

# Common pascal units documentation

Pasdoc

August 15, 2007

# Contents

<b>1</b>	<b>Unit collects</b>	<b>4</b>
1.1	Description . . . . .	4
1.2	Overview . . . . .	4
1.3	Classes, Interfaces, Objects and Records . . . . .	4
1.4	Types . . . . .	9
1.5	Constants . . . . .	10
1.6	Author . . . . .	11
<b>2</b>	<b>Unit crc</b>	<b>12</b>
2.1	Description . . . . .	12
2.2	Overview . . . . .	12
2.3	Functions and Procedures . . . . .	12
2.4	Author . . . . .	14
<b>3</b>	<b>Unit dateutil</b>	<b>15</b>
3.1	Description . . . . .	15
3.2	Overview . . . . .	15
3.3	Classes, Interfaces, Objects and Records . . . . .	17
3.4	Functions and Procedures . . . . .	18
3.5	Types . . . . .	25
3.6	Constants . . . . .	26
3.7	Author . . . . .	26
<b>4</b>	<b>Unit extdos</b>	<b>27</b>
4.1	Description . . . . .	27
4.2	Overview . . . . .	27
4.3	Classes, Interfaces, Objects and Records . . . . .	28
4.4	Functions and Procedures . . . . .	30
4.5	Types . . . . .	35
4.6	Constants . . . . .	36
4.7	Author . . . . .	36

<b>5</b>	<b>Unit fileio</b>	<b>37</b>
5.1	Description . . . . .	37
5.2	Overview . . . . .	37
5.3	Functions and Procedures . . . . .	38
5.4	Author . . . . .	39
<b>6</b>	<b>Unit fs</b>	<b>40</b>
6.1	Description . . . . .	40
6.2	Overview . . . . .	40
6.3	Classes, Interfaces, Objects and Records . . . . .	40
<b>7</b>	<b>Unit ietf</b>	<b>45</b>
7.1	Description . . . . .	45
7.2	Overview . . . . .	45
7.3	Functions and Procedures . . . . .	45
7.4	Author . . . . .	47
<b>8</b>	<b>Unit iso3166</b>	<b>48</b>
8.1	Description . . . . .	48
8.2	Overview . . . . .	48
8.3	Functions and Procedures . . . . .	48
8.4	Author . . . . .	49
<b>9</b>	<b>Unit iso639</b>	<b>50</b>
9.1	Description . . . . .	50
9.2	Overview . . . . .	50
9.3	Functions and Procedures . . . . .	50
9.4	Author . . . . .	52
<b>10</b>	<b>Unit locale</b>	<b>53</b>
10.1	Description . . . . .	53
10.2	Overview . . . . .	53
10.3	Functions and Procedures . . . . .	54
10.4	Constants . . . . .	57
10.5	Author . . . . .	58
<b>11</b>	<b>Unit unicode</b>	<b>59</b>
11.1	Description . . . . .	59
11.2	Overview . . . . .	59
11.3	Classes, Interfaces, Objects and Records . . . . .	62
11.4	Functions and Procedures . . . . .	63
11.5	Types . . . . .	79
11.6	Constants . . . . .	81
11.7	Author . . . . .	102

<b>12 Unit utils</b>	<b>103</b>
12.1 Description . . . . .	103
12.2 Overview . . . . .	103
12.3 Functions and Procedures . . . . .	104
12.4 Types . . . . .	110
12.5 Constants . . . . .	110
12.6 Author . . . . .	110

# Chapter 1

## Unit collects

### 1.1 Description

Collection units

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

### 1.2 Overview

TStackItem record

TStack 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

### 1.3 Classes, Interfaces, Objects and Records

TStackItem record \_\_\_\_\_

Fields

next next: PStackItem;

data data: pointer;

## TStack Object

---

### Hierarchy

TStack > TObject

### Methods

#### init

**Declaration** public constructor init;

#### done

**Declaration** public destructor done;

#### isEmpty

**Declaration** public function isEmpty: boolean;

#### peek

**Declaration** public function peek: pointer;

#### pop

**Declaration** public function pop: pointer;

#### push

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

## 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**

no description available, TExtendedCollection description followsBase collection object

### **Methods**

#### **FreeItem**

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

## **1.4 Types**

### **TItemList** ---

**Declaration** TItemList = Array [0..MaxCollectionSize - 1] Of Pointer;

## **PItemList**

---

**Declaration** PItemList = ^TItemList;

## **PStackItem**

---

**Declaration** PStackItem = ^TStackItem;

## **PStack**

---

**Declaration** PStack = ^TStack;

**Description** Stack object

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

## **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;

## **1.5 Constants**

### **MaxCollectionSize**

---

**Declaration** MaxCollectionSize = 8192;

### **coIndexError**

---

**Declaration** coIndexError = -1;

**coOverflow**

---

**Declaration** `coOverflow = -2;`

## **1.6 Author**

Carl Eric Codere

# Chapter 2

## Unit crc

### 2.1 Description

CRC and checksum generation unit

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

### 2.2 Overview

`UpdateAdler32` Calculates an Adler-32 checksum value

`UpdateCRC` Calculates a standard 16-bit CRC

`UpdateCrc16` Calculates a CRC-16 CCITT value

`UpdateCrc32` Calculates a CRC-32 CCITT value

`UpdateFletcher8` Calculates an 8-bit fletcher checksum value

### 2.3 Functions and Procedures

#### `UpdateAdler32`

---

**Declaration** `function UpdateAdler32(InitAdler: longword; b: byte): longword;`

**Description** Calculates an Adler-32 checksum value

Routine to get the Adler-32 checksum as defined in IETF RFC 1950.

Normally to be compatible with the standard, the first call to this routine should set `InitAdler` to 1, and the final result of the should be taken as is.

**Parameters** `InitAdler` The value of the previous Adler32

`b` The data byte to get the Adler32 of

**Returns** The updated Adler32 value

## UpdateCRC

---

**Declaration** `function UpdateCRC(InitCrc: word; b: byte): word;`

**Description** Calculates a standard 16-bit CRC

Standard CRC-16 bit algorithm as used in the ARC archiver.

The first call to this routine should set **InitCRC** to 0, and the final result of the should be taken as is.

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

**b** The data byte to get the Crc of

**Returns** The updated Crc value

## UpdateCrc16

---

**Declaration** `function UpdateCrc16(InitCrc: word; b: byte): word;`

**Description** Calculates a CRC-16 CCITT value

Routine to get the CRC-16 CCITT value.

Normally to be compatible with the CCITT standards, the first call to this routine should set **InitCRC** to \$FFFF, and the final result of the CRC-16 should be taken as is.

p.s : This has not been verified against hardware.

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

**b** The data byte to get the CRC-16 of

**Returns** The updated CRC-16 value

## UpdateCrc32

---

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

**Description** Calculates a CRC-32 CCITT value

Routine to get the CRC-32 CCITT value.

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

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

**b** The data byte to get the CRC-32 of

**Returns** The updated CRC-32 value

## UpdateFletcher8

---

**Declaration** `function UpdateFletcher8(InitFletcher: word; b: byte): word;`

**Description** Calculates an 8-bit fletcher checksum value

Routine to get the Fletcher 8-bit checksum as defined in IETF RFC 1146

Normally to be compatible with the standard, the first call to this routine should set `InitFletcher` to 0, and the final result of the should be taken as is.

**Parameters** `InitCRC` The value of the previous Adler32

`b` The data byte to get the Adler32 of

**Returns** The updated Adler32 value

## 2.4 Author

Carl Eric Codere

## Chapter 3

# Unit dateutil

### 3.1 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` is not the same as on the Delphi compilers (Internally `TDatetime` is stored as a Julian date) 3. The milliseconds field is only an approximation, and should not be considered as accurate. 4. Because 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).

### 3.2 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 `TDatetime` value.

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

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

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

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



**DaysBetween** Returns the number of days between two specified TDateTime values.

**DecodeDate** Returns Year, Month, and Day values for a TDateTime value.

**DecodeDateTime** Returns Year, Month, Day, Hour, Minute, Second, and Millisecond values for a TDateTime.

**DecodeTime** Breaks a TDateTime 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 TDateTime 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 TDateTime 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 TDateTime value.

**MonthOf** Returns the month of the year represented by a TDateTime value.

**Now** Returns the current date and time.

**SameDate** Indicates whether two TDateTime values represent the same year, month, and day.

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

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

**SecondOf** Returns the second of the minute represented by a TDateTime value.

**Time** Returns the current time.



## Fields

**LowDateTime** LowDateTime: longword;

**HighDateTime** HighDateTime: longword;

## 3.4 Functions and Procedures

### CurrentYear

---

**Declaration** function CurrentYear: word;

**Description** Returns the current year

### Date

---

**Declaration** function Date: TDateTime;

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

### DateOf

---

**Declaration** function DateOf(const AValue: TDateTime): TDateTime;

**Description** Strips the time portion from a TDateTime value.

### DateTimeToStr

---

**Declaration** function DateTimeToStr(DateTime: TDateTime): string;

**Description** Converts a TDateTime 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: TDateTime; utc: boolean): string;

**Description** Converts a TDateTime 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: TDateTime): string;`

**Description** Converts a TDateTime 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: TDateTime): Word;`

**Description** Returns the day of the month represented by a TDateTime value.

## DaysBetween

---

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

**Description** Returns the number of days between two specified TDateTime values.

## DecodeDate

---

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

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

## DecodeDateTime

---

**Declaration** `procedure DecodeDateTime(const AValue: TDateTime; var Year, Month, Day, Hour, Minute, Second, Millisecond: Word);`

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

## DecodeTime

---

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

**Description** Breaks a TDateTime 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: TDateTime;`

**Description** Returns the current time.

## HourOf

---

**Declaration** `function HourOf(const AValue: TDateTime): Word;`

**Description** Returns the hour of the day represented by a TDateTime value.

## IncDay

---

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

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

## IncHour

---

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

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

## IncMilliSecond

---

**Declaration** `function IncMilliSecond(const AValue: TDateTime; const ANumberOfMilliseconds: big_integer_t): TDateTime;`

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

## IncMinute

---

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

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

## IncSecond

---

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

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

## IncWeek

---

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

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

## IsPM

---

**Declaration** `function IsPM(const AValue: TDateTime): Boolean;`

**Description** Indicates whether the time portion of a specified TDateTime 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: TDateTime): Word;`

**Description** Returns the minute of the hour represented by a TDateTime value.

## MonthOf

---

**Declaration** `function MonthOf(const AValue: TDateTime): Word;`

**Description** Returns the month of the year represented by a TDateTime value.

## Now

---

**Declaration** `function Now: TDateTime;`

**Description** Returns the current date and time.

## SameDate

---

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

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

## SameDateTime

---

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

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

## SameTime

---

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

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

## SecondOf

---

**Declaration** `function SecondOf(const AValue: TDateTime): Word;`

**Description** Returns the second of the minute represented by a TDateTime value.

## Time

---

**Declaration** `function Time: TDateTime;`

**Description** Returns the current time.

## TimeOf

---

**Declaration** `function TimeOf(const AValue: TDateTime): Tdatetime;`

**Description** Strips the date portion from a Tdatetime value

## TimeToStr

---

**Declaration** `function TimeToStr(Time: TDateTime): string;`

**Description** Converts a TDateTime 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: TDateTime;`

**Description** Returns a TDateTime value that represents the current date.

## TryEncodeDate

---

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

**Description** Returns a TDateTime 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: TDateTime): Boolean;`

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

## TryEncodeTime

---

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

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

## TryFileTimeToDateTimeExt

---

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

**Description** Converts a Win32 FILETIME to a TDateTime

This routine converts a 64-bit FILETIME format time into a TDateTime 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: TDateTime): Boolean;`

**Description** Converts a string of date in ISO 8601 to a TDateTime 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: TDateTime): Boolean;`

**Description** Converts a string date-time representation to a TDateTime 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: 'YYYYM-MDD;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: TDateTime; var UTC: boolean) : Boolean;`

**Description** Converts a string to a TDateTime 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: 'YYYYM-MDD;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: TDateTime): Boolean;`

**Description** Converts a string time to a TDateTime value with an error default

Supported formats: 1) ISO 8601 time format (complete representation) with optional time-zone 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: TDateTime; var UTC: boolean): boolean;`

**Description** Converts a UNIX time\_t to a TDateTime

This routine converts a UNIX time\_t format time into a TDateTime 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: TDateTime): Word;`

**Description** Returns the year represented by a TDateTime value.

## 3.5 Types

### TDatetime

---

**Declaration** `TDatetime = real;`

**Description** This is the Julian Day number

### float

---

**Declaration** `float = real;`

### platformword

---

**Declaration** `platformword = system.word;`

## 3.6 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;

## 3.7 Author

Carl Eric Codere

# Chapter 4

## Unit extdos

### 4.1 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.

### 4.2 Overview

**TFileAssociation** record Information on file associations for the shell

**TFileStats** record

**TSearchRecExt** record Returned by **FindFirstEx**(4.4) and **FindNextEx**(4.4)

**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**(4.4).

**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**(4.4).

**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

## 4.3 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(4.4)`

## Fields

<b>name</b>	<b>name:</b> <code>utf8string</code> ; Name of the resource on disk
<b>size</b>	<b>size:</b> <code>big_integer_t</code> ; Size of the resource on disk
<b>owner</b>	<b>owner:</b> <code>utf8string</code> ; Owner (User name) of the resource on disk
<b>ctime</b>	<b>ctime:</b> <code>TDateTime</code> ; Creation time of the resource
<b>mtime</b>	<b>mtime:</b> <code>TDateTime</code> ; Last modification time of the resource
<b>atime</b>	<b>atime:</b> <code>TDateTime</code> ; Last access time of the resource
<b>nlink</b>	<b>nlink:</b> <code>integer</code> ; Number of links to resource
<b>attributes</b>	<b>attributes:</b> <code>tresourceattributes</code> ; Attributes for this file
<b>association</b>	<b>association:</b> <code>tfileassociation</code> ; association for this file (operating system)
<b>streamcount</b>	<b>streamcount:</b> <code>integer</code> ; number of parallel streams for this resource
<b>accesses</b>	<b>accesses:</b> <code>integer</code> ; number of file accesses since file's creation
<b>utc</b>	<b>utc:</b> <code>boolean</code> ; indicates if the times are in UTC format, this is always true, unless the filesystem does not support this information.
<b>dev</b>	<b>dev:</b> <code>array[0..127] of char</code> ; Device where this file resides, this value is represented as an hexadecimal null terminated string and is operating system dependent.
<b>ino</b>	<b>ino:</b> <code>array[0..127] of char</code> ; 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(4.4) and FindNextEx(4.4)

### Fields

**Stats**            `Stats:  TFileStats;`  
                   File statistics

**FindHandle**     `FindHandle :  THandle;`  
                   Operating system specific data

**W32FindData**    `W32FindData :  TWin32FindDataW;`

**IncludeAttr**     `IncludeAttr :  longint;`

**SearchAttr**      `SearchAttr:  TResourceAttributes;`

## 4.4 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(4.4).

**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(4.4).

**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: TDateTime): 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: TDateTime): 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: TDateTime): 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 occurred

## 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 occurred

## 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: tdatetime): 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: tdatetime): 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: tdatetime): 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

## 4.5 Types

### resourceattribute

---

**Declaration** `resourceattribute = (...);`

**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

### resourceattributes

---

**Declaration** `resourceattributes = set of resourceattribute;`

## 4.6 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 TDateTime 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 \*\*

## 4.7 Author

Carl Eric Codre

# Chapter 5

## Unit fileio

### 5.1 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.

### 5.2 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

## 5.3 Functions and Procedures

### FileAssign

---

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

**Description** Assign a filename to a file

This is used 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 filemode 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.

## 5.4 Author

Carl Eric Codere



# Chapter 6

## Unit fs

### 6.1 Description

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

### 6.2 Overview

TDOSFileSystem Object  
TWin32FileSystem Object  
TPOSIXFileSystem Object  
TOS2FileSystem Object  
TAmigaFFSFileSystem Object  
TIS09660Level1FileSystem Object  
TIS09660Level2Filesystem Object  
TUDFFilesystem Object  
TJolietFileSystem Object  
THFSPlusFileSystem Object

### 6.3 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 > TObj

### 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 > TObj

### 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 > TObj

### 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

## TUDFFfilesystem Object

---

### Hierarchy

TUDFFfilesystem > 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 7

## Unit ietf

### 7.1 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).

### 7.2 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

### 7.3 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 successful, or FALSE if the URN is malformed

## 7.4 Author

Carl Eric Codere



## Chapter 8

# Unit iso3166

### 8.1 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.

### 8.2 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

### 8.3 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

## **8.4 Author**

Carl Eric Codere

# Chapter 9

## Unit iso639

### 9.1 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](http://www.loc.gov/standards/iso639-2/ISO-639-2_values_8bits.txt)  
The database used is from 2004-10-19.

### 9.2 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

### 9.3 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

## 9.4 Author

Carl Eric Codere

# Chapter 10

## Unit locale

### 10.1 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 Calendar 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/iso-time.html>

### 10.2 Overview

`GetCharEncoding`

`GetISODateString`

`GetISODateStringBasic`

`GetISODateTimeString`

`GetISOTimeString`

`GetISOTimeStringBasic`

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

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

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

MicrosoftCodePageToMIMECharset

MicrosoftLangageCodeToISOCode

UNIXToDateTime

## 10.3 Functions and Procedures

### 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 (:).

## 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



## 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 extended 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.

## 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.

## 10.4 Constants

### CHAR\_ENCODING\_UTF8

---

**Declaration** `CHAR_ENCODING_UTF8 = 0;`

**Description** Character encoding value: UTF-8 storage format

### CHAR\_ENCODING\_UNKNOWN

---

**Declaration** `CHAR_ENCODING_UNKNOWN = -1;`

**Description** Character encoding value: unknown format

### CHAR\_ENCODING\_UTF32BE

---

**Declaration** `CHAR_ENCODING_UTF32BE = 1;`

**Description** Character encoding value: UTF-32 Big endian

### CHAR\_ENCODING\_UTF32LE

---

**Declaration** `CHAR_ENCODING_UTF32LE = 2;`

**Description** Character encoding value: UTF-32 Little endian

### CHAR\_ENCODING\_UTF16LE

---

**Declaration** `CHAR_ENCODING_UTF16LE = 3;`

**Description** Character encoding value: UTF-16 Little endian

### CHAR\_ENCODING\_UTF16BE

---

**Declaration** `CHAR_ENCODING_UTF16BE = 4;`

**Description** Character encoding value: UTF-16 Big endian

### CHAR\_ENCODING\_BYTE

---

**Declaration** `CHAR_ENCODING_BYTE = 5;`

**Description** Character encoding value: One byte per character storage format

## **CHAR\_ENCODING\_UTF16**

---

**Declaration** CHAR\_ENCODING\_UTF16 = 6;

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

## **CHAR\_ENCODING\_UTF32**

---

**Declaration** CHAR\_ENCODING\_UTF32 = 7;

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

## **10.5 Author**

Carl Eric Codere

# Chapter 11

## Unit unicode

### 11.1 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. 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

### 11.2 Overview

TCaseTableInfo record

THexDigitsInfo record

TWhiteSpaceInfo record

TTerminalsInfo record

TNumericInfo record

**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

`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_convertttoiso8859_1` Converts an UCS-4 string to an ISO-8859-1 string

`ucs4_convertttoutf8` 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.lowercase` 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

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

`utf8stringdispose` Frees an UTF-8 string that was allocated with `utf8strdispose(11.4)`

`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

## 11.3 Classes, Interfaces, Objects and Records

**TCaseTableInfo** record

---

### Fields

**lower** lower: `ucs4char`;

**upper** upper: `ucs4char`;

### THexDigitsInfo record

---

#### Fields

**lower** lower: ucs2char;  
**upper** upper: ucs2char;

### TWhiteSpaceInfo record

---

#### Fields

**lower** lower: ucs2char;  
**upper** upper: ucs2char;

### TTerminalsInfo record

---

#### Fields

**lower** lower: ucs2char;  
**upper** upper: ucs2char;

### TNumericInfo record

---

#### Fields

**CodePoint** CodePoint: ucs4char;  
**value** value: longint;

## 11.4 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: ucs4string; 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** **desttype** Indicates the single byte encoding scheme - case-insensitive

**Returns** UNICODE\_ERR\_OK(11.6) if there was no error in the conversion

## ConvertToUCS4

---

**Declaration** `function ConvertToUCS4(source: string; var dest: ucs4string; 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 escape sequences of the form : \uxxxxxxxx where xxxxxxxx is the hex representation of the character, 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(11.6) if there was no error in the conversion

## ConvertUCS2ToUCS4

---

**Declaration** `function ConvertUCS2ToUCS4(src: array of ucs2char; var dst: ucs4string): 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(11.6) if there was no error in the conversion

## ConvertUCS4ToUCS2

---

**Declaration** `function ConvertUCS4ToUCS2(src: array of ucs4char; var dst: ucs2string): 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(11.6) 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(11.6) 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(11.6) if there was no error in the conversion

---

### ConvertUTF16ToUCS4

---

**Declaration** `function ConvertUTF16ToUCS4(src: utf16string; var dst: ucs4string): 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(11.6) if there was no error in the conversion

## ConvertUTF8ToUCS4

---

**Declaration** `function ConvertUTF8ToUCS4(src: utf8string; var dst: ucs4string): 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(11.6)` 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 terminated 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(11.4)` 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(11.4)` 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

## 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(11.4)` 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(11.4)` 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(11.4)` 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:ucs4string);`

**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 escape sequences of the form : `\uxxxxxxxx` where xxxxxxxx is the hex representation of the character.

## 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 escape sequences of the form : `\uxxxxxxxx` where xxxxxxxx is the hex representation of the character.

## ucs4strupastoUTF8

---

**Declaration** `function ucs4strupastoUTF8(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: ucs4string):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 occurrence 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: ucs4string;const s1: ucs4string; 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: ucs4string;const s1: ucs4string; const s2: string);`

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



## ucs4\_converttiso8859\_1

---

**Declaration** `function ucs4_converttiso8859_1(const s: ucs4string): 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: ucs4string): 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: ucs4string; const s: ucs4string; index: integer; count: integer);`

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

## ucs4\_delete

---

**Declaration** `procedure ucs4_delete(var s: ucs4string; index: integer; count: integer);`

**Description** Deletes a substring from a string

## ucs4\_equal

---

**Declaration** `function ucs4_equal(const s1,s2: ucs4string): 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 multilingual 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\_lowercase**

---

**Declaration** `function ucs4_lowercase(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: ucs4string; const s: ucs4string): integer;`

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

## **ucs4\_posascii**

---

**Declaration** `function ucs4_posascii(const substr: string; const s: ucs4string): integer;`

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

## **ucs4\_removeaccents**

---

**Declaration** `procedure ucs4_removeaccents(var resultstr: ucs4string; s2: ucs4string);`

**Description** Returns the base string representation of a canonical string

. This routine is useful for converted multilingual 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: ucs4string);`

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

### **ucs4\_trimleft**

---

**Declaration** `procedure ucs4_trimleft(var s: ucs4string);`

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

### **ucs4\_trimright**

---

**Declaration** `procedure ucs4_trimright(var s: ucs4string);`

**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.

## 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`(11.4)

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 terminated 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(11.4)` 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(11.4)` 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 escape sequences of the form : \uxxxxxxxx where xxxxxxxx is the hex representation of the character.

## 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 escape sequences of the form : \uxxxxxxxx where xxxxxxxx is the hex representation of the character.

## 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 escape sequences of the form : \uxxxxxxxx where xxxxxxxx is the hex representation of the character.

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

## 11.5 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.

## **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;`

## 11.6 Constants

### **MAX\_STRING\_LENGTH**

---

**Declaration** `MAX_STRING_LENGTH = 2048;`

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

### **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 exceeded

### **UNICODE\_ERR\_INCOMPLETE\_CONVERSION**

---

**Declaration** `UNICODE_ERR_INCOMPLETE_CONVERSION = -3;`

**Description** Return status: Some character 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 = #xEF#\$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;

## MAX\_CASETABLE

---

**Declaration** MAX\_CASETABLE =931;

## MAX\_UCS2\_CASETABLE

---

**Declaration** MAX\_UCS2\_CASETABLE =891;

## MAX\_ANSI\_CASETABLE

---

**Declaration** MAX\_ANSI\_CASETABLE =58;

## MAX\_ASCII\_CASETABLE

---

**Declaration** MAX\_ASCII\_CASETABLE =26;

## CaseTable

---

**Declaration** CaseTable: array[1..MAX\_CASETABLE] of TCaseTableInfo = ( (lower: \$0061; upper: \$0041), (lower: \$0062; upper: \$0042), (lower: \$0063; upper: \$0043), (lower: \$0064; upper: \$0044), (lower: \$0065; upper: \$0045), (lower: \$0066; upper: \$0046), (lower: \$0067; upper: \$0047), (lower: \$0068; upper: \$0048), (lower: \$0069; upper: \$0049), (lower: \$006A; upper: \$004A), (lower: \$006B; upper: \$004B), (lower: \$006C; upper: \$004C),

(lower: \$006D; upper: \$004D), (lower: \$006E; upper: \$004E), (lower: \$006F; upper: \$004F), (lower: \$0070; upper: \$0050), (lower: \$0071; upper: \$0051), (lower: \$0072; upper: \$0052), (lower: \$0073; upper: \$0053), (lower: \$0074; upper: \$0054), (lower: \$0075; upper: \$0055), (lower: \$0076; upper: \$0056), (lower: \$0077; upper: \$0057), (lower: \$0078; upper: \$0058), (lower: \$0079; upper: \$0059), (lower: \$007A; upper: \$005A), (lower: \$00B5; upper: \$039C), (lower: \$00E0; upper: \$00C0), (lower: \$00E1; upper: \$00C1), (lower: \$00E2; upper: \$00C2), (lower: \$00E3; upper: \$00C3), (lower: \$00E4; upper: \$00C4), (lower: \$00E5; upper: \$00C5), (lower: \$00E6; upper: \$00C6), (lower: \$00E7; upper: \$00C7), (lower: \$00E8; upper: \$00C8), (lower: \$00E9; upper: \$00C9), (lower: \$00EA; upper: \$00CA), (lower: \$00EB; upper: \$00CB), (lower: \$00EC; upper: \$00CC), (lower: \$00ED; upper: \$00CD), (lower: \$00EE; upper: \$00CE), (lower: \$00EF; upper: \$00CF), (lower: \$00F0; upper: \$00D0), (lower: \$00F1; upper: \$00D1), (lower: \$00F2; upper: \$00D2), (lower: \$00F3; upper: \$00D3), (lower: \$00F4; upper: \$00D4), (lower: \$00F5; upper: \$00D5), (lower: \$00F6; upper: \$00D6), (lower: \$00F8; upper: \$00D8), (lower: \$00F9; upper: \$00D9), (lower: \$00FA; upper: \$00DA), (lower: \$00FB; upper: \$00DB), (lower: \$00FC; upper: \$00DC), (lower: \$00FD; upper: \$00DD), (lower: \$00FE; upper: \$00DE), (lower: \$00FF; upper: \$0178), (lower: \$0101; upper: \$0100), (lower: \$0103; upper: \$0102), (lower: \$0105; upper: \$0104), (lower: \$0107; upper: \$0106), (lower: \$0109; upper: \$0108), (lower: \$010B; upper: \$010A), (lower: \$010D; upper: \$010C), (lower: \$010F; upper: \$010E), (lower: \$0111; upper: \$0110), (lower: \$0113; upper: \$0112), (lower: \$0115; upper: \$0114), (lower: \$0117; upper: \$0116), (lower: \$0119; upper: \$0118), (lower: \$011B; upper: \$011A), (lower: \$011D; upper: \$011C), (lower: \$011F; upper: \$011E), (lower: \$0121; upper: \$0120), (lower: \$0123; upper: \$0122), (lower: \$0125; upper: \$0124), (lower: \$0127; upper: \$0126), (lower: \$0129; upper: \$0128), (lower: \$012B; upper: \$012A), (lower: \$012D; upper: \$012C), (lower: \$012F; upper: \$012E), (lower: \$0131; upper: \$0049), (lower: \$0133; upper: \$0132), (lower: \$0135; upper: \$0134), (lower: \$0137; upper: \$0136), (lower: \$013A; upper: \$0139), (lower: \$013C; upper: \$013B), (lower: \$013E; upper: \$013D), (lower: \$0140; upper: \$013F), (lower: \$0142; upper: \$0141), (lower: \$0144; upper: \$0143), (lower: \$0146; upper: \$0145), (lower: \$0148; upper: \$0147), (lower: \$014B; upper: \$014A), (lower: \$014D; upper: \$014C), (lower: \$014F; upper: \$014E), (lower: \$0151; upper: \$0150), (lower: \$0153; upper: \$0152), (lower: \$0155; upper: \$0154), (lower: \$0157; upper: \$0156), (lower: \$0159; upper: \$0158), (lower: \$015B; upper: \$015A), (lower: \$015D; upper: \$015C), (lower: \$015F; upper: \$015E), (lower: \$0161; upper: \$0160), (lower: \$0163; upper: \$0162), (lower: \$0165; upper: \$0164), (lower: \$0167; upper: \$0166), (lower: \$0169; upper: \$0168), (lower: \$016B; upper: \$016A), (lower: \$016D; upper: \$016C), (lower: \$016F; upper: \$016E), (lower: \$0171; upper: \$0170), (lower: \$0173; upper: \$0172), (lower: \$0175; upper: \$0174), (lower: \$0177; upper: \$0176), (lower: \$017A; upper: \$0179), (lower: \$017C; upper: \$017B), (lower:

\$017E; upper: \$017D), (lower: \$017F; upper: \$0053), (lower: \$0180; upper:  
 \$0243), (lower: \$0183; upper: \$0182), (lower: \$0185; upper: \$0184),  
 (lower: \$0188; upper: \$0187), (lower: \$018C; upper: \$018B), (lower:  
 \$0192; upper: \$0191), (lower: \$0195; upper: \$01F6), (lower: \$0199; upper:  
 \$0198), (lower: \$019A; upper: \$023D), (lower: \$019E; upper: \$0220),  
 (lower: \$01A1; upper: \$01A0), (lower: \$01A3; upper: \$01A2), (lower:  
 \$01A5; upper: \$01A4), (lower: \$01A8; upper: \$01A7), (lower: \$01AD; upper:  
 \$01AC), (lower: \$01B0; upper: \$01AF), (lower: \$01B4; upper: \$01B3),  
 (lower: \$01B6; upper: \$01B5), (lower: \$01B9; upper: \$01B8), (lower:  
 \$01BD; upper: \$01BC), (lower: \$01BF; upper: \$01F7), (lower: \$01C5; upper:  
 \$01C4), (lower: \$01C6; upper: \$01C4), (lower: \$01C8; upper: \$01C7),  
 (lower: \$01C9; upper: \$01C7), (lower: \$01CB; upper: \$01CA), (lower:  
 \$01CC; upper: \$01CA), (lower: \$01CE; upper: \$01CD), (lower: \$01D0; upper:  
 \$01CF), (lower: \$01D2; upper: \$01D1), (lower: \$01D4; upper: \$01D3),  
 (lower: \$01D6; upper: \$01D5), (lower: \$01D8; upper: \$01D7), (lower:  
 \$01DA; upper: \$01D9), (lower: \$01DC; upper: \$01DB), (lower: \$01DD; upper:  
 \$018E), (lower: \$01DF; upper: \$01DE), (lower: \$01E1; upper: \$01E0),  
 (lower: \$01E3; upper: \$01E2), (lower: \$01E5; upper: \$01E4), (lower:  
 \$01E7; upper: \$01E6), (lower: \$01E9; upper: \$01E8), (lower: \$01EB; upper:  
 \$01EA), (lower: \$01ED; upper: \$01EC), (lower: \$01EF; upper: \$01EE),  
 (lower: \$01F2; upper: \$01F1), (lower: \$01F3; upper: \$01F1), (lower:  
 \$01F5; upper: \$01F4), (lower: \$01F9; upper: \$01F8), (lower: \$01FB; upper:  
 \$01FA), (lower: \$01FD; upper: \$01FC), (lower: \$01FF; upper: \$01FE),  
 (lower: \$0201; upper: \$0200), (lower: \$0203; upper: \$0202), (lower:  
 \$0205; upper: \$0204), (lower: \$0207; upper: \$0206), (lower: \$0209; upper:  
 \$0208), (lower: \$020B; upper: \$020A), (lower: \$020D; upper: \$020C),  
 (lower: \$020F; upper: \$020E), (lower: \$0211; upper: \$0210), (lower:  
 \$0213; upper: \$0212), (lower: \$0215; upper: \$0214), (lower: \$0217; upper:  
 \$0216), (lower: \$0219; upper: \$0218), (lower: \$021B; upper: \$021A),  
 (lower: \$021D; upper: \$021C), (lower: \$021F; upper: \$021E), (lower:  
 \$0223; upper: \$0222), (lower: \$0225; upper: \$0224), (lower: \$0227; upper:  
 \$0226), (lower: \$0229; upper: \$0228), (lower: \$022B; upper: \$022A),  
 (lower: \$022D; upper: \$022C), (lower: \$022F; upper: \$022E), (lower:  
 \$0231; upper: \$0230), (lower: \$0233; upper: \$0232), (lower: \$023C; upper:  
 \$023B), (lower: \$0242; upper: \$0241), (lower: \$0247; upper: \$0246),  
 (lower: \$0249; upper: \$0248), (lower: \$024B; upper: \$024A), (lower:  
 \$024D; upper: \$024C), (lower: \$024F; upper: \$024E), (lower: \$0253; upper:  
 \$0181), (lower: \$0254; upper: \$0186), (lower: \$0256; upper: \$0189),  
 (lower: \$0257; upper: \$018A), (lower: \$0259; upper: \$018F), (lower:  
 \$025B; upper: \$0190), (lower: \$0260; upper: \$0193), (lower: \$0263; upper:  
 \$0194), (lower: \$0268; upper: \$0197), (lower: \$0269; upper: \$0196),  
 (lower: \$026B; upper: \$2C62), (lower: \$026F; upper: \$019C), (lower:  
 \$0272; upper: \$019D), (lower: \$0275; upper: \$019F), (lower: \$027D; upper:  
 \$2C64), (lower: \$0280; upper: \$01A6), (lower: \$0283; upper: \$01A9),  
 (lower: \$0288; upper: \$01AE), (lower: \$0289; upper: \$0244), (lower:  
 \$028A; upper: \$01B1), (lower: \$028B; upper: \$01B2), (lower: \$028C; upper:

\$0245), (lower: \$0292; upper: \$01B7), (lower: \$0345; upper: \$0399),  
 (lower: \$037B; upper: \$03FD), (lower: \$037C; upper: \$03FE), (lower:  
 \$037D; upper: \$03FF), (lower: \$03AC; upper: \$0386), (lower: \$03AD; upper:  
 \$0388), (lower: \$03AE; upper: \$0389), (lower: \$03AF; upper: \$038A),  
 (lower: \$03B1; upper: \$0391), (lower: \$03B2; upper: \$0392), (lower:  
 \$03B3; upper: \$0393), (lower: \$03B4; upper: \$0394), (lower: \$03B5; upper:  
 \$0395), (lower: \$03B6; upper: \$0396), (lower: \$03B7; upper: \$0397),  
 (lower: \$03B8; upper: \$0398), (lower: \$03B9; upper: \$0399), (lower:  
 \$03BA; upper: \$039A), (lower: \$03BB; upper: \$039B), (lower: \$03BC; upper:  
 \$039C), (lower: \$03BD; upper: \$039D), (lower: \$03BE; upper: \$039E),  
 (lower: \$03BF; upper: \$039F), (lower: \$03C0; upper: \$03A0), (lower:  
 \$03C1; upper: \$03A1), (lower: \$03C2; upper: \$03A3), (lower: \$03C3; upper:  
 \$03A3), (lower: \$03C4; upper: \$03A4), (lower: \$03C5; upper: \$03A5),  
 (lower: \$03C6; upper: \$03A6), (lower: \$03C7; upper: \$03A7), (lower:  
 \$03C8; upper: \$03A8), (lower: \$03C9; upper: \$03A9), (lower: \$03CA; upper:  
 \$03AA), (lower: \$03CB; upper: \$03AB), (lower: \$03CC; upper: \$038C),  
 (lower: \$03CD; upper: \$038E), (lower: \$03CE; upper: \$038F), (lower:  
 \$03D0; upper: \$0392), (lower: \$03D1; upper: \$0398), (lower: \$03D5; upper:  
 \$03A6), (lower: \$03D6; upper: \$03A0), (lower: \$03D9; upper: \$03D8),  
 (lower: \$03DB; upper: \$03DA), (lower: \$03DD; upper: \$03DC), (lower:  
 \$03DF; upper: \$03DE), (lower: \$03E1; upper: \$03E0), (lower: \$03E3; upper:  
 \$03E2), (lower: \$03E5; upper: \$03E4), (lower: \$03E7; upper: \$03E6),  
 (lower: \$03E9; upper: \$03E8), (lower: \$03EB; upper: \$03EA), (lower:  
 \$03ED; upper: \$03EC), (lower: \$03EF; upper: \$03EE), (lower: \$03F0; upper:  
 \$039A), (lower: \$03F1; upper: \$03A1), (lower: \$03F2; upper: \$03F9),  
 (lower: \$03F5; upper: \$0395), (lower: \$03F8; upper: \$03F7), (lower:  
 \$03FB; upper: \$03FA), (lower: \$0430; upper: \$0410), (lower: \$0431; upper:  
 \$0411), (lower: \$0432; upper: \$0412), (lower: \$0433; upper: \$0413),  
 (lower: \$0434; upper: \$0414), (lower: \$0435; upper: \$0415), (lower:  
 \$0436; upper: \$0416), (lower: \$0437; upper: \$0417), (lower: \$0438; upper:  
 \$0418), (lower: \$0439; upper: \$0419), (lower: \$043A; upper: \$041A),  
 (lower: \$043B; upper: \$041B), (lower: \$043C; upper: \$041C), (lower:  
 \$043D; upper: \$041D), (lower: \$043E; upper: \$041E), (lower: \$043F; upper:  
 \$041F), (lower: \$0440; upper: \$0420), (lower: \$0441; upper: \$0421),  
 (lower: \$0442; upper: \$0422), (lower: \$0443; upper: \$0423), (lower:  
 \$0444; upper: \$0424), (lower: \$0445; upper: \$0425), (lower: \$0446; upper:  
 \$0426), (lower: \$0447; upper: \$0427), (lower: \$0448; upper: \$0428),  
 (lower: \$0449; upper: \$0429), (lower: \$044A; upper: \$042A), (lower:  
 \$044B; upper: \$042B), (lower: \$044C; upper: \$042C), (lower: \$044D; upper:  
 \$042D), (lower: \$044E; upper: \$042E), (lower: \$044F; upper: \$042F),  
 (lower: \$0450; upper: \$0400), (lower: \$0451; upper: \$0401), (lower:  
 \$0452; upper: \$0402), (lower: \$0453; upper: \$0403), (lower: \$0454; upper:  
 \$0404), (lower: \$0455; upper: \$0405), (lower: \$0456; upper: \$0406),  
 (lower: \$0457; upper: \$0407), (lower: \$0458; upper: \$0408), (lower:  
 \$0459; upper: \$0409), (lower: \$045A; upper: \$040A), (lower: \$045B; upper:  
 \$040B), (lower: \$045C; upper: \$040C), (lower: \$045D; upper: \$040D),

(lower: \$045E; upper: \$040E), (lower: \$045F; upper: \$040F), (lower:  
\$0461; upper: \$0460), (lower: \$0463; upper: \$0462), (lower: \$0465; upper:  
\$0464), (lower: \$0467; upper: \$0466), (lower: \$0469; upper: \$0468),  
(lower: \$046B; upper: \$046A), (lower: \$046D; upper: \$046C), (lower:  
\$046F; upper: \$046E), (lower: \$0471; upper: \$0470), (lower: \$0473; upper:  
\$0472), (lower: \$0475; upper: \$0474), (lower: \$0477; upper: \$0476),  
(lower: \$0479; upper: \$0478), (lower: \$047B; upper: \$047A), (lower:  
\$047D; upper: \$047C), (lower: \$047F; upper: \$047E), (lower: \$0481; upper:  
\$0480), (lower: \$048B; upper: \$048A), (lower: \$048D; upper: \$048C),  
(lower: \$048F; upper: \$048E), (lower: \$0491; upper: \$0490), (lower:  
\$0493; upper: \$0492), (lower: \$0495; upper: \$0494), (lower: \$0497; upper:  
\$0496), (lower: \$0499; upper: \$0498), (lower: \$049B; upper: \$049A),  
(lower: \$049D; upper: \$049C), (lower: \$049F; upper: \$049E), (lower:  
\$04A1; upper: \$04A0), (lower: \$04A3; upper: \$04A2), (lower: \$04A5; upper:  
\$04A4), (lower: \$04A7; upper: \$04A6), (lower: \$04A9; upper: \$04A8),  
(lower: \$04AB; upper: \$04AA), (lower: \$04AD; upper: \$04AC), (lower:  
\$04AF; upper: \$04AE), (lower: \$04B1; upper: \$04B0), (lower: \$04B3; upper:  
\$04B2), (lower: \$04B5; upper: \$04B4), (lower: \$04B7; upper: \$04B6),  
(lower: \$04B9; upper: \$04B8), (lower: \$04BB; upper: \$04BA), (lower:  
\$04BD; upper: \$04BC), (lower: \$04BF; upper: \$04BE), (lower: \$04C2; upper:  
\$04C1), (lower: \$04C4; upper: \$04C3), (lower: \$04C6; upper: \$04C5),  
(lower: \$04C8; upper: \$04C7), (lower: \$04CA; upper: \$04C9), (lower:  
\$04CC; upper: \$04CB), (lower: \$04CE; upper: \$04CD), (lower: \$04CF; upper:  
\$04C0), (lower: \$04D1; upper: \$04D0), (lower: \$04D3; upper: \$04D2),  
(lower: \$04D5; upper: \$04D4), (lower: \$04D7; upper: \$04D6), (lower:  
\$04D9; upper: \$04D8), (lower: \$04DB; upper: \$04DA), (lower: \$04DD; upper:  
\$04DC), (lower: \$04DF; upper: \$04DE), (lower: \$04E1; upper: \$04E0),  
(lower: \$04E3; upper: \$04E2), (lower: \$04E5; upper: \$04E4), (lower:  
\$04E7; upper: \$04E6), (lower: \$04E9; upper: \$04E8), (lower: \$04EB; upper:  
\$04EA), (lower: \$04ED; upper: \$04EC), (lower: \$04EF; upper: \$04EE),  
(lower: \$04F1; upper: \$04F0), (lower: \$04F3; upper: \$04F2), (lower:  
\$04F5; upper: \$04F4), (lower: \$04F7; upper: \$04F6), (lower: \$04F9; upper:  
\$04F8), (lower: \$04FB; upper: \$04FA), (lower: \$04FD; upper: \$04FC),  
(lower: \$04FF; upper: \$04FE), (lower: \$0501; upper: \$0500), (lower:  
\$0503; upper: \$0502), (lower: \$0505; upper: \$0504), (lower: \$0507; upper:  
\$0506), (lower: \$0509; upper: \$0508), (lower: \$050B; upper: \$050A),  
(lower: \$050D; upper: \$050C), (lower: \$050F; upper: \$050E), (lower:  
\$0511; upper: \$0510), (lower: \$0513; upper: \$0512), (lower: \$0561; upper:  
\$0531), (lower: \$0562; upper: \$0532), (lower: \$0563; upper: \$0533),  
(lower: \$0564; upper: \$0534), (lower: \$0565; upper: \$0535), (lower:  
\$0566; upper: \$0536), (lower: \$0567; upper: \$0537), (lower: \$0568; upper:  
\$0538), (lower: \$0569; upper: \$0539), (lower: \$056A; upper: \$053A),  
(lower: \$056B; upper: \$053B), (lower: \$056C; upper: \$053C), (lower:  
\$056D; upper: \$053D), (lower: \$056E; upper: \$053E), (lower: \$056F; upper:  
\$053F), (lower: \$0570; upper: \$0540), (lower: \$0571; upper: \$0541),  
(lower: \$0572; upper: \$0542), (lower: \$0573; upper: \$0543), (lower:

\$0574; upper: \$0544), (lower: \$0575; upper: \$0545), (lower: \$0576; upper:  
 \$0546), (lower: \$0577; upper: \$0547), (lower: \$0578; upper: \$0548),  
 (lower: \$0579; upper: \$0549), (lower: \$057A; upper: \$054A), (lower:  
 \$057B; upper: \$054B), (lower: \$057C; upper: \$054C), (lower: \$057D; upper:  
 \$054D), (lower: \$057E; upper: \$054E), (lower: \$057F; upper: \$054F),  
 (lower: \$0580; upper: \$0550), (lower: \$0581; upper: \$0551), (lower:  
 \$0582; upper: \$0552), (lower: \$0583; upper: \$0553), (lower: \$0584; upper:  
 \$0554), (lower: \$0585; upper: \$0555), (lower: \$0586; upper: \$0556),  
 (lower: \$1D7D; upper: \$2C63), (lower: \$1E01; upper: \$1E00), (lower:  
 \$1E03; upper: \$1E02), (lower: \$1E05; upper: \$1E04), (lower: \$1E07; upper:  
 \$1E06), (lower: \$1E09; upper: \$1E08), (lower: \$1E0B; upper: \$1E0A),  
 (lower: \$1E0D; upper: \$1E0C), (lower: \$1E0F; upper: \$1E0E), (lower:  
 \$1E11; upper: \$1E10), (lower: \$1E13; upper: \$1E12), (lower: \$1E15; upper:  
 \$1E14), (lower: \$1E17; upper: \$1E16), (lower: \$1E19; upper: \$1E18),  
 (lower: \$1E1B; upper: \$1E1A), (lower: \$1E1D; upper: \$1E1C), (lower:  
 \$1E1F; upper: \$1E1E), (lower: \$1E21; upper: \$1E20), (lower: \$1E23; upper:  
 \$1E22), (lower: \$1E25; upper: \$1E24), (lower: \$1E27; upper: \$1E26),  
 (lower: \$1E29; upper: \$1E28), (lower: \$1E2B; upper: \$1E2A), (lower:  
 \$1E2D; upper: \$1E2C), (lower: \$1E2F; upper: \$1E2E), (lower: \$1E31; upper:  
 \$1E30), (lower: \$1E33; upper: \$1E32), (lower: \$1E35; upper: \$1E34),  
 (lower: \$1E37; upper: \$1E36), (lower: \$1E39; upper: \$1E38), (lower:  
 \$1E3B; upper: \$1E3A), (lower: \$1E3D; upper: \$1E3C), (lower: \$1E3F; upper:  
 \$1E3E), (lower: \$1E41; upper: \$1E40), (lower: \$1E43; upper: \$1E42),  
 (lower: \$1E45; upper: \$1E44), (lower: \$1E47; upper: \$1E46), (lower:  
 \$1E49; upper: \$1E48), (lower: \$1E4B; upper: \$1E4A), (lower: \$1E4D; upper:  
 \$1E4C), (lower: \$1E4F; upper: \$1E4E), (lower: \$1E51; upper: \$1E50),  
 (lower: \$1E53; upper: \$1E52), (lower: \$1E55; upper: \$1E54), (lower:  
 \$1E57; upper: \$1E56), (lower: \$1E59; upper: \$1E58), (lower: \$1E5B; upper:  
 \$1E5A), (lower: \$1E5D; upper: \$1E5C), (lower: \$1E5F; upper: \$1E5E),  
 (lower: \$1E61; upper: \$1E60), (lower: \$1E63; upper: \$1E62), (lower:  
 \$1E65; upper: \$1E64), (lower: \$1E67; upper: \$1E66), (lower: \$1E69; upper:  
 \$1E68), (lower: \$1E6B; upper: \$1E6A), (lower: \$1E6D; upper: \$1E6C),  
 (lower: \$1E6F; upper: \$1E6E), (lower: \$1E71; upper: \$1E70), (lower:  
 \$1E73; upper: \$1E72), (lower: \$1E75; upper: \$1E74), (lower: \$1E77; upper:  
 \$1E76), (lower: \$1E79; upper: \$1E78), (lower: \$1E7B; upper: \$1E7A),  
 (lower: \$1E7D; upper: \$1E7C), (lower: \$1E7F; upper: \$1E7E), (lower:  
 \$1E81; upper: \$1E80), (lower: \$1E83; upper: \$1E82), (lower: \$1E85; upper:  
 \$1E84), (lower: \$1E87; upper: \$1E86), (lower: \$1E89; upper: \$1E88),  
 (lower: \$1E8B; upper: \$1E8A), (lower: \$1E8D; upper: \$1E8C), (lower:  
 \$1E8F; upper: \$1E8E), (lower: \$1E91; upper: \$1E90), (lower: \$1E93; upper:  
 \$1E92), (lower: \$1E95; upper: \$1E94), (lower: \$1E9B; upper: \$1E90),  
 (lower: \$1EA1; upper: \$1EA0), (lower: \$1EA3; upper: \$1EA2), (lower:  
 \$1EA5; upper: \$1EA4), (lower: \$1EA7; upper: \$1EA6), (lower: \$1EA9; upper:  
 \$1EA8), (lower: \$1EAB; upper: \$1EAA), (lower: \$1EAD; upper: \$1EAC),  
 (lower: \$1EAF; upper: \$1EAE), (lower: \$1EB1; upper: \$1EB0), (lower:  
 \$1EB3; upper: \$1EB2), (lower: \$1EB5; upper: \$1EB4), (lower: \$1EB7; upper:



\$1EB6), (lower: \$1EB9; upper: \$1EB8), (lower: \$1EBB; upper: \$1EBA),  
 (lower: \$1EBD; upper: \$1EBC), (lower: \$1EBF; upper: \$1EBE), (lower:  
 \$1EC1; upper: \$1EC0), (lower: \$1EC3; upper: \$1EC2), (lower: \$1EC5; upper:  
 \$1EC4), (lower: \$1EC7; upper: \$1EC6), (lower: \$1EC9; upper: \$1EC8),  
 (lower: \$1ECB; upper: \$1ECA), (lower: \$1ECD; upper: \$1ECC), (lower:  
 \$1ECF; upper: \$1ECE), (lower: \$1ED1; upper: \$1ED0), (lower: \$1ED3; upper:  
 \$1ED2), (lower: \$1ED5; upper: \$1ED4), (lower: \$1ED7; upper: \$1ED6),  
 (lower: \$1ED9; upper: \$1ED8), (lower: \$1EDB; upper: \$1EDA), (lower:  
 \$1EDD; upper: \$1EDC), (lower: \$1EDF; upper: \$1EDE), (lower: \$1EE1; upper:  
 \$1EE0), (lower: \$1EE3; upper: \$1EE2), (lower: \$1EE5; upper: \$1EE4),  
 (lower: \$1EE7; upper: \$1EE6), (lower: \$1EE9; upper: \$1EE8), (lower:  
 \$1EEB; upper: \$1EEA), (lower: \$1EED; upper: \$1EEC), (lower: \$1EEF; upper:  
 \$1EEE), (lower: \$1EF1; upper: \$1EF0), (lower: \$1EF3; upper: \$1EF2),  
 (lower: \$1EF5; upper: \$1EF4), (lower: \$1EF7; upper: \$1EF6), (lower:  
 \$1EF9; upper: \$1EF8), (lower: \$1F00; upper: \$1F08), (lower: \$1F01; upper:  
 \$1F09), (lower: \$1F02; upper: \$1F0A), (lower: \$1F03; upper: \$1F0B),  
 (lower: \$1F04; upper: \$1F0C), (lower: \$1F05; upper: \$1F0D), (lower:  
 \$1F06; upper: \$1F0E), (lower: \$1F07; upper: \$1F0F), (lower: \$1F10; upper:  
 \$1F18), (lower: \$1F11; upper: \$1F19), (lower: \$1F12; upper: \$1F1A),  
 (lower: \$1F13; upper: \$1F1B), (lower: \$1F14; upper: \$1F1C), (lower:  
 \$1F15; upper: \$1F1D), (lower: \$1F20; upper: \$1F28), (lower: \$1F21; upper:  
 \$1F29), (lower: \$1F22; upper: \$1F2A), (lower: \$1F23; upper: \$1F2B),  
 (lower: \$1F24; upper: \$1F2C), (lower: \$1F25; upper: \$1F2D), (lower:  
 \$1F26; upper: \$1F2E), (lower: \$1F27; upper: \$1F2F), (lower: \$1F30; upper:  
 \$1F38), (lower: \$1F31; upper: \$1F39), (lower: \$1F32; upper: \$1F3A),  
 (lower: \$1F33; upper: \$1F3B), (lower: \$1F34; upper: \$1F3C), (lower:  
 \$1F35; upper: \$1F3D), (lower: \$1F36; upper: \$1F3E), (lower: \$1F37; upper:  
 \$1F3F), (lower: \$1F40; upper: \$1F48), (lower: \$1F41; upper: \$1F49),  
 (lower: \$1F42; upper: \$1F4A), (lower: \$1F43; upper: \$1F4B), (lower:  
 \$1F44; upper: \$1F4C), (lower: \$1F45; upper: \$1F4D), (lower: \$1F51; upper:  
 \$1F59), (lower: \$1F53; upper: \$1F5B), (lower: \$1F55; upper: \$1F5D),  
 (lower: \$1F57; upper: \$1F5F), (lower: \$1F60; upper: \$1F68), (lower:  
 \$1F61; upper: \$1F69), (lower: \$1F62; upper: \$1F6A), (lower: \$1F63; upper:  
 \$1F6B), (lower: \$1F64; upper: \$1F6C), (lower: \$1F65; upper: \$1F6D),  
 (lower: \$1F66; upper: \$1F6E), (lower: \$1F67; upper: \$1F6F), (lower:  
 \$1F70; upper: \$1FBA), (lower: \$1F71; upper: \$1FBB), (lower: \$1F72; upper:  
 \$1FC8), (lower: \$1F73; upper: \$1FC9), (lower: \$1F74; upper: \$1FCA),  
 (lower: \$1F75; upper: \$1FCB), (lower: \$1F76; upper: \$1FDA), (lower:  
 \$1F77; upper: \$1FDB), (lower: \$1F78; upper: \$1FF8), (lower: \$1F79; upper:  
 \$1FF9), (lower: \$1F7A; upper: \$1FEA), (lower: \$1F7B; upper: \$1FEB),  
 (lower: \$1F7C; upper: \$1FFA), (lower: \$1F7D; upper: \$1FFB), (lower:  
 \$1F80; upper: \$1F88), (lower: \$1F81; upper: \$1F89), (lower: \$1F82; upper:  
 \$1F8A), (lower: \$1F83; upper: \$1F8B), (lower: \$1F84; upper: \$1F8C),  
 (lower: \$1F85; upper: \$1F8D), (lower: \$1F86; upper: \$1F8E), (lower:  
 \$1F87; upper: \$1F8F), (lower: \$1F90; upper: \$1F98), (lower: \$1F91; upper:  
 \$1F99), (lower: \$1F92; upper: \$1F9A), (lower: \$1F93; upper: \$1F9B),

(lower: \$1F94; upper: \$1F9C), (lower: \$1F95; upper: \$1F9D), (lower: \$1F96; upper: \$1F9E), (lower: \$1F97; upper: \$1F9F), (lower: \$1FA0; upper: \$1FA8), (lower: \$1FA1; upper: \$1FA9), (lower: \$1FA2; upper: \$1FAA), (lower: \$1FA3; upper: \$1FAB), (lower: \$1FA4; upper: \$1FAC), (lower: \$1FA5; upper: \$1FAD), (lower: \$1FA6; upper: \$1FAE), (lower: \$1FA7; upper: \$1FAF), (lower: \$1FB0; upper: \$1FB8), (lower: \$1FB1; upper: \$1FB9), (lower: \$1FB3; upper: \$1FBC), (lower: \$1FBE; upper: \$0399), (lower: \$1FC3; upper: \$1FCC), (lower: \$1FD0; upper: \$1FD8), (lower: \$1FD1; upper: \$1FD9), (lower: \$1FE0; upper: \$1FE8), (lower: \$1FE1; upper: \$1FE9), (lower: \$1FE5; upper: \$1FEC), (lower: \$1FF3; upper: \$1FFC), (lower: \$214E; upper: \$2132), (lower: \$2170; upper: \$2160), (lower: \$2171; upper: \$2161), (lower: \$2172; upper: \$2162), (lower: \$2173; upper: \$2163), (lower: \$2174; upper: \$2164), (lower: \$2175; upper: \$2165), (lower: \$2176; upper: \$2166), (lower: \$2177; upper: \$2167), (lower: \$2178; upper: \$2168), (lower: \$2179; upper: \$2169), (lower: \$217A; upper: \$216A), (lower: \$217B; upper: \$216B), (lower: \$217C; upper: \$216C), (lower: \$217D; upper: \$216D), (lower: \$217E; upper: \$216E), (lower: \$217F; upper: \$216F), (lower: \$2184; upper: \$2183), (lower: \$24D0; upper: \$24B6), (lower: \$24D1; upper: \$24B7), (lower: \$24D2; upper: \$24B8), (lower: \$24D3; upper: \$24B9), (lower: \$24D4; upper: \$24BA), (lower: \$24D5; upper: \$24BB), (lower: \$24D6; upper: \$24BC), (lower: \$24D7; upper: \$24BD), (lower: \$24D8; upper: \$24BE), (lower: \$24D9; upper: \$24BF), (lower: \$24DA; upper: \$24C0), (lower: \$24DB; upper: \$24C1), (lower: \$24DC; upper: \$24C2), (lower: \$24DD; upper: \$24C3), (lower: \$24DE; upper: \$24C4), (lower: \$24DF; upper: \$24C5), (lower: \$24E0; upper: \$24C6), (lower: \$24E1; upper: \$24C7), (lower: \$24E2; upper: \$24C8), (lower: \$24E3; upper: \$24C9), (lower: \$24E4; upper: \$24CA), (lower: \$24E5; upper: \$24CB), (lower: \$24E6; upper: \$24CC), (lower: \$24E7; upper: \$24CD), (lower: \$24E8; upper: \$24CE), (lower: \$24E9; upper: \$24CF), (lower: \$2C30; upper: \$2C00), (lower: \$2C31; upper: \$2C01), (lower: \$2C32; upper: \$2C02), (lower: \$2C33; upper: \$2C03), (lower: \$2C34; upper: \$2C04), (lower: \$2C35; upper: \$2C05), (lower: \$2C36; upper: \$2C06), (lower: \$2C37; upper: \$2C07), (lower: \$2C38; upper: \$2C08), (lower: \$2C39; upper: \$2C09), (lower: \$2C3A; upper: \$2C0A), (lower: \$2C3B; upper: \$2C0B), (lower: \$2C3C; upper: \$2C0C), (lower: \$2C3D; upper: \$2C0D), (lower: \$2C3E; upper: \$2C0E), (lower: \$2C3F; upper: \$2C0F), (lower: \$2C40; upper: \$2C10), (lower: \$2C41; upper: \$2C11), (lower: \$2C42; upper: \$2C12), (lower: \$2C43; upper: \$2C13), (lower: \$2C44; upper: \$2C14), (lower: \$2C45; upper: \$2C15), (lower: \$2C46; upper: \$2C16), (lower: \$2C47; upper: \$2C17), (lower: \$2C48; upper: \$2C18), (lower: \$2C49; upper: \$2C19), (lower: \$2C4A; upper: \$2C1A), (lower: \$2C4B; upper: \$2C1B), (lower: \$2C4C; upper: \$2C1C), (lower: \$2C4D; upper: \$2C1D), (lower: \$2C4E; upper: \$2C1E), (lower: \$2C4F; upper: \$2C1F), (lower: \$2C50; upper: \$2C20), (lower: \$2C51; upper: \$2C21), (lower: \$2C52; upper: \$2C22), (lower: \$2C53; upper: \$2C23), (lower: \$2C54; upper: \$2C24), (lower: \$2C55; upper: \$2C25), (lower: \$2C56; upper: \$2C26), (lower: \$2C57; upper: \$2C27), (lower:

\$2C58; upper: \$2C28), (lower: \$2C59; upper: \$2C29), (lower: \$2C5A; upper:  
 \$2C2A), (lower: \$2C5B; upper: \$2C2B), (lower: \$2C5C; upper: \$2C2C),  
 (lower: \$2C5D; upper: \$2C2D), (lower: \$2C5E; upper: \$2C2E), (lower:  
 \$2C61; upper: \$2C60), (lower: \$2C65; upper: \$023A), (lower: \$2C66; upper:  
 \$023E), (lower: \$2C68; upper: \$2C67), (lower: \$2C6A; upper: \$2C69),  
 (lower: \$2C6C; upper: \$2C6B), (lower: \$2C76; upper: \$2C75), (lower:  
 \$2C81; upper: \$2C80), (lower: \$2C83; upper: \$2C82), (lower: \$2C85; upper:  
 \$2C84), (lower: \$2C87; upper: \$2C86), (lower: \$2C89; upper: \$2C88),  
 (lower: \$2C8B; upper: \$2C8A), (lower: \$2C8D; upper: \$2C8C), (lower:  
 \$2C8F; upper: \$2C8E), (lower: \$2C91; upper: \$2C90), (lower: \$2C93; upper:  
 \$2C92), (lower: \$2C95; upper: \$2C94), (lower: \$2C97; upper: \$2C96),  
 (lower: \$2C99; upper: \$2C98), (lower: \$2C9B; upper: \$2C9A), (lower:  
 \$2C9D; upper: \$2C9C), (lower: \$2C9F; upper: \$2C9E), (lower: \$2CA1; upper:  
 \$2CA0), (lower: \$2CA3; upper: \$2CA2), (lower: \$2CA5; upper: \$2CA4),  
 (lower: \$2CA7; upper: \$2CA6), (lower: \$2CA9; upper: \$2CA8), (lower:  
 \$2CAB; upper: \$2CAA), (lower: \$2CAD; upper: \$2CAC), (lower: \$2CAF; upper:  
 \$2CAE), (lower: \$2CB1; upper: \$2CB0), (lower: \$2CB3; upper: \$2CB2),  
 (lower: \$2CB5; upper: \$2CB4), (lower: \$2CB7; upper: \$2CB6), (lower:  
 \$2CB9; upper: \$2CB8), (lower: \$2CBB; upper: \$2CBA), (lower: \$2CBD; upper:  
 \$2CBC), (lower: \$2CBF; upper: \$2CBE), (lower: \$2CC1; upper: \$2CC0),  
 (lower: \$2CC3; upper: \$2CC2), (lower: \$2CC5; upper: \$2CC4), (lower:  
 \$2CC7; upper: \$2CC6), (lower: \$2CC9; upper: \$2CC8), (lower: \$2CCB; upper:  
 \$2CCA), (lower: \$2CCD; upper: \$2CCC), (lower: \$2CCF; upper: \$2CCE),  
 (lower: \$2CD1; upper: \$2CD0), (lower: \$2CD3; upper: \$2CD2), (lower:  
 \$2CD5; upper: \$2CD4), (lower: \$2CD7; upper: \$2CD6), (lower: \$2CD9; upper:  
 \$2CD8), (lower: \$2CDB; upper: \$2CDA), (lower: \$2CDD; upper: \$2CDC),  
 (lower: \$2CDF; upper: \$2CDE), (lower: \$2CE1; upper: \$2CE0), (lower:  
 \$2CE3; upper: \$2CE2), (lower: \$2D00; upper: \$10A0), (lower: \$2D01; upper:  
 \$10A1), (lower: \$2D02; upper: \$10A2), (lower: \$2D03; upper: \$10A3),  
 (lower: \$2D04; upper: \$10A4), (lower: \$2D05; upper: \$10A5), (lower:  
 \$2D06; upper: \$10A6), (lower: \$2D07; upper: \$10A7), (lower: \$2D08; upper:  
 \$10A8), (lower: \$2D09; upper: \$10A9), (lower: \$2D0A; upper: \$10AA),  
 (lower: \$2D0B; upper: \$10AB), (lower: \$2D0C; upper: \$10AC), (lower:  
 \$2D0D; upper: \$10AD), (lower: \$2D0E; upper: \$10AE), (lower: \$2D0F; upper:  
 \$10AF), (lower: \$2D10; upper: \$10B0), (lower: \$2D11; upper: \$10B1),  
 (lower: \$2D12; upper: \$10B2), (lower: \$2D13; upper: \$10B3), (lower:  
 \$2D14; upper: \$10B4), (lower: \$2D15; upper: \$10B5), (lower: \$2D16; upper:  
 \$10B6), (lower: \$2D17; upper: \$10B7), (lower: \$2D18; upper: \$10B8),  
 (lower: \$2D19; upper: \$10B9), (lower: \$2D1A; upper: \$10BA), (lower:  
 \$2D1B; upper: \$10BB), (lower: \$2D1C; upper: \$10BC), (lower: \$2D1D; upper:  
 \$10BD), (lower: \$2D1E; upper: \$10BE), (lower: \$2D1F; upper: \$10BF),  
 (lower: \$2D20; upper: \$10C0), (lower: \$2D21; upper: \$10C1), (lower:  
 \$2D22; upper: \$10C2), (lower: \$2D23; upper: \$10C3), (lower: \$2D24; upper:  
 \$10C4), (lower: \$2D25; upper: \$10C5), (lower: \$FF41; upper: \$FF21),  
 (lower: \$FF42; upper: \$FF22), (lower: \$FF43; upper: \$FF23), (lower:  
 \$FF44; upper: \$FF24), (lower: \$FF45; upper: \$FF25), (lower: \$FF46; upper:

```

$FF26), (lower: $FF47; upper: $FF27), (lower: $FF48; upper: $FF28),
(lower: $FF49; upper: $FF29), (lower: $FF4A; upper: $FF2A), (lower:
$FF4B; upper: $FF2B), (lower: $FF4C; upper: $FF2C), (lower: $FF4D; upper:
$FF2D), (lower: $FF4E; upper: $FF2E), (lower: $FF4F; upper: $FF2F),
(lower: $FF50; upper: $FF30), (lower: $FF51; upper: $FF31), (lower:
$FF52; upper: $FF32), (lower: $FF53; upper: $FF33), (lower: $FF54; upper:
$FF34), (lower: $FF55; upper: $FF35), (lower: $FF56; upper: $FF36),
(lower: $FF57; upper: $FF37), (lower: $FF58; upper: $FF38), (lower:
$FF59; upper: $FF39), (lower: $FF5A; upper: $FF3A), (lower: $10428;
upper: $10400), (lower: $10429; upper: $10401), (lower: $1042A; upper:
$10402), (lower: $1042B; upper: $10403), (lower: $1042C; upper: $10404),
(lower: $1042D; upper: $10405), (lower: $1042E; upper: $10406), (lower:
$1042F; upper: $10407), (lower: $10430; upper: $10408), (lower: $10431;
upper: $10409), (lower: $10432; upper: $1040A), (lower: $10433; upper:
$1040B), (lower: $10434; upper: $1040C), (lower: $10435; upper: $1040D),
(lower: $10436; upper: $1040E), (lower: $10437; upper: $1040F), (lower:
$10438; upper: $10410), (lower: $10439; upper: $10411), (lower: $1043A;
upper: $10412), (lower: $1043B; upper: $10413), (lower: $1043C; upper:
$10414), (lower: $1043D; upper: $10415), (lower: $1043E; upper: $10416),
(lower: $1043F; upper: $10417), (lower: $10440; upper: $10418), (lower:
$10441; upper: $10419), (lower: $10442; upper: $1041A), (lower: $10443;
upper: $1041B), (lower: $10444; upper: $1041C), (lower: $10445; upper:
$1041D), (lower: $10446; upper: $1041E), (lower: $10447; upper: $1041F),
(lower: $10448; upper: $10420), (lower: $10449; upper: $10421), (lower:
$1044A; upper: $10422), (lower: $1044B; upper: $10423), (lower: $1044C;
upper: $10424), (lower: $1044D; upper: $10425), (lower: $1044E; upper:
$10426), (lower: $1044F; upper: $10427) );

```

## MAX\_DIGITS

---

Declaration MAX\_DIGITS =290;

## MAX\_UCS2\_DIGITS

---

Declaration MAX\_UCS2\_DIGITS =230;

## MAX\_ANSI\_DIGITS

---

Declaration MAX\_ANSI\_DIGITS =10;

## MAX\_ASCII\_DIGITS

---

Declaration MAX\_ASCII\_DIGITS =10;

## Digits

---

Declaration Digits: array[1..MAX\_DIGITS] of ucs4char = ( \$0030, \$0031, \$0032, \$0033,

```

$0034, $0035, $0036, $0037, $0038, $0039, $0660, $0661, $0662, $0663, $0664,
$0665, $0666, $0667, $0668, $0669, $06F0, $06F1, $06F2, $06F3, $06F4, $06F5,
$06F6, $06F7, $06F8, $06F9, $07C0, $07C1, $07C2, $07C3, $07C4, $07C5, $07C6,
$07C7, $07C8, $07C9, $0966, $0967, $0968, $0969, $096A, $096B, $096C, $096D,
$096E, $096F, $09E6, $09E7, $09E8, $09E9, $09EA, $09EB, $09EC, $09ED, $09EE,
$09EF, $0A66, $0A67, $0A68, $0A69, $0A6A, $0A6B, $0A6C, $0A6D, $0A6E, $0A6F,
$0AE6, $0AE7, $0AE8, $0AE9, $0AEA, $0AEB, $0AEC, $0AED, $0AEE, $0AEF, $0B66,
$0B67, $0B68, $0B69, $0B6A, $0B6B, $0B6C, $0B6D, $0B6E, $0B6F, $0BE6, $0BE7,
$0BE8, $0BE9, $0BEA, $0BEB, $0BEC, $0BED, $0BEE, $0BEF, $0C66, $0C67, $0C68,
$0C69, $0C6A, $0C6B, $0C6C, $0C6D, $0C6E, $0C6F, $0CE6, $0CE7, $0CE8, $0CE9,
$0CEA, $0CEB, $0CEC, $0CED, $0CEE, $0CEF, $0D66, $0D67, $0D68, $0D69, $0D6A,
$0D6B, $0D6C, $0D6D, $0D6E, $0D6F, $0E50, $0E51, $0E52, $0E53, $0E54, $0E55,
$0E56, $0E57, $0E58, $0E59, $0ED0, $0ED1, $0ED2, $0ED3, $0ED4, $0ED5, $0ED6,
$0ED7, $0ED8, $0ED9, $0F20, $0F21, $0F22, $0F23, $0F24, $0F25, $0F26, $0F27,
$0F28, $0F29, $1040, $1041, $1042, $1043, $1044, $1045, $1046, $1047, $1048,
$1049, $17E0, $17E1, $17E2, $17E3, $17E4, $17E5, $17E6, $17E7, $17E8, $17E9,
$1810, $1811, $1812, $1813, $1814, $1815, $1816, $1817, $1818, $1819, $1946,
$1947, $1948, $1949, $194A, $194B, $194C, $194D, $194E, $194F, $19D0, $19D1,
$19D2, $19D3, $19D4, $19D5, $19D6, $19D7, $19D8, $19D9, $1B50, $1B51, $1B52,
$1B53, $1B54, $1B55, $1B56, $1B57, $1B58, $1B59, $FF10, $FF11, $FF12, $FF13,
$FF14, $FF15, $FF16, $FF17, $FF18, $FF19, $104A0, $104A1, $104A2, $104A3,
$104A4, $104A5, $104A6, $104A7, $104A8, $104A9, $1D7CE, $1D7CF, $1D7D0,
$1D7D1, $1D7D2, $1D7D3, $1D7D4, $1D7D5, $1D7D6, $1D7D7, $1D7D8, $1D7D9,
$1D7DA, $1D7DB, $1D7DC, $1D7DD, $1D7DE, $1D7DF, $1D7E0, $1D7E1, $1D7E2,
$1D7E3, $1D7E4, $1D7E5, $1D7E6, $1D7E7, $1D7E8, $1D7E9, $1D7EA, $1D7EB,
$1D7EC, $1D7ED, $1D7EE, $1D7EF, $1D7F0, $1D7F1, $1D7F2, $1D7F3, $1D7F4,
$1D7F5, $1D7F6, $1D7F7, $1D7F8, $1D7F9, $1D7FA, $1D7FB, $1D7FC, $1D7FD,
$1D7FE, $1D7FF );

```

## MAX\_HEXDIGITS

---

**Declaration** MAX\_HEXDIGITS =6;

## MAX\_UCS2\_HEXDIGITS

---

**Declaration** MAX\_UCS2\_HEXDIGITS =6;

## MAX\_ANSI\_HEXDIGITS

---

**Declaration** MAX\_ANSI\_HEXDIGITS =3;

## MAX\_ASCII\_HEXDIGITS

---

**Declaration** MAX\_ASCII\_HEXDIGITS =3;

## HexDigits

---

**Declaration** HexDigits: array[1..MAX\_HEXDIGITS] of THexDigitsInfo = ( (lower: \$0030; upper: \$0039), (lower: \$0041; upper: \$0046), (lower: \$0061; upper: \$0066), (lower: \$FF10; upper: \$FF19), (lower: \$FF21; upper: \$FF26), (lower: \$FF41; upper: \$FF46) );

## MAX\_WHITESPACE

---

**Declaration** MAX\_WHITESPACE =12;

## MAX\_UCS2\_WHITESPACE

---

**Declaration** MAX\_UCS2\_WHITESPACE =12;

## MAX\_ANSI\_WHITESPACE

---

**Declaration** MAX\_ANSI\_WHITESPACE =4;

## MAX\_ASCII\_WHITESPACE

---

**Declaration** MAX\_ASCII\_WHITESPACE =2;

## WhiteSpace

---

**Declaration** WhiteSpace: array[1..MAX\_WHITESPACE] of TWhiteSpaceInfo = ( (lower: \$0009; upper: \$000D), (lower: \$0020; upper: \$0020), (lower: \$0085; upper: \$0085), (lower: \$00A0; upper: \$00A0), (lower: \$1680; upper: \$1680), (lower: \$180E; upper: \$180E), (lower: \$2000; upper: \$200A), (lower: \$2028; upper: \$2028), (lower: \$2029; upper: \$2029), (lower: \$202F; upper: \$202F), (lower: \$205F; upper: \$205F), (lower: \$3000; upper: \$3000) );

## MAX\_TERMINALS

---

**Declaration** MAX\_TERMINALS =30;

## MAX\_UCS2\_TERMINALS

---

**Declaration** MAX\_UCS2\_TERMINALS =30;

## MAX\_ANSI\_TERMINALS

---

**Declaration** MAX\_ANSI\_TERMINALS =3;

## MAX\_ASCII\_TERMINALS

---

**Declaration** MAX\_ASCII\_TERMINALS =3;

## Terminals

---

**Declaration** Terminals: array[1..MAX\_TERMINALS] of TTerminalsInfo = ( (lower: \$0021; upper: \$0021), (lower: \$002E; upper: \$002E), (lower: \$003F; upper: \$003F), (lower: \$055C; upper: \$055C), (lower: \$055E; upper: \$055E), (lower: \$0589; upper: \$0589), (lower: \$061F; upper: \$061F), (lower: \$06D4; upper: \$06D4), (lower: \$0700; upper: \$0702), (lower: \$07F9; upper: \$07F9), (lower: \$0964; upper: \$0965), (lower: \$104A; upper: \$104B), (lower: \$1362; upper: \$1362), (lower: \$1367; upper: \$1368), (lower: \$166E; upper: \$166E), (lower: \$1803; upper: \$1803), (lower: \$1809; upper: \$1809), (lower: \$1944; upper: \$1945), (lower: \$1B5A; upper: \$1B5B), (lower: \$1B5E; upper: \$1B5F), (lower: \$203C; upper: \$203D), (lower: \$2047; upper: \$2049), (lower: \$3002; upper: \$3002), (lower: \$A876; upper: \$A877), (lower: \$FE52; upper: \$FE52), (lower: \$FE56; upper: \$FE57), (lower: \$FF01; upper: \$FF01), (lower: \$FF0E; upper: \$FF0E), (lower: \$FF1F; upper: \$FF1F), (lower: \$FF61; upper: \$FF61) );

## MAX\_NUMERIC\_MAPPINGS

---

**Declaration** MAX\_NUMERIC\_MAPPINGS =831;

## MAX\_NUMERIC\_UCS2\_MAPPINGS

---

**Declaration** MAX\_NUMERIC\_UCS2\_MAPPINGS =532;

## MAX\_NUMERIC\_ANSI\_MAPPINGS

---

**Declaration** MAX\_NUMERIC\_ANSI\_MAPPINGS =16;

## MAX\_NUMERIC\_ASCII\_MAPPINGS

---

**Declaration** MAX\_NUMERIC\_ASCII\_MAPPINGS =10;

## NumericalMappings

---

**Declaration** NumericalMappings: array[1..MAX\_NUMERIC\_MAPPINGS] of TNumericInfo = ( (CodePoint: \$0030; value: 0), (CodePoint: \$0031; value: 1), (CodePoint: \$0032; value: 2), (CodePoint: \$0033; value: 3), (CodePoint: \$0034; value: 4), (CodePoint: \$0035; value: 5), (CodePoint: \$0036; value: 6), (CodePoint: \$0037; value: 7), (CodePoint: \$0038; value: 8), (CodePoint: \$0039; value: 9), (CodePoint: \$00B2; value: 2), (CodePoint: \$00B3; value: 3), (CodePoint: \$00B9; value: 1), (CodePoint: \$00BC; value: -2), (CodePoint: \$00BD; value: -2), (CodePoint: \$00BE; value: -2), (CodePoint: \$0660; value: 0), (CodePoint: \$0661; value: 1), (CodePoint: \$0662; value: 2), (CodePoint: \$0663; value: 3), (CodePoint: \$0664; value: 4), (CodePoint: \$0665; value: 5), (CodePoint: \$0666; value: 6), (CodePoint: \$0667; value: 7), (CodePoint: \$0668; value: 8), (CodePoint: \$0669; value: 9), (CodePoint: \$06F0; value: 0), (CodePoint: \$06F1; value: 1),

[illegible]



(CodePoint: \$0D67; value: 1), (CodePoint: \$0D68; value: 2), (CodePoint: \$0D69; value: 3), (CodePoint: \$0D6A; value: 4), (CodePoint: \$0D6B; value: 5), (CodePoint: \$0D6C; value: 6), (CodePoint: \$0D6D; value: 7), (CodePoint: \$0D6E; value: 8), (CodePoint: \$0D6F; value: 9), (CodePoint: \$0E50; value: 0), (CodePoint: \$0E51; value: 1), (CodePoint: \$0E52; value: 2), (CodePoint: \$0E53; value: 3), (CodePoint: \$0E54; value: 4), (CodePoint: \$0E55; value: 5), (CodePoint: \$0E56; value: 6), (CodePoint: \$0E57; value: 7), (CodePoint: \$0E58; value: 8), (CodePoint: \$0E59; value: 9), (CodePoint: \$0ED0; value: 0), (CodePoint: \$0ED1; value: 1), (CodePoint: \$0ED2; value: 2), (CodePoint: \$0ED3; value: 3), (CodePoint: \$0ED4; value: 4), (CodePoint: \$0ED5; value: 5), (CodePoint: \$0ED6; value: 6), (CodePoint: \$0ED7; value: 7), (CodePoint: \$0ED8; value: 8), (CodePoint: \$0ED9; value: 9), (CodePoint: \$0F20; value: 0), (CodePoint: \$0F21; value: 1), (CodePoint: \$0F22; value: 2), (CodePoint: \$0F23; value: 3), (CodePoint: \$0F24; value: 4), (CodePoint: \$0F25; value: 5), (CodePoint: \$0F26; value: 6), (CodePoint: \$0F27; value: 7), (CodePoint: \$0F28; value: 8), (CodePoint: \$0F29; value: 9), (CodePoint: \$0F2A; value: -2), (CodePoint: \$0F2B; value: -2), (CodePoint: \$0F2C; value: -2), (CodePoint: \$0F2D; value: -2), (CodePoint: \$0F2E; value: -2), (CodePoint: \$0F2F; value: -2), (CodePoint: \$0F30; value: -2), (CodePoint: \$0F31; value: -2), (CodePoint: \$0F32; value: -2), (CodePoint: \$0F33; value: -2), (CodePoint: \$1040; value: 0), (CodePoint: \$1041; value: 1), (CodePoint: \$1042; value: 2), (CodePoint: \$1043; value: 3), (CodePoint: \$1044; value: 4), (CodePoint: \$1045; value: 5), (CodePoint: \$1046; value: 6), (CodePoint: \$1047; value: 7), (CodePoint: \$1048; value: 8), (CodePoint: \$1049; value: 9), (CodePoint: \$1369; value: 1), (CodePoint: \$136A; value: 2), (CodePoint: \$136B; value: 3), (CodePoint: \$136C; value: 4), (CodePoint: \$136D; value: 5), (CodePoint: \$136E; value: 6), (CodePoint: \$136F; value: 7), (CodePoint: \$1370; value: 8), (CodePoint: \$1371; value: 9), (CodePoint: \$1372; value: 10), (CodePoint: \$1373; value: 20), (CodePoint: \$1374; value: 30), (CodePoint: \$1375; value: 40), (CodePoint: \$1376; value: 50), (CodePoint: \$1377; value: 60), (CodePoint: \$1378; value: 70), (CodePoint: \$1379; value: 80), (CodePoint: \$137A; value: 90), (CodePoint: \$137B; value: 100), (CodePoint: \$137C; value: 10000), (CodePoint: \$16EE; value: 17), (CodePoint: \$16EF; value: 18), (CodePoint: \$16F0; value: 19), (CodePoint: \$17E0; value: 0), (CodePoint: \$17E1; value: 1), (CodePoint: \$17E2; value: 2), (CodePoint: \$17E3; value: 3), (CodePoint: \$17E4; value: 4), (CodePoint: \$17E5; value: 5), (CodePoint: \$17E6; value: 6), (CodePoint: \$17E7; value: 7), (CodePoint: \$17E8; value: 8), (CodePoint: \$17E9; value: 9), (CodePoint: \$17F0; value: 0), (CodePoint: \$17F1; value: 1), (CodePoint: \$17F2; value: 2), (CodePoint: \$17F3; value: 3), (CodePoint: \$17F4; value: 4), (CodePoint: \$17F5; value: 5), (CodePoint: \$17F6; value: 6), (CodePoint: \$17F7; value: 7), (CodePoint: \$17F8; value: 8), (CodePoint: \$17F9; value: 9), (CodePoint: \$1810; value: 0), (CodePoint: \$1811; value: 1), (CodePoint: \$1812; value: 2), (CodePoint: \$1813; value: 3), (CodePoint:

\$1814; value: 4), (CodePoint: \$1815; value: 5), (CodePoint: \$1816; value: 6), (CodePoint: \$1817; value: 7), (CodePoint: \$1818; value: 8), (CodePoint: \$1819; value: 9), (CodePoint: \$1946; value: 0), (CodePoint: \$1947; value: 1), (CodePoint: \$1948; value: 2), (CodePoint: \$1949; value: 3), (CodePoint: \$194A; value: 4), (CodePoint: \$194B; value: 5), (CodePoint: \$194C; value: 6), (CodePoint: \$194D; value: 7), (CodePoint: \$194E; value: 8), (CodePoint: \$194F; value: 9), (CodePoint: \$19D0; value: 0), (CodePoint: \$19D1; value: 1), (CodePoint: \$19D2; value: 2), (CodePoint: \$19D3; value: 3), (CodePoint: \$19D4; value: 4), (CodePoint: \$19D5; value: 5), (CodePoint: \$19D6; value: 6), (CodePoint: \$19D7; value: 7), (CodePoint: \$19D8; value: 8), (CodePoint: \$19D9; value: 9), (CodePoint: \$1B50; value: 0), (CodePoint: \$1B51; value: 1), (CodePoint: \$1B52; value: 2), (CodePoint: \$1B53; value: 3), (CodePoint: \$1B54; value: 4), (CodePoint: \$1B55; value: 5), (CodePoint: \$1B56; value: 6), (CodePoint: \$1B57; value: 7), (CodePoint: \$1B58; value: 8), (CodePoint: \$1B59; value: 9), (CodePoint: \$2070; value: 0), (CodePoint: \$2074; value: 4), (CodePoint: \$2075; value: 5), (CodePoint: \$2076; value: 6), (CodePoint: \$2077; value: 7), (CodePoint: \$2078; value: 8), (CodePoint: \$2079; value: 9), (CodePoint: \$2080; value: 0), (CodePoint: \$2081; value: 1), (CodePoint: \$2082; value: 2), (CodePoint: \$2083; value: 3), (CodePoint: \$2084; value: 4), (CodePoint: \$2085; value: 5), (CodePoint: \$2086; value: 6), (CodePoint: \$2087; value: 7), (CodePoint: \$2088; value: 8), (CodePoint: \$2089; value: 9), (CodePoint: \$2153; value: -2), (CodePoint: \$2154; value: -2), (CodePoint: \$2155; value: -2), (CodePoint: \$2156; value: -2), (CodePoint: \$2157; value: -2), (CodePoint: \$2158; value: -2), (CodePoint: \$2159; value: -2), (CodePoint: \$215A; value: -2), (CodePoint: \$215B; value: -2), (CodePoint: \$215C; value: -2), (CodePoint: \$215D; value: -2), (CodePoint: \$215E; value: -2), (CodePoint: \$215F; value: 1), (CodePoint: \$2160; value: 1), (CodePoint: \$2161; value: 2), (CodePoint: \$2162; value: 3), (CodePoint: \$2163; value: 4), (CodePoint: \$2164; value: 5), (CodePoint: \$2165; value: 6), (CodePoint: \$2166; value: 7), (CodePoint: \$2167; value: 8), (CodePoint: \$2168; value: 9), (CodePoint: \$2169; value: 10), (CodePoint: \$216A; value: 11), (CodePoint: \$216B; value: 12), (CodePoint: \$216C; value: 50), (CodePoint: \$216D; value: 100), (CodePoint: \$216E; value: 500), (CodePoint: \$216F; value: 1000), (CodePoint: \$2170; value: 1), (CodePoint: \$2171; value: 2), (CodePoint: \$2172; value: 3), (CodePoint: \$2173; value: 4), (CodePoint: \$2174; value: 5), (CodePoint: \$2175; value: 6), (CodePoint: \$2176; value: 7), (CodePoint: \$2177; value: 8), (CodePoint: \$2178; value: 9), (CodePoint: \$2179; value: 10), (CodePoint: \$217A; value: 11), (CodePoint: \$217B; value: 12), (CodePoint: \$217C; value: 50), (CodePoint: \$217D; value: 100), (CodePoint: \$217E; value: 500), (CodePoint: \$217F; value: 1000), (CodePoint: \$2180; value: 1000), (CodePoint: \$2181; value: 5000), (CodePoint: \$2182; value: 10000), (CodePoint: \$2460; value: 1), (CodePoint: \$2461; value: 2), (CodePoint: \$2462; value: 3), (CodePoint: \$2463; value: 4), (CodePoint: \$2464; value: 5), (CodePoint: \$2465; value:

6), (CodePoint: \$2466; value: 7), (CodePoint: \$2467; value: 8),  
 (CodePoint: \$2468; value: 9), (CodePoint: \$2469; value: 10), (CodePoint:  
 \$246A; value: 11), (CodePoint: \$246B; value: 12), (CodePoint: \$246C;  
 value: 13), (CodePoint: \$246D; value: 14), (CodePoint: \$246E; value:  
 15), (CodePoint: \$246F; value: 16), (CodePoint: \$2470; value: 17),  
 (CodePoint: \$2471; value: 18), (CodePoint: \$2472; value: 19), (CodePoint:  
 \$2473; value: 20), (CodePoint: \$2474; value: 1), (CodePoint: \$2475;  
 value: 2), (CodePoint: \$2476; value: 3), (CodePoint: \$2477; value: 4),  
 (CodePoint: \$2478; value: 5), (CodePoint: \$2479; value: 6), (CodePoint:  
 \$247A; value: 7), (CodePoint: \$247B; value: 8), (CodePoint: \$247C; value:  
 9), (CodePoint: \$247D; value: 10), (CodePoint: \$247E; value: 11),  
 (CodePoint: \$247F; value: 12), (CodePoint: \$2480; value: 13), (CodePoint:  
 \$2481; value: 14), (CodePoint: \$2482; value: 15), (CodePoint: \$2483;  
 value: 16), (CodePoint: \$2484; value: 17), (CodePoint: \$2485; value:  
 18), (CodePoint: \$2486; value: 19), (CodePoint: \$2487; value: 20),  
 (CodePoint: \$2488; value: 1), (CodePoint: \$2489; value: 2), (CodePoint:  
 \$248A; value: 3), (CodePoint: \$248B; value: 4), (CodePoint: \$248C; value:  
 5), (CodePoint: \$248D; value: 6), (CodePoint: \$248E; value: 7),  
 (CodePoint: \$248F; value: 8), (CodePoint: \$2490; value: 9), (CodePoint:  
 \$2491; value: 10), (CodePoint: \$2492; value: 11), (CodePoint: \$2493;  
 value: 12), (CodePoint: \$2494; value: 13), (CodePoint: \$2495; value:  
 14), (CodePoint: \$2496; value: 15), (CodePoint: \$2497; value: 16),  
 (CodePoint: \$2498; value: 17), (CodePoint: \$2499; value: 18), (CodePoint:  
 \$249A; value: 19), (CodePoint: \$249B; value: 20), (CodePoint: \$24EA;  
 value: 0), (CodePoint: \$24EB; value: 11), (CodePoint: \$24EC; value: 12),  
 (CodePoint: \$24ED; value: 13), (CodePoint: \$24EE; value: 14), (CodePoint:  
 \$24EF; value: 15), (CodePoint: \$24F0; value: 16), (CodePoint: \$24F1;  
 value: 17), (CodePoint: \$24F2; value: 18), (CodePoint: \$24F3; value:  
 19), (CodePoint: \$24F4; value: 20), (CodePoint: \$24F5; value: 1),  
 (CodePoint: \$24F6; value: 2), (CodePoint: \$24F7; value: 3), (CodePoint:  
 \$24F8; value: 4), (CodePoint: \$24F9; value: 5), (CodePoint: \$24FA; value:  
 6), (CodePoint: \$24FB; value: 7), (CodePoint: \$24FC; value: 8),  
 (CodePoint: \$24FD; value: 9), (CodePoint: \$24FE; value: 10), (CodePoint:  
 \$24FF; value: 0), (CodePoint: \$2776; value: 1), (CodePoint: \$2777; value:  
 2), (CodePoint: \$2778; value: 3), (CodePoint: \$2779; value: 4),  
 (CodePoint: \$277A; value: 5), (CodePoint: \$277B; value: 6), (CodePoint:  
 \$277C; value: 7), (CodePoint: \$277D; value: 8), (CodePoint: \$277E; value:  
 9), (CodePoint: \$277F; value: 10), (CodePoint: \$2780; value: 1),  
 (CodePoint: \$2781; value: 2), (CodePoint: \$2782; value: 3), (CodePoint:  
 \$2783; value: 4), (CodePoint: \$2784; value: 5), (CodePoint: \$2785; value:  
 6), (CodePoint: \$2786; value: 7), (CodePoint: \$2787; value: 8),  
 (CodePoint: \$2788; value: 9), (CodePoint: \$2789; value: 10), (CodePoint:  
 \$278A; value: 1), (CodePoint: \$278B; value: 2), (CodePoint: \$278C; value:  
 3), (CodePoint: \$278D; value: 4), (CodePoint: \$278E; value: 5),  
 (CodePoint: \$278F; value: 6), (CodePoint: \$2790; value: 7), (CodePoint:  
 \$2791; value: 8), (CodePoint: \$2792; value: 9), (CodePoint: \$2793; value:

10), (CodePoint: \$2CFD; value: -2), (CodePoint: \$3007; value: 0),  
 (CodePoint: \$3021; value: 1), (CodePoint: \$3022; value: 2), (CodePoint:  
 \$3023; value: 3), (CodePoint: \$3024; value: 4), (CodePoint: \$3025; value:  
 5), (CodePoint: \$3026; value: 6), (CodePoint: \$3027; value: 7),  
 (CodePoint: \$3028; value: 8), (CodePoint: \$3029; value: 9), (CodePoint:  
 \$3038; value: 10), (CodePoint: \$3039; value: 20), (CodePoint: \$303A;  
 value: 30), (CodePoint: \$3192; value: 1), (CodePoint: \$3193; value: 2),  
 (CodePoint: \$3194; value: 3), (CodePoint: \$3195; value: 4), (CodePoint:  
 \$3220; value: 1), (CodePoint: \$3221; value: 2), (CodePoint: \$3222; value:  
 3), (CodePoint: \$3223; value: 4), (CodePoint: \$3224; value: 5),  
 (CodePoint: \$3225; value: 6), (CodePoint: \$3226; value: 7), (CodePoint:  
 \$3227; value: 8), (CodePoint: \$3228; value: 9), (CodePoint: \$3229; value:  
 10), (CodePoint: \$3251; value: 21), (CodePoint: \$3252; value: 22),  
 (CodePoint: \$3253; value: 23), (CodePoint: \$3254; value: 24), (CodePoint:  
 \$3255; value: 25), (CodePoint: \$3256; value: 26), (CodePoint: \$3257;  
 value: 27), (CodePoint: \$3258; value: 28), (CodePoint: \$3259; value:  
 29), (CodePoint: \$325A; value: 30), (CodePoint: \$325B; value: 31),  
 (CodePoint: \$325C; value: 32), (CodePoint: \$325D; value: 33), (CodePoint:  
 \$325E; value: 34), (CodePoint: \$325F; value: 35), (CodePoint: \$3280;  
 value: 1), (CodePoint: \$3281; value: 2), (CodePoint: \$3282; value: 3),  
 (CodePoint: \$3283; value: 4), (CodePoint: \$3284; value: 5), (CodePoint:  
 \$3285; value: 6), (CodePoint: \$3286; value: 7), (CodePoint: \$3287; value:  
 8), (CodePoint: \$3288; value: 9), (CodePoint: \$3289; value: 10),  
 (CodePoint: \$32B1; value: 36), (CodePoint: \$32B2; value: 37), (CodePoint:  
 \$32B3; value: 38), (CodePoint: \$32B4; value: 39), (CodePoint: \$32B5;  
 value: 40), (CodePoint: \$32B6; value: 41), (CodePoint: \$32B7; value:  
 42), (CodePoint: \$32B8; value: 43), (CodePoint: \$32B9; value: 44),  
 (CodePoint: \$32BA; value: 45), (CodePoint: \$32BB; value: 46), (CodePoint:  
 \$32BC; value: 47), (CodePoint: \$32BD; value: 48), (CodePoint: \$32BE;  
 value: 49), (CodePoint: \$32BF; value: 50), (CodePoint: \$FF10; value: 0),  
 (CodePoint: \$FF11; value: 1), (CodePoint: \$FF12; value: 2), (CodePoint:  
 \$FF13; value: 3), (CodePoint: \$FF14; value: 4), (CodePoint: \$FF15; value:  
 5), (CodePoint: \$FF16; value: 6), (CodePoint: \$FF17; value: 7),  
 (CodePoint: \$FF18; value: 8), (CodePoint: \$FF19; value: 9), (CodePoint:  
 \$10107; value: 1), (CodePoint: \$10108; value: 2), (CodePoint: \$10109;  
 value: 3), (CodePoint: \$1010A; value: 4), (CodePoint: \$1010B; value: 5),  
 (CodePoint: \$1010C; value: 6), (CodePoint: \$1010D; value: 7), (CodePoint:  
 \$1010E; value: 8), (CodePoint: \$1010F; value: 9), (CodePoint: \$10110;  
 value: 10), (CodePoint: \$10111; value: 20), (CodePoint: \$10112; value:  
 30), (CodePoint: \$10113; value: 40), (CodePoint: \$10114; value: 50),  
 (CodePoint: \$10115; value: 60), (CodePoint: \$10116; value: 70),  
 (CodePoint: \$10117; value: 80), (CodePoint: \$10118; value: 90),  
 (CodePoint: \$10119; value: 100), (CodePoint: \$1011A; value: 200),  
 (CodePoint: \$1011B; value: 300), (CodePoint: \$1011C; value: 400),  
 (CodePoint: \$1011D; value: 500), (CodePoint: \$1011E; value: 600),  
 (CodePoint: \$1011F; value: 700), (CodePoint: \$10120; value: 800),

```

(CodePoint: $10121; value: 900), (CodePoint: $10122; value: 1000),
(CodePoint: $10123; value: 2000), (CodePoint: $10124; value: 3000),
(CodePoint: $10125; value: 4000), (CodePoint: $10126; value: 5000),
(CodePoint: $10127; value: 6000), (CodePoint: $10128; value: 7000),
(CodePoint: $10129; value: 8000), (CodePoint: $1012A; value: 9000),
(CodePoint: $1012B; value: 10000), (CodePoint: $1012C; value: 20000),
(CodePoint: $1012D; value: 30000), (CodePoint: $1012E; value: 40000),
(CodePoint: $1012F; value: 50000), (CodePoint: $10130; value: 60000),
(CodePoint: $10131; value: 70000), (CodePoint: $10132; value: 80000),
(CodePoint: $10133; value: 90000), (CodePoint: $10140; value: -2),
(CodePoint: $10141; value: -2), (CodePoint: $10142; value: 1),
(CodePoint: $10143; value: 5), (CodePoint: $10144; value: 50),
(CodePoint: $10145; value: 500), (CodePoint: $10146; value: 5000),
(CodePoint: $10147; value: 50000), (CodePoint: $10148; value: 5),
(CodePoint: $10149; value: 10), (CodePoint: $1014A; value: 50),
(CodePoint: $1014B; value: 100), (CodePoint: $1014C; value: 500),
(CodePoint: $1014D; value: 1000), (CodePoint: $1014E; value: 5000),
(CodePoint: $1014F; value: 5), (CodePoint: $10150; value: 10),
(CodePoint: $10151; value: 50), (CodePoint: $10152; value: 100),
(CodePoint: $10153; value: 500), (CodePoint: $10154; value: 1000),
(CodePoint: $10155; value: 10000), (CodePoint: $10156; value: 50000),
(CodePoint: $10157; value: 10), (CodePoint: $10158; value: 1),
(CodePoint: $10159; value: 1), (CodePoint: $1015A; value: 1), (CodePoint:
$1015B; value: 2), (CodePoint: $1015C; value: 2), (CodePoint: $1015D;
value: 2), (CodePoint: $1015E; value: 2), (CodePoint: $1015F; value: 5),
(CodePoint: $10160; value: 10), (CodePoint: $10161; value: 10),
(CodePoint: $10162; value: 10), (CodePoint: $10163; value: 10),
(CodePoint: $10164; value: 10), (CodePoint: $10165; value: 30),
(CodePoint: $10166; value: 50), (CodePoint: $10167; value: 50),
(CodePoint: $10168; value: 50), (CodePoint: $10169; value: 50),
(CodePoint: $1016A; value: 100), (CodePoint: $1016B; value: 300),
(CodePoint: $1016C; value: 500), (CodePoint: $1016D; value: 500),
(CodePoint: $1016E; value: 500), (CodePoint: $1016F; value: 500),
(CodePoint: $10170; value: 500), (CodePoint: $10171; value: 1000),
(CodePoint: $10172; value: 5000), (CodePoint: $10173; value: 5),
(CodePoint: $10174; value: 50), (CodePoint: $10175; value: -2),
(CodePoint: $10176; value: -2), (CodePoint: $10177; value: -2),
(CodePoint: $10178; value: -2), (CodePoint: $1018A; value: 0),
(CodePoint: $10320; value: 1), (CodePoint: $10321; value: 5), (CodePoint:
$10322; value: 10), (CodePoint: $10323; value: 50), (CodePoint: $10341;
value: 90), (CodePoint: $1034A; value: 900), (CodePoint: $103D1; value:
1), (CodePoint: $103D2; value: 2), (CodePoint: $103D3; value: 10),
(CodePoint: $103D4; value: 20), (CodePoint: $103D5; value: 100),
(CodePoint: $104A0; value: 0), (CodePoint: $104A1; value: 1), (CodePoint:
$104A2; value: 2), (CodePoint: $104A3; value: 3), (CodePoint: $104A4;
value: 4), (CodePoint: $104A5; value: 5), (CodePoint: $104A6; value: 6),

```

(CodePoint: \$104A7; value: 7), (CodePoint: \$104A8; value: 8), (CodePoint: \$104A9; value: 9), (CodePoint: \$10916; value: 1), (CodePoint: \$10917; value: 10), (CodePoint: \$10918; value: 20), (CodePoint: \$10919; value: 100), (CodePoint: \$10A40; value: 1), (CodePoint: \$10A41; value: 2), (CodePoint: \$10A42; value: 3), (CodePoint: \$10A43; value: 4), (CodePoint: \$10A44; value: 10), (CodePoint: \$10A45; value: 20), (CodePoint: \$10A46; value: 100), (CodePoint: \$10A47; value: 1000), (CodePoint: \$12400; value: 2), (CodePoint: \$12401; value: 3), (CodePoint: \$12402; value: 4), (CodePoint: \$12403; value: 5), (CodePoint: \$12404; value: 6), (CodePoint: \$12405; value: 7), (CodePoint: \$12406; value: 8), (CodePoint: \$12407; value: 9), (CodePoint: \$12408; value: 3), (CodePoint: \$12409; value: 4), (CodePoint: \$1240A; value: 5), (CodePoint: \$1240B; value: 6), (CodePoint: \$1240C; value: 7), (CodePoint: \$1240D; value: 8), (CodePoint: \$1240E; value: 9), (CodePoint: \$1240F; value: 4), (CodePoint: \$12410; value: 5), (CodePoint: \$12411; value: 6), (CodePoint: \$12412; value: 7), (CodePoint: \$12413; value: 8), (CodePoint: \$12414; value: 9), (CodePoint: \$12415; value: 1), (CodePoint: \$12416; value: 2), (CodePoint: \$12417; value: 3), (CodePoint: \$12418; value: 4), (CodePoint: \$12419; value: 5), (CodePoint: \$1241A; value: 6), (CodePoint: \$1241B; value: 7), (CodePoint: \$1241C; value: 8), (CodePoint: \$1241D; value: 9), (CodePoint: \$1241E; value: 1), (CodePoint: \$1241F; value: 2), (CodePoint: \$12420; value: 3), (CodePoint: \$12421; value: 4), (CodePoint: \$12422; value: 5), (CodePoint: \$12423; value: 2), (CodePoint: \$12424; value: 3), (CodePoint: \$12425; value: 3), (CodePoint: \$12426; value: 4), (CodePoint: \$12427; value: 5), (CodePoint: \$12428; value: 6), (CodePoint: \$12429; value: 7), (CodePoint: \$1242A; value: 8), (CodePoint: \$1242B; value: 9), (CodePoint: \$1242C; value: 1), (CodePoint: \$1242D; value: 2), (CodePoint: \$1242E; value: 3), (CodePoint: \$1242F; value: 3), (CodePoint: \$12430; value: 4), (CodePoint: \$12431; value: 5), (CodePoint: \$12434; value: 1), (CodePoint: \$12435; value: 2), (CodePoint: \$12436; value: 3), (CodePoint: \$12437; value: 3), (CodePoint: \$12438; value: 4), (CodePoint: \$12439; value: 5), (CodePoint: \$1243A; value: 3), (CodePoint: \$1243B; value: 3), (CodePoint: \$1243C; value: 4), (CodePoint: \$1243D; value: 4), (CodePoint: \$1243E; value: 4), (CodePoint: \$1243F; value: 4), (CodePoint: \$12440; value: 6), (CodePoint: \$12441; value: 7), (CodePoint: \$12442; value: 7), (CodePoint: \$12443; value: 7), (CodePoint: \$12444; value: 8), (CodePoint: \$12445; value: 8), (CodePoint: \$12446; value: 9), (CodePoint: \$12447; value: 9), (CodePoint: \$12448; value: 9), (CodePoint: \$12449; value: 9), (CodePoint: \$1244A; value: 2), (CodePoint: \$1244B; value: 3), (CodePoint: \$1244C; value: 4), (CodePoint: \$1244D; value: 5), (CodePoint: \$1244E; value: 6), (CodePoint: \$1244F; value: 1), (CodePoint: \$12450; value: 2), (CodePoint: \$12451; value: 3), (CodePoint: \$12452; value: 4), (CodePoint: \$12453; value: 4), (CodePoint: \$12454; value: 5), (CodePoint: \$12455; value: 5), (CodePoint: \$12458; value: 1), (CodePoint: \$12459; value: 2), (CodePoint: \$1245A; value: -2), (CodePoint: \$1245B; value: -2), (CodePoint: \$1245C; value: -2), (CodePoint: \$1245D; value: -2), (CodePoint: \$1245E; value: -2),

```

(CodePoint: $1245F; value: -2), (CodePoint: $12460; value: -2),
(CodePoint: $12461; value: -2), (CodePoint: $12462; value: -2),
(CodePoint: $1D360; value: 1), (CodePoint: $1D361; value: 2), (CodePoint:
$1D362; value: 3), (CodePoint: $1D363; value: 4), (CodePoint: $1D364;
value: 5), (CodePoint: $1D365; value: 6), (CodePoint: $1D366; value: 7),
(CodePoint: $1D367; value: 8), (CodePoint: $1D368; value: 9), (CodePoint:
$1D369; value: 10), (CodePoint: $1D36A; value: 20), (CodePoint: $1D36B;
value: 30), (CodePoint: $1D36C; value: 40), (CodePoint: $1D36D; value:
50), (CodePoint: $1D36E; value: 60), (CodePoint: $1D36F; value: 70),
(CodePoint: $1D370; value: 80), (CodePoint: $1D371; value: 90),
(CodePoint: $1D7CE; value: 0), (CodePoint: $1D7CF; value: 1), (CodePoint:
$1D7D0; value: 2), (CodePoint: $1D7D1; value: 3), (CodePoint: $1D7D2;
value: 4), (CodePoint: $1D7D3; value: 5), (CodePoint: $1D7D4; value: 6),
(CodePoint: $1D7D5; value: 7), (CodePoint: $1D7D6; value: 8), (CodePoint:
$1D7D7; value: 9), (CodePoint: $1D7D8; value: 0), (CodePoint: $1D7D9;
value: 1), (CodePoint: $1D7DA; value: 2), (CodePoint: $1D7DB; value: 3),
(CodePoint: $1D7DC; value: 4), (CodePoint: $1D7DD; value: 5), (CodePoint:
$1D7DE; value: 6), (CodePoint: $1D7DF; value: 7), (CodePoint: $1D7E0;
value: 8), (CodePoint: $1D7E1; value: 9), (CodePoint: $1D7E2; value: 0),
(CodePoint: $1D7E3; value: 1), (CodePoint: $1D7E4; value: 2), (CodePoint:
$1D7E5; value: 3), (CodePoint: $1D7E6; value: 4), (CodePoint: $1D7E7;
value: 5), (CodePoint: $1D7E8; value: 6), (CodePoint: $1D7E9; value: 7),
(CodePoint: $1D7EA; value: 8), (CodePoint: $1D7EB; value: 9), (CodePoint:
$1D7EC; value: 0), (CodePoint: $1D7ED; value: 1), (CodePoint: $1D7EE;
value: 2), (CodePoint: $1D7EF; value: 3), (CodePoint: $1D7F0; value: 4),
(CodePoint: $1D7F1; value: 5), (CodePoint: $1D7F2; value: 6), (CodePoint:
$1D7F3; value: 7), (CodePoint: $1D7F4; value: 8), (CodePoint: $1D7F5;
value: 9), (CodePoint: $1D7F6; value: 0), (CodePoint: $1D7F7; value: 1),
(CodePoint: $1D7F8; value: 2), (CodePoint: $1D7F9; value: 3), (CodePoint:
$1D7FA; value: 4), (CodePoint: $1D7FB; value: 5), (CodePoint: $1D7FC;
value: 6), (CodePoint: $1D7FD; value: 7), (CodePoint: $1D7FE; value: 8),
(CodePoint: $1D7FF; value: 9) );

```

## 11.7 Author

Carl Eric Codre

# Chapter 12

## Unit utils

### 12.1 Description

General utilities common to all platforms.

### 12.2 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



**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**

## 12.3 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 beginning 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

## 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 retrieve 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.

## 12.4 Types

### PshortString

---

**Declaration** `PshortString = ^ShortString;`

## 12.5 Constants

### EXIT\_DOSERROR

---

**Declaration** `EXIT_DOSERROR = 2;`

### EXIT\_ERROR

---

**Declaration** `EXIT_ERROR = 1;`

### WhiteSpace

---

**Declaration** `WhiteSpace = [' ',#10,#13,#9];`

## 12.6 Author

Carl Eric Codere