IN_HOUR,
SECOND_IN_MINUTE, MILLISECOND_CHAR, MONTH_IN_YEAR_CHAR];
```

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)

9.5 Author

Carl Eric Codere

Chapter 10

Unit unicode

10.1 Description

unicode support unit

This unit contains routines to convert between the different unicode encoding schemes.
All UNICODE/ISO 10646 pascal styled strings are limited to MAX_STRING_LENGTH characters for normal strings and MAX_LONG_STRING_LENGTH for long string types, both can be used in most of the routines thanks to open array paraemters. Null terminated unicode strings are limited by the compiler and integer type size.
Since all these encoding are variable length, except the UCS-4 (which is equivalent to UTF-32 according to ISO 10646:2003) and UCS-2 encoding, to parse through characters, every string should be converted to UCS-4 or UCS-2 before being used.
The principal encoding scheme for this unit is UCS-4.
Unicode tables are based on Unicode 4.1

10.2 Overview

`ansistrdispose` Disposes of an ansi null-terminated string

`ansistrnew` Allocates and copies an ISO-8859-1 null-terminated string from a null-terminated string

`ansistrnewstr` Allocates and copies an ansi string to a null-terminated string

`ansistrpas` Converts a null-terminated ISO-8859-1 string to a Pascal-style ASCII encoded string.

`asciistrnew` Allocates and copies an ascii null-terminated string from a null-terminated string

`asciistrnewstr` Allocates and copies an ascii string to a null-terminated string

`asciistrpas` Converts a null-terminated ASCII string to a Pascal-style ASCII encoded string.

`ConvertFromUCS4` Convert an UCS-4 string to a single byte encoded string

`ConvertToUCS4` Convert a byte encoded string to an UCS-4 string

`ConvertUCS2ToUCS4` Convert an UCS-2 string to an UCS-4 string

`ConvertUCS4ToUCS2` Convert an UCS-4 string to an UCS-2 string

`ConvertUCS4toUTF16` Convert an UCS-4 string to an UTF-16 string

`convertUCS4toUTF8` Convert an UCS-4 string to an UTF-8 string

`ConvertUTF16ToUCS4` Convert an UTF-16 string to an UCS-4 string

`ConvertUTF8ToUCS4` Convert an UTF-8 string to an UCS-4 string

`ucs2strdispose` Disposes of an UCS-2 null terminated string on the heap

`ucs2strlcopyucs4` Convert an UCS-2 null terminated string to an UCS-4 null terminated string

`ucs2strlen` Returns the number of characters in the null terminated UCS-2 string

`ucs2strnew` Converts an UCS-4 null terminated string to an UCS-2 null terminated string

`ucs2_isvalid` Checks if the UCS-2 character is valid

`ucs2_length` Returns the current length of an UCS-2 string

`ucs2_setlength` Set the new dynamic length of an ucs-2 string

`ucs2_upcase` Converts a character to an uppercase character

`ucs4strcheck`

`ucs4strdispose` Disposes of an UCS-4 null terminated string on the heap

`ucs4strfill` Fills an UCS-4 null terminated string with the specified UCS-4 character

`ucs4strlen` Returns the number of characters in the null terminated UCS-4 string

`ucs4strnew` Converts a null terminated string to an UCS-4 null terminated string

`ucs4strnewFromucs4` Converts an UCS-4 string to a null terminated UCS-4 string.

`ucs4strnewstr` Converts a pascal string to an UCS-4 null terminated string

`ucs4strnewucs2` Converts an UCS-2 null terminated string to an UCS-4 null terminated string

`ucs4strnewucs4` Allocates and copies an UCS-4 null terminated string

`ucs4strpas` Converts a null-terminated UCS-4 string to a Pascal-style UCS-4 string.

`ucs4strpastoASCII` Converts a null-terminated UCS-4 string to a Pascal-style ASCII encoded string.

`ucs4strpastoISO8859_1` Converts a null-terminated UCS-4 string to a Pascal-style ISO 8859-1 encoded string.

`ucs4strpastoUTF8` Converts a null-terminated UCS-4 string to a Pascal-style UTF-8 encoded string.

`ucs4strpcopy` Copies a Pascal-style UCS-4 string to a null-terminated UCS-4 string.

`ucs4strposISO8859_1`

`ucs4strtrim` Allocate and copy trimmed an UCS-4 null terminated string

`ucs4_concat` Concatenates two UCS-4 strings, and gives a resulting UCS-4 string

`ucs4_concatascii` Concatenates an UCS-4 string with an ASCII string, and gives a resulting UCS-4 string

`ucs4_converttoiso8859_1` Converts an UCS-4 string to an ISO-8859-1 string

`ucs4_converttoutf8` Converts an UCS-4 string to an UTF-8 string

`ucs4_copy` Returns an UCS-4 substring of an UCS-4 string

`ucs4_delete` Deletes a substring from a string

`ucs4_equal` Checks if both UCS-4 strings are equal

`ucs4_equalascii` Checks if an ASCII string is equal to an UCS-4 string

`ucs4_getbasechar` Returns the base character representation of a canonical character

`ucs4_getNumericValue` Retrieves the Unicode numeric value of a character

`ucs4_isdigit` Determines if the specified character is a digit character

`ucs4_ishexdigit` Determines if the specified character is an hex-digit character

`ucs4_issupported` Checks if conversion from/to this character set format to/from UCS-4 is supported

`ucs4_isterminal` Determines if the specified character is a sentence terminating character

`ucs4_isvalid` Checks if the UCS-4 character is valid

`ucs4_iswhitespace` Determines if the specified character is a whitespace character

`ucs4_length` Returns the current length of an UCS-4 string

`ucs4_lowcase` Converts a character to a lowercase character

`ucs4_pos` Searches for an UCS-4 substring in an UCS-4 string

`ucs4_posascii` Searches for an ASCII substring in an UCS-4 string

`ucs4_removeaccents` Returns the base string representation of a canonical string

`ucs4_setlength` Set the new dynamic length of an UCS-4 string

`ucs4_trim` Trims trailing and leading spaces and control characters from an UCS-4 string.

`ucs4_trimleft` Trims leading spaces and control characters from an UCS-4 string.

`ucs4_trimright` Trims trailing spaces and control characters from an UCS-4 string.

`ucs4_upcase` Converts a character to an uppercase character

`utf16_length` Returns the current length of an UTF-16 string

`utf16_setlength` Set the length of an UTF-16 string

`utf16_sizeencoding` Returns the number of characters that are used to encode this character

`utf8islegal` Validates the legality of an UTF-8 null terminated string

`utf8strdispose` Disposes of an UTF-8 null terminated string on the heap

`utf8stringdispose` Frees an UTF-8 string that was allocated with `utf8strdispose`(10.3)

`utf8stringdup` Allocates and copies an UTF-8 string to a pointer to an UTF-8 string

`utf8strlcopyucs4` Convert an UTF-8 null terminated string to an UCS-4 null terminated string

`utf8strlen` Returns the length of the string, not counting the null character

`utf8strnew` Converts an UCS-4 null terminated string to an UTF-8 null terminated string

`utf8strnewstr` Converts an UTF-8 string to a null terminated UTF-8 string.

`utf8strnewutf8` Allocates and copies an UTF-8 null terminated string

`utf8strpas` Converts a null-terminated UTF-8 string to a Pascal-style UTF-8 encoded string.

`utf8strpastoASCII` Converts a null-terminated UTF-8 string to a Pascal-style ASCII encoded string.

`utf8strpastoISO8859_1` Converts a null-terminated UTF-8 string to a Pascal-style ISO 8859-1 encoded string.

`utf8strpastostring` Converts an UTF-8 null terminated string to a string encoded to a different code page

`utf8_islegal` Returns if the specified UTF-8 string is legal or not

`utf8_length` Returns the current length of an UTF-8 string

`utf8_setlength` Set the length of an UTF-8 string

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

10.3 Functions and Procedures

ansistrdispose

Declaration `function ansistrdispose(p: pchar): pchar;`

Description Disposes of an ansi null-terminated string

Does nothing if the pointer passed is already nil.

ansistrnew

Declaration `function ansistrnew(src: pchar): pchar;`

Description Allocates and copies an ISO-8859-1 null-terminated string from a null-terminated string

The value returned is nil, if the src pointer was also nil.

ansistrnewstr

Declaration `function ansistrnewstr(const str: string): pchar;`

Description Allocates and copies an ansi string to a null-terminated string

The value returned is nil, if the length of str = 0.

ansistrpas

Declaration `function ansistrpas(src: pchar): string;`

Description Converts a null-terminated ISO-8859-1 string to a Pascal-style ASCII encoded string.

The returned string shall be empty if the pointer was nil.

asciistrnew

Declaration `function asciistrnew(src: pchar): pchar;`

Description Allocates and copies an ascii null-terminated string from a null-terminated string

The value returned is nil, if the src pointer was also nil.

asciistrnewstr

Declaration `function asciistrnewstr(const str: string): pchar;`

Description Allocates and copies an ascii string to a null-terminated string

The value returned is nil, if the length of str = 0.

asciistrpas

Declaration `function asciistrpas(src: pchar): string;`

Description Converts a null-terminated ASCII string to a Pascal-style ASCII encoded string.

The returned string shall be empty if the pointer was nil.

ConvertFromUCS4

Declaration `function ConvertFromUCS4(const source: array of ucs4char; var dest: string; desttype: string): integer;`

Description Convert an UCS-4 string to a single byte encoded string

This routine converts an UCS-4 string stored in native byte order (native endian) to a single-byte encoded string.

The string is limited to MAX_STRING_LENGTH 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 and UTF-8.

Parameters **source** Source UCS-4 string, such as ucs4string or longucs4string

desttype Indicates the single byte encoding scheme - case-insensitive

Returns `UNICODE_ERR_OK`(10.5) if there was no error in the conversion

ConvertToUCS4

Declaration `function ConvertToUCS4(source: string; var dest: array of ucs4char; srctype: string): integer;`

Description Convert a byte encoded string to an UCS-4 string

This routine converts a single byte encoded string to an UCS-4 string stored in native byte order

Characters that cannot be converted are converted to the default fallback character: DEFAULT_FALLBACK_CHAR and an error code will also be returned by the function

The string is limited to MAX_STRING_LENGTH 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 - case-insensitive

Returns `UNICODE_ERR_OK`(10.5) if there was no error in the conversion

ConvertUCS2ToUCS4

Declaration `function ConvertUCS2ToUCS4(src: array of ucs2char; var dst: array of ucs4char): integer;`

Description Convert an UCS-2 string to an UCS-4 string

This routine converts an UCS-2 string to an UCS-4 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 UCS-4 coded string

Returns `UNICODE_ERR_OK`(10.5) if there was no error in the conversion

ConvertUCS4ToUCS2

Declaration `function ConvertUCS4ToUCS2(src: array of ucs4char; var dst: array of ucs2char): integer;`

Description Convert an UCS-4 string to an UCS-2 string

This routine converts an UCS-4 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 UCS-4 character or a complete UCS-4 string

dest Resulting UCS-2 coded string

Returns `UNICODE_ERR_OK`(10.5) if there was no error in the conversion

ConvertUCS4toUTF16

Declaration `function ConvertUCS4toUTF16(src: array of ucs4char; var dest: utf16string): integer;`

Description Convert an UCS-4 string to an UTF-16 string

This routine converts an UCS-4 string to an UTF-16 string. Both strings must be stored in native byte order (native endian).

Parameters **src** Either a single UCS-4 character or a complete UCS-4 string

dest Resulting UTF-16 coded string

Returns `UNICODE_ERR_OK`(10.5) if there was no error in the conversion

convertUCS4toUTF8

Declaration `function convertUCS4toUTF8(s: array of ucs4char; var outstr: utf8string): integer;`

Description Convert an UCS-4 string to an UTF-8 string

Converts an UCS-4 string or character in native endian to an UTF-8 string.

Parameters **s** Either a single UCS-4 character or a complete UCS-4 string

outstr Resulting UTF-8 coded string

Returns `UNICODE_ERR_OK`(10.5) if there was no error in the conversion

ConvertUTF16ToUCS4

Declaration `function ConvertUTF16ToUCS4(src: utf16string; var dst: array of ucs4char): integer;`

Description Convert an UTF-16 string to an UCS-4 string

This routine converts an UTF-16 string to an UCS-4 string. Both strings must be stored in native byte order (native endian).

Returns `UNICODE_ERR_OK`(10.5) if there was no error in the conversion

ConvertUTF8ToUCS4

Declaration `function ConvertUTF8ToUCS4(src: utf8string; var dst: array of ucs4char): integer;`

Description Convert an UTF-8 string to an UCS-4 string

This routine converts an UTF-8 string to an UCS-4 string that is stored in native byte order.

Returns `UNICODE_ERR_OK`(10.5) if there was no error in the conversion

ucs2strdispose

Declaration `function ucs2strdispose(str: pucs2char): pucs2char;`

Description Disposes of an UCS-2 null terminated string on the heap

Disposes of a string that was previously allocated with `ucs2strnew`, and sets the pointer to nil.

ucs2strlcopyucs4

Declaration `function ucs2strlcopyucs4(src: pucs2char; dst: pucs4char; maxlen: integer): pucs4char;`

Description Convert an UCS-2 null terminated string to an UCS-4 null terminated string

This routine converts an UCS-2 encoded null terminared string to an UCS-4 null terminated string that is stored in native byte order, up to length conversion. The destination buffer should already have been allocated.

Returns nil if there was an error in the conversion

ucs2strlen

Declaration `function ucs2strlen(str: pucs2char): integer;`

Description Returns the number of characters in the null terminated UCS-2 string

Parameters **str** The UCS-2 null terminated string to check

Returns The number of characters in str, not counting the null character

ucs2strnew

Declaration `function ucs2strnew(src: pucs4char): pucs2char;`

Description Converts an UCS-4 null terminated string to an UCS-2 null terminated string

The memory for the buffer is allocated. Use `ucs2strdispose`(10.3) to dispose of the allocated string. The string is null terminated. If src is nil, this routine returns nil, and does not allocate anything.

Returns nil if the conversion cannot be represented in UCS-2 encoding, or nil if there was an error

ucs2_isvalid

Declaration `function ucs2_isvalid(ch: ucs2char): boolean;`

Description Checks if the UCS-2 character is valid

This routine verifies if the UCS-2 character is within the valid ranges of UCS-2 characters, as specified in the Unicode standard 4.0. BOM characters are NOT valid with this routine.

ucs2_length

Declaration `function ucs2_length(const s: array of ucs2char): integer;`

Description Returns the current length of an UCS-2 string

ucs2_setlength

Declaration `procedure ucs2_setlength(var s: array of ucs2char; l: integer);`

Description Set the new dynamic length of an ucs-2 string

ucs2_upcase

Declaration `function ucs2_upcase(c: ucs2char): ucs2char;`

Description Converts a character to an uppercase character

This routine only supports the simple form case folding algorithm (e.g full form is not supported).

ucs4strcheck

Declaration `procedure ucs4strcheck(p: pucs4char; maxcount: integer; value: ucs4char);`

Description This routine checks the validity of an UCS-4 null terminated string. It first skips to the null character, and if maxcount is greater than the index, verifies that the values in memory are of value VALUE.

Otherwise, returns an ERROR. This routine is used with ucs4strfill.

ucs4strdispose

Declaration `function ucs4strdispose(str: pucs4char): pucs4char;`

Description Disposes of an UCS-4 null terminated string on the heap

Disposes of a string that was previously allocated with `ucs4strnew`, and sets the pointer to nil.

ucs4strfill

Declaration `function ucs4strfill(var p: pucs4char; count: integer; value: ucs4char): pucs4char;`

Description Fills an UCS-4 null terminated string with the specified UCS-4 character

Fills a memory region consisting of ucs-4 characters with the specified UCS-4 character.

ucs4strlen

Declaration `function ucs4strlen(str: pucs4char): integer;`

Description Returns the number of characters in the null terminated UCS-4 string

Parameters **str** The UCS-4 null terminated string to check

Returns The number of characters in str, not counting the null character

ucs4strnew

Declaration `function ucs4strnew(str: pchar; srctype: string): pucs4char;`

Description Converts a null terminated string to an UCS-4 null terminated string

The memory for the buffer is allocated. Use `ucs4strdispose`(10.3) to dispose of the allocated string. The string is null terminated. If str is nil, then this routine returns nil and does not allocate anything.

Parameters **str** The string to convert, single character coded, or UTF-8 coded

srctype The encoding of the string, UTF-8 is also valid - case-insensitive

ucs4strnewFromucs4

Declaration `function ucs4strnewFromucs4(src: array of ucs4char): pucs4char;`

Description Converts an UCS-4 string to a null terminated UCS-4 string.

The memory for the storage of the string is allocated by the routine, and the ending null character is also added.

Returns The newly allocated UCS-4 null terminated string

ucs4strnewstr

Declaration `function ucs4strnewstr(str: string; srctype: string): pucs4char;`

Description Converts a pascal string to an UCS-4 null terminated string

The memory for the buffer is allocated. Use `ucs4strdispose`(10.3) to dispose of the allocated string. The string is null terminated. If the original string contains some null characters, those nulls are removed from the resulting string.

Parameters **str** The string to convert, single character coded

srctype The encoding of the string, UTF-8 is also valid - case-insensitive

ucs4strnewucs2

Declaration `function ucs4strnewucs2(str: pucs2char): pucs4char;`

Description Converts an UCS-2 null terminated string to an UCS-4 null terminated string

The memory for the buffer is allocated. Use `ucs4strdispose`(10.3) to dispose of the allocated string. The string is null terminated. If str is nil, then this routine returns nil and does not allocate anything.

Parameters **str** The string to convert, UCS-2 encoded

ucs4strnewucs4

Declaration `function ucs4strnewucs4(src: pucs4char): pucs4char;`

Description Allocates and copies an UCS-4 null terminated string

The memory for the buffer is allocated. Use `ucs4strdispose`(10.3) to dispose of the allocated string. The string is copied from src and is null terminated. If the parameter is nil, this routine returns nil and does not allocate anything.

ucs4strpas

Declaration `procedure ucs4strpas(str: pucs4char; var res:array of ucs4char);`

Description Converts a null-terminated UCS-4 string to a Pascal-style UCS-4 string.

ucs4strpastoASCII

Declaration `function ucs4strpastoASCII(str: pucs4char): string;`

Description Converts a null-terminated UCS-4 string to a Pascal-style ASCII encoded string.

If the length is greater than the supported maximum string length, the string is truncated.

Characters that cannot be converted are converted to the default fallback character: DEFAULT_FALLBACK_CHAR

ucs4strpastoISO8859_1

Declaration `function ucs4strpastoISO8859_1(str: pucs4char): string;`

Description Converts a null-terminated UCS-4 string to a Pascal-style ISO 8859-1 encoded string.

If the length is greater than the supported maximum string length, the string is truncated.

Characters that cannot be converted are converted to the default fallback character: DEFAULT_FALLBACK_CHAR

ucs4strpastoUTF8

Declaration `function ucs4strpastoUTF8(str: pucs4char): utf8string;`

Description Converts a null-terminated UCS-4 string to a Pascal-style UTF-8 encoded string.

If the length is greater than the supported maximum string length, the string is truncated.

ucs4strpcopy

Declaration `function ucs4strpcopy(dest: pucs4char; source: array of ucs4char):pucs4char;`

Description Copies a Pascal-style UCS-4 string to a null-terminated UCS-4 string.

This routine does not perform any length checking. If the source string contains some null characters, those nulls are removed from the resulting string.

The destination buffer must have room for at least Length(Source)+1 characters.

ucs4strposISO8859_1

Declaration `function ucs4strposISO8859_1(S: pucs4char; Str2: PChar): pucs4char;`

Description Returns a pointer to the first occurence of an ISO-8859-1 encoded null terminated string in a UCS-4 encoded null terminated string.

ucs4strtrim

Declaration `function ucs4strtrim(const p: pucs4char): pucs4char;`

Description Allocate and copy trimmed an UCS-4 null terminated string

Allocates a new UCS-4 null terminated string, and copies the existing string, avoiding a copy of the whitespace at the start and end of the string

ucs4_concat

Declaration `procedure ucs4_concat(var resultstr: array of ucs4char ;const s1: array of ucs4char; const s2: array of ucs4char);`

Description Concatenates two UCS-4 strings, and gives a resulting UCS-4 string

ucs4_concatascii

Declaration `procedure ucs4_concatascii(var resultstr: array of ucs4char;const s1: array of ucs4char; const s2: string);`

Description Concatenates an UCS-4 string with an ASCII string, and gives a resulting UCS-4 string

ucs4_converttoiso8859_1

Declaration `function ucs4_converttoiso8859_1(const s: array of ucs4char): string;`

Description Converts an UCS-4 string to an ISO-8859-1 string

Parameters **s** The UCS-4 string to convert

Returns The converted string with possible escape characters if a character could not be converted

ucs4_converttoutf8

Declaration `function ucs4_converttoutf8(const src: array of ucs4char): utf8string;`

Description Converts an UCS-4 string to an UTF-8 string

If there is an error (such as reserved special characters), returns an empty string

Parameters **s** The UCS-4 string to convert

Returns The converted string

ucs4_copy

Declaration `procedure ucs4_copy(var resultstr: array of ucs4char; const s: array of ucs4char; index: integer; count: integer);`

Description Returns an UCS-4 substring of an UCS-4 string

ucs4_delete

Declaration `procedure ucs4_delete(var s: array of ucs4char; index: integer; count: integer);`

Description Deletes a substring from a string

ucs4_equal

Declaration `function ucs4_equal(const s1,s2: array of ucs4char): boolean;`

Description Checks if both UCS-4 strings are equal

ucs4_equalascii

Declaration `function ucs4_equalascii(const s1 : array of ucs4char; s2: string): boolean;`

Description Checks if an ASCII string is equal to an UCS-4 string

ucs4_getbasechar

Declaration `function ucs4_getbasechar(c: ucs4char): ucs4char;`

Description Returns the base character representation of a canonical character

. This routine is useful for converted multilangual strings to their ASCII equivalents. It converts an accented characters and replaces them with the base equivalent or returns the input character if there is no base equivalent.

ucs4_getNumericValue

Declaration `function ucs4_getNumericValue(c: ucs4char): longint;`

Description Retrieves the Unicode numeric value of a character

Returns the numeric value of a character that can be used to represent a number.

If a character does not have a numeric value, the routine returns -1. If a character cannot be represented as a positive integer (such as fractional characters), -2 is returned.

ucs4_isdigit

Declaration `function ucs4_isdigit(c: ucs4char): boolean;`

Description Determines if the specified character is a digit character

Represented by Unicode character class Decimal, Nd

ucs4_ishexdigit

Declaration `function ucs4_ishexdigit(c: ucs4char): boolean;`

Description Determines if the specified character is an hex-digit character

ucs4_issupported

Declaration `function ucs4_issupported(s: string): boolean;`

Description Checks if conversion from/to this character set format to/from UCS-4 is supported

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

Returns true if conversion to/from UCS-4 is supported with this character set, otherwise FALSE

ucs4_isterminal

Declaration `function ucs4_isterminal(c: ucs4char): boolean;`

Description Determines if the specified character is a sentence terminating character

ucs4_isvalid

Declaration `function ucs4_isvalid(c: ucs4char): boolean;`

Description Checks if the UCS-4 character is valid

This routine verifies if the UCS-4 character is within the valid ranges of UCS-4 characters, as specified in the Unicode standard 4.0. BOM characters are NOT valid with this routine.

ucs4_iswhitespace

Declaration `function ucs4_iswhitespace(c: ucs4char): boolean;`

Description Determines if the specified character is a whitespace character

ucs4_length

Declaration `function ucs4_length(const s: array of ucs4char): integer;`

Description Returns the current length of an UCS-4 string

ucs4_lowcase

Declaration `function ucs4_lowcase(c: ucs4char): ucs4char;`

Description Converts a character to a lowercase character

This routine only supports the simple form case folding algorithm (e.g full form is not supported).

ucs4_pos

Declaration `function ucs4_pos(const substr: array of ucs4char; const s: array of ucs4char): integer;`

Description Searches for an UCS-4 substring in an UCS-4 string

ucs4_posascii

Declaration `function ucs4_posascii(const substr: string; const s: array of ucs4char): integer;`

Description Searches for an ASCII substring in an UCS-4 string

ucs4_removeaccents

Declaration `procedure ucs4_removeaccents(var resultstr: array of ucs4char;s2: array of ucs4char);`

Description Returns the base string representation of a canonical string

. This routine is useful for converted multilangual strings to their ASCII equivalents. It removes all accented characters and replaces them with their base equivalent.

ucs4_setlength

Declaration `procedure ucs4_setlength(var s: array of ucs4char; l: integer);`

Description Set the new dynamic length of an UCS-4 string

ucs4_trim

Declaration `procedure ucs4_trim(var s: array of ucs4char);`

Description Trims trailing and leading spaces and control characters from an UCS-4 string.

ucs4_trimleft

Declaration `procedure ucs4_trimleft(var s: array of ucs4char);`

Description Trims leading spaces and control characters from an UCS-4 string.

ucs4_trimright

Declaration `procedure ucs4_trimright(var s: array of ucs4char);`

Description Trims trailing spaces and control characters from an UCS-4 string.

ucs4_upcase

Declaration `function ucs4_upcase(c: ucs4char): ucs4char;`

Description Converts a character to an uppercase character

This routine only supports the simple form case folding algorithm (e.g full form is not supported).

utf16_length

Declaration `function utf16_length(const s: array of utf16char): integer;`

Description Returns the current length of an UTF-16 string

utf16_setlength

Declaration `procedure utf16_setlength(var s: array of utf16char; l: integer);`

Description Set the length of an UTF-16 string

utf16_sizeencoding

Declaration `function utf16_sizeencoding(c: utf16char): 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.

utf8islegal

Declaration `function utf8islegal(p: pchar): boolean;`

Description Validates the legality of an UTF-8 null terminated string

Verifies that the UTF-8 encoded strings is encoded in a legal way.

Returns FALSE if the string is illegal, otherwise returns TRUE

utf8strdispose

Declaration `function utf8strdispose(p: pchar): pchar;`

Description Disposes of an UTF-8 null terminated string on the heap

Disposes of a string that was previously allocated with `utf8strnew`, and sets the pointer to nil.

utf8stringdispose

Declaration `procedure utf8stringdispose(var p : putf8shortstring);`

Description Frees an UTF-8 string that was allocated with `utf8strdispose`(10.3)

Verifies if the pointer is different than nil before freeing it.

utf8stringdup

Declaration `function utf8stringdup(const s : string) : putf8shortstring;`

Description Allocates and copies an UTF-8 string to a pointer to an UTF-8 string

.

Even if the length of the string is of zero bytes, the string will still be allocated and returned.

utf8strlcopyucs4

Declaration `function utf8strlcopyucs4(src: pchar; dst: pucs4char; maxlen: integer): pucs4char;`

Description Convert an UTF-8 null terminated string to an UCS-4 null terminated string

This routine converts an UTF-8 null terminared string to an UCS-4 null terminated string that is stored in native byte order, up to length conversion.

Returns nil if there was no error in the conversion

utf8strlen

Declaration `function utf8strlen(s: putf8char): integer;`

Description Returns the length of the string, not counting the null character

If the pointer is nil, the value returned is zero.

utf8strnew

Declaration `function utf8strnew(src: pucs4char): pchar;`

Description Converts an UCS-4 null terminated string to an UTF-8 null terminated string

The memory for the buffer is allocated. Use `utf8strdispose`(10.3) to dispose of the allocated string. The string is null terminated. If the parameter is nil, this routine returns nil and does not allocate anything.

utf8strnewstr

Declaration `function utf8strnewstr(str: utf8string): putf8char;`

Description Converts an UTF-8 string to a null terminated UTF-8 string.

The memory for the storage of the string is allocated by the routine, and the ending null character is also added.

Returns The newly allocated UTF-8 null terminated string

utf8strnewutf8

Declaration `function utf8strnewutf8(src: pchar): pchar;`

Description Allocates and copies an UTF-8 null terminated string

The memory for the buffer is allocated. Use `utf8strdispose`(10.3) to dispose of the allocated string. The string is copied from src and is null terminated. If the parameter is nil, this routine returns nil and does not allocate anything.

utf8strpas

Declaration `function utf8strpas(src: pchar): string;`

Description Converts a null-terminated UTF-8 string to a Pascal-style UTF-8 encoded string.

The string is empty if the pointer was nil.

utf8strpastoASCII

Declaration `function utf8strpastoASCII(src: pchar): string;`

Description Converts a null-terminated UTF-8 string to a Pascal-style ASCII encoded string.

Characters that cannot be converted are converted to the default fallback character: DEFAULT_FALLBACK_CHAR

utf8strpastoISO8859_1

Declaration `function utf8strpastoISO8859_1(src: pchar): string;`

Description Converts a null-terminated UTF-8 string to a Pascal-style ISO 8859-1 encoded string.

Characters that cannot be converted are converted to the default fallback character: DEFAULT_FALLBACK_CHAR

utf8strpastostring

Declaration `function utf8strpastostring(src: pchar; desttype: string): string;`

Description Converts an UTF-8 null terminated string to a string encoded to a different code page

Characters that cannot be converted are converted to the default fallback character: DEFAULT_FALLBACK_CHAR

If the null character string does not fit in the resulting string, it is truncated.

.

Parameters **src** Null terminated UTF-8 encoded string

desttype Encoding type for resulting string

Returns an empty string on error, or a correctly encoded string

utf8_islegal

Declaration `function utf8_islegal(const s: utf8string): boolean;`

Description Returns if the specified UTF-8 string is legal or not

Verifies that the UTF-8 encoded strings is encoded in a legal way.

Returns FALSE if the string is illegal, otherwise returns TRUE

utf8_length

Declaration `function utf8_length(const s: utf8string): integer;`

Description Returns the current length of an UTF-8 string

utf8_setlength

Declaration `procedure utf8_setlength(var s: utf8string; l: integer);`

Description Set the length of an UTF-8 string

utf8_sizeencoding

Declaration `function utf8_sizeencoding(c: utf8char): integer;`

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

10.4 Types

utf8char

Declaration `utf8char = char;`

Description UTF-8 base data type

putf8char

Declaration `putf8char = pchar;`

utf16char

Declaration `utf16char = word;`

Description UTF-16 base data type

ucs4char

Declaration `ucs4char = longword;`

Description UCS-4 base data type

pucs4char

Declaration `pucs4char = ^ucs4char;`

Description UCS-4 null terminated string

ucs2char

Declaration `ucs2char = word;`

Description UCS-2 base data type

pucs2char

Declaration `pucs2char = ^ucs2char;`

ucs2string

Declaration `ucs2string = array[0..MAX_STRING_LENGTH] of ucs2char;`

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

ucs4string

Declaration `ucs4string = array[0..MAX_STRING_LENGTH] of ucs4char;`

Description UCS-4 string declaration. Index 0 contains the active length of the string in characters.

longucs4string

Declaration `longucs4string = array[0..MAX_LONG_STRING_LENGTH] of ucs4char;`

Description UCS-4 long string declaration. Index 0 contains the active length of the string in characters. This type takes a lot more memory than the regular ucs4string and should seldom be used..

plongucs4string

Declaration `plongucs4string = ^longucs4string;`

longucs2string

Declaration `longucs2string = array[0..MAX_LONG_STRING_LENGTH] of ucs2char;`

Description UCS-2 long string declaration. Index 0 contains the active length of the string in characters. This type takes a lot more memory than the regular ucs2string and should seldom be used.

plongucs2strung

Declaration `plongucs2strung = ^longucs2string;`

utf8string

Declaration `utf8string = string;`

Description UTF-8 string declaration. This can either map to a short string or an ansi string depending on the compilation options.

putf8shortstring

Declaration `putf8shortstring = ^shortstring;`

Description UTF-8 string pointer declaration. This is always a shortstring value.

utf16string

Declaration `utf16string = array[0..MAX_STRING_LENGTH] of utf16char;`

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

ucs4strarray

Declaration `ucs4strarray = array[0..MAX_UCS4_CHARS] of ucs4char;`

pucs4strarray

Declaration `pucs4strarray = ^ucs4strarray;`

ucs2strarray

Declaration `ucs2strarray = array[0..MAX_UCS2_CHARS] of ucs2char;`

pucs2strarray

Declaration `pucs2strarray = ^ucs2strarray;`

10.5 Constants

MAX_STRING_LENGTH

Declaration `MAX_STRING_LENGTH = 2048;`

Description Gives the number of characters that can be contained in a string

MAX_LONG_STRING_LENGTH

Declaration `MAX_LONG_STRING_LENGTH = high(word);`

Description Gives the number of characters that can be contained in a longstring

DEFAULT_FALLBACK_CHAR

Declaration `DEFAULT_FALLBACK_CHAR: string[1] = '?';`

Description Default fallback character when conversion of a character does not succeed when converting to a single byte character set.

MAX_UCS4_CHARS

Declaration `MAX_UCS4_CHARS = high(smallint) div (sizeof(ucs4char));`

Description Maximum size of a null-terminated UCS-4 character string

MAX_UCS2_CHARS

Declaration `MAX_UCS2_CHARS = high(smallint) div (sizeof(ucs2char))-1;`

Description Maximum size of a null-terminated UCS-4 character string

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: UCS-4 big endian encoding signature

BOM_UTF32_LE

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

Description Byte order mark: UCS-4 little endian encoding signature

BOM_UTF16_BE

Declaration `BOM_UTF16_BE = #$FE#$FF;`

BOM_UTF16_LE

Declaration `BOM_UTF16_LE = #$FF#$FE;`

10.6 Author

Carl Eric Codre

Chapter 11

Unit utils

11.1 Description

General utilities common to all platforms.

11.2 Overview

`AddDoubleQuotes` Trims and adds double quotes to the string if it contains spaces

`AnsiDirectoryExists` Verifies the existence of a directory

`AnsiFileExists` Verifies the existence of a filename

`boolstr` Convert a boolean value to an ASCII representation

`ChangeFileExt`

`CleanString`

`CompareByte`

`decstr` Convert a value to an ASCII decimal representation

`decstrunsigned` Convert a value to an ASCII decimal representation

`EscapeToPascal`

`FillTo`

`fillwithzero`

`hexstr` Convert a value to an ASCII hexadecimal representation

`LowString` Convert a string to lowercase ASCII

`Printf` Format a string and print it out to the console

`realstr` Convert a value to an ASCII decimal representation

`RemoveDoubleQuotes` Removes the leading and ending double quotes from a string

`removenulls`

`StreamErrorProcedure` Generic stream error procedure

`StrGetNextLine` Returns the next-line in a possible multiple line string

`stringdispose`

`stringdup`

`StrToken` Separates a string into its individual token representations

`SwapLong` Change the endian of a 32-bit value

`SwapWord` Change the endian of a 16-bit value

`Trim` Trim removes leading and trailing spaces and control characters from the given string S

`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`

`ValUnsignedDecimal`

11.3 Functions and Procedures

AddDoubleQuotes

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

Description Trims and adds double quotes to the string if it contains spaces

.

AnsiDirectoryExists

Declaration `Function AnsiDirectoryExists(DName : string): Boolean;`

Description Verifies the existence of a directory

This routine verifies if the directory named can be opened or if it actually exists.

Only works with the current active code page table.

Parameters **DName** Name of the directory to check

Returns FALSE if the directory cannot be opened or if it does not exist.

AnsiFileExists

Declaration `Function AnsiFileExists(const 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.

Only works with the current active code page table.

Parameters **FName** Name of the file to check

Returns FALSE if the file cannot be opened or if it does not exist

boolstr

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.

ChangeFileExt

Declaration `function ChangeFileExt(const FileName, Extension: string): string;`

CleanString

Declaration `function CleanString(const s: string): string;`

Description Cleans up a string of illegal control characters, newlines and tabs. It does the following on the string:

This only works on UTF-8/ASCII/ISO-8859 strings.

1) Removes characters from code #0..#31 at any location in the string. 2) Trims spaces at the begining and end of the string.

CompareByte

Declaration `function CompareByte(buf1,buf2: pchar;len:longint):integer;`

decstr

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.

Parameters **val** Signed 32-bit value to convert

decstrunsigned

Declaration `function decstrunsigned(l : longword;cnt: byte): string;`

Description Convert a value to an ASCII decimal representation

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

Parameters **val** unsigned 32-bit value to convert

EscapeToPascal

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

FillTo

Declaration `function FillTo(s : string; tolength: integer): string;`

fillwithzero

Declaration `function fillwithzero(s: string; newlength: integer): string;`

hexstr

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

Description Convert a value to an ASCII hexadecimal representation

Convert a value to an ASCII hexadecimal representation. All ascii character are returned in upper case letters.

LowString

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

Description Convert a string to lowercase ASCII

Converts a string containing ASCII characters to a string in lower case ASCII characters.

Printf

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.

Parameters

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

realstr

Declaration `function realstr(val : real;cnt : byte) : string;`

Description Convert a value to an ASCII decimal representation

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

Parameters

val Real value to convert

RemoveDoubleQuotes

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

Description Removes the leading and ending double quotes from a string

If there is no double quotes at the beginning or end of the string, it returns the unmodified string.

removenulls

Declaration `function removenulls(const s: string): string;`

Description Remove all null characters from a string.

StreamErrorProcedure

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

Description Generic stream error procedure

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

StrGetNextLine

Declaration `function StrGetNextLine(var Text: String) : String;`

Description Returns the next-line in a possible multiple line string

This routine is used to retrive a line that may contain end of line ASCII characters in one of the standard Operating System formats.

Parameters **Text** Text string to parse, deletes itself

Returns The string, excluding the EOLN characters

stringdispose

Declaration `procedure stringdispose(var p : pShortstring);`

stringdup

Declaration `function stringdup(const s : string) : pShortstring;`

StrToken

Declaration `function StrToken(var Text: String; Delimiter: Char; UseQuotes: boolean) : String;`

Description Separates a string into its individual token representations

When Text returns an empty string, this indicates that all the data has been parsed.

Parameters **Text** Text string to tokenize, deletes itself

Delimiter The character delimiter for tokens

UseQuotes Are Quotes allowed for escaping the delimiter

Returns The current token

SwapLong

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

Description Change the endian of a 32-bit value

SwapWord

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

Description Change the endian of a 16-bit value

Trim

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

Description Trim removes leading and trailing spaces and control characters from the given string S

TrimLeft

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

Description Remove all whitespace from the start of a string

TrimRight

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

Description Remove all whitespace from the end of a string

UpString

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

Description Convert a string to uppercase ASCII

Converts a string containing ASCII characters to a string in upper case ASCII characters.

ValBinary

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

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

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

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.

ValUnsignedDecimal

Declaration `function ValUnsignedDecimal(const S:String; var code:  integer):longword;`

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

11.4 Types

PshortString

Declaration `PshortString = ^ShortString;`

11.5 Constants

EXIT_DOSERROR

Declaration `EXIT_DOSERROR = 2;`

EXIT_ERROR

Declaration `EXIT_ERROR = 1;`

WhiteSpace

Declaration `WhiteSpace = [' ',#10,#13,#9];`

11.6 Author

Carl Eric Codere