

ASN1C

ASN.1 Compiler
Version 5.8
C Common Runtime Functions
Reference Manual

The software described in this document is furnished under a license agreement and may be used only in accordance with the terms of this agreement.

Copyright Notice

Copyright © 1997-2005 Objective Systems, Inc.

All Rights Reserved

This document may be distributed in any form, electronic or otherwise, provided that it is distributed in its entirety and that the copyright and this notice are included.

Author's Contact Information:

Comments, suggestions, and inquiries regarding ASN1C may be submitted via electronic mail to info@obj-sys.com.

CHANGE HISTORY

Date	Author	Version	Description
11/16/2005	ED	5.8	Initial version

Table of Contents

TABLE OF CONTENTS	III
C COMMON LIBRARY FUNCTIONS	7
ASN1C C COMMON RUNTIME FUNCTIONS MODULE DOCUMENTATION	8
C RUNTIME COMMON FUNCTIONS	8
<i>Define Documentation</i>	8
MEMORY ALLOCATION MACROS AND FUNCTIONS	9
<i>Detailed Description</i>	9
<i>Defines</i>	10
<i>Typedefs</i>	10
<i>Functions</i>	10
<i>Define Documentation</i>	11
<i>Function Documentation</i>	15
CONTEXT MANAGEMENT FUNCTIONS	16
<i>Detailed Description</i>	16
<i>Functions</i>	16
<i>Function Documentation</i>	16
LINKED LIST UTILITY FUNCTIONS	18
<i>Detailed Description</i>	18
<i>Defines</i>	18
<i>Functions</i>	18
<i>Define Documentation</i>	19
DIAGNOSTIC TRACE FUNCTIONS	23
<i>Detailed Description</i>	23
<i>Functions</i>	23
<i>Function Documentation</i>	23
ERROR FORMATTING AND PRINT FUNCTIONS	26
<i>Detailed Description</i>	26
<i>Functions</i>	26
<i>Function Documentation</i>	26
MEMORY BUFFER MANAGEMENT FUNCTIONS	29
<i>Detailed Description</i>	29
<i>Functions</i>	29
<i>Function Documentation</i>	29
OBJECT IDENTIFIER HELPER FUNCTIONS	32
<i>Detailed Description</i>	32
<i>Functions</i>	32
<i>Function Documentation</i>	32
LINKED LIST AND STACK UTILITY FUNCTIONS	33
<i>Detailed Description</i>	33
<i>Functions</i>	33
<i>Function Documentation</i>	33
REAL HELPER FUNCTIONS	36
<i>Detailed Description</i>	36
<i>Functions</i>	36
<i>Function Documentation</i>	36
FORMATTED PRINTING FUNCTIONS	36
<i>Detailed Description</i>	37
<i>Functions</i>	37
<i>Function Documentation</i>	37

BINARY CODED DECIMAL (BCD) HELPER FUNCTIONS.....	39
<i>Detailed Description</i>	39
<i>Functions</i>	40
<i>Function Documentation</i>	40
CHARACTER STRING CONVERSION FUNCTIONS.....	41
<i>Detailed Description</i>	41
<i>Functions</i>	41
<i>Function Documentation</i>	41
BIG INTEGER HELPER FUNCTIONS.....	46
<i>Detailed Description</i>	46
<i>Functions</i>	46
<i>Function Documentation</i>	46
TIME HELPER FUNCTIONS.....	50
<i>Detailed Description</i>	50
<i>Functions</i>	50
<i>Function Documentation</i>	51
COMPARISON FUNCTIONS.....	53
<i>Detailed Description</i>	53
<i>Functions</i>	53
<i>Function Documentation</i>	54
COMPARISON TO STANDARD OUTPUT FUNCTIONS.....	59
<i>Detailed Description</i>	59
<i>Functions</i>	59
<i>Function Documentation</i>	60
COPY FUNCTIONS.....	64
<i>Detailed Description</i>	64
<i>Functions</i>	64
<i>Function Documentation</i>	64
PRINT VALUES TO STANDARD OUTPUT.....	68
<i>Detailed Description</i>	68
<i>Functions</i>	68
<i>Function Documentation</i>	69
PRINT STREAM STRUCTURE AND FUNCTION DEFINITIONS.....	72
<i>Classes</i>	72
<i>Typedefs</i>	72
<i>Functions</i>	73
<i>Variables</i>	73
<i>Typedef Documentation</i>	73
<i>Function Documentation</i>	73
<i>Variable Documentation</i>	74
PRINT VALUES TO USER SPECIFIED STREAM.....	74
<i>Detailed Description</i>	74
<i>Functions</i>	74
<i>Function Documentation</i>	75
PRINT VALUES TO TEXT BUFFER FUNCTIONS.....	81
<i>Detailed Description</i>	81
<i>Functions</i>	81
<i>Function Documentation</i>	82
TCP/IP AND UDP SOCKET FUNCTIONS.....	87
<i>Functions</i>	87
<i>Function Documentation</i>	88
STREAM STRUCTURE DEFINITIONS.....	91
<i>Classes</i>	91
<i>Defines</i>	91
<i>Typedefs</i>	92
<i>Typedef Documentation</i>	92

LOW-LEVEL UNBUFFERED STREAM FUNCTIONS	93
<i>Detailed Description</i>	93
<i>Functions</i>	93
<i>Function Documentation</i>	94
BUFFERED STREAM FUNCTIONS	97
<i>Detailed Description</i>	97
<i>Functions</i>	97
<i>Function Documentation</i>	98
FILE, SOCKET, AND MEMORY STREAMS	100
<i>Functions</i>	100
<i>Function Documentation</i>	101
ASN1C C COMMON RUNTIME FUNCTIONS CLASS DOCUMENTATION	105
OSRTPRINTSTREAM STRUCT REFERENCE	105
<i>Detailed Description</i>	105
<i>Public Attributes</i>	105
OSRTSTREAM STRUCT REFERENCE	106
<i>Detailed Description</i>	106
<i>Public Attributes</i>	106
<i>Member Data Documentation</i>	106
ASN1C C COMMON RUNTIME FUNCTIONS FILE DOCUMENTATION	108
ASN1TYPE.H FILE REFERENCE	108
<i>Detailed Description</i>	108
<i>Classes</i>	108
<i>Defines</i>	109
<i>Typedefs</i>	113
<i>Enumerations</i>	114
<i>Functions</i>	114
RTCOMPARE.H FILE REFERENCE	118
<i>Detailed Description</i>	118
<i>Functions</i>	118
RTCOPY.H FILE REFERENCE	120
<i>Detailed Description</i>	120
<i>Defines</i>	120
<i>Functions</i>	120
RTMEMORY.H FILE REFERENCE	121
<i>Detailed Description</i>	121
<i>Defines</i>	121
<i>Typedefs</i>	121
<i>Functions</i>	121
RTPRINT.H FILE REFERENCE	123
<i>Detailed Description</i>	123
<i>Functions</i>	123
RTPRINTSTREAM.H FILE REFERENCE	124
<i>Detailed Description</i>	124
<i>Classes</i>	124
<i>Typedefs</i>	124
<i>Functions</i>	124
<i>Variables</i>	124
RTPRINTTOSTREAM.H FILE REFERENCE	125
<i>Detailed Description</i>	125
<i>Functions</i>	125
RTPRINTTOSTRING.H FILE REFERENCE	126
<i>Detailed Description</i>	126
<i>Functions</i>	126

RTSOCKET.H FILE REFERENCE	128
<i>Detailed Description</i>	128
<i>Defines</i>	128
<i>Typedefs</i>	128
<i>Functions</i>	128
<i>Typedef Documentation</i>	128
RTSTREAM.H FILE REFERENCE.....	130
<i>Detailed Description</i>	130
<i>Classes</i>	130
<i>Defines</i>	130
<i>Typedefs</i>	130
<i>Functions</i>	130
INDEX	132

C Common Library Functions

The **C run-time common library** contains common C functions used by the encoding rules (BER/DER, PER, and XER) low-level encode/decode functions. These functions are identified by their *rt* prefixes. The categories of functions provided are as follows:

- Memory Allocation macros and functions handle memory management for the ASN1C run-time.
- Context Management functions handle the allocation, initialization, and destruction of ASN.1 context variables (variables of type ASN1CTXT) that handle the working data used during encoding or decoding a message.
- Diagnostic Trace functions allow the output of trace messages to standard output that trace the execution of compiler generated functions.
- Error Formatting and Print functions allow information about the encode/decode errors to be added to a context block structure and printed out.
- Memory Buffer Management functions handle the allocation, expansion, and de-allocation of dynamic memory buffers used by some encode/decode functions.
- Object Identifier Helper functions provide assistance in working with the object identifier ASN.1 type.
- Linked List and Stack Utility functions are used to maintain linked lists and stacks used within the ASN.1 run-time library functions.
- REAL Helper functions - REAL helper functions provide assistance in working with the REAL ASN.1 type. Two functions are provided to obtain the plus and minus infinity special values.
- Formatted Printing functions allow raw ASN.1 data fields to be formatted and printed to standard output and other output devices.
- Binary Coded Decimal (BCD) helper functions provide assistance in working with BCD numbers.
- Character String Conversion functions convert between standard null-terminated C strings and different ASN.1 string types.
- Big Integer Helper functions are arbitrary-precision integer manipulating functions used to maintain big integers used within the ASN.1 run-time functions.
- Comparison functions allow comparison of the values of primitive ASN.1 types. They make it possible to compare complex structures and determine what elements within those structures are different.
- Comparison to Standard Output functions do the same actions as the other comparison functions, but print the comparison results to standard output instead of to the buffer.
- Copy functions - This group of functions allows copying values of primitive ASN.1 types.
- Print Values to Standard Output functions print the output in a "name=value" format, where the value format is obtained by calling one of the ToString functions with the given value.

ASN1C C Common Runtime Functions Module Documentation

C Runtime Common Functions

Define Documentation

#define HEXCHARTONIBBLE(ch, b)

```
Value: if (ch >= '0' && ch <= '9') b = (unsigned char)(ch - '0'); \
else if (ch >= 'a' && ch <= 'f') b = (unsigned char)((ch - 'a') + 10); \
else if (ch >= 'A' && ch <= 'F') b = (unsigned char)((ch - 'A') + 10); \
else b = 0xFF;
```

#define NIBBLETOHEXCHAR(b, ch)

```
Value: if (b >= 0 && b <= 9) ch = (char)(b + '0'); \
else if (b >= 0x0a && b <= 0x0f) ch = (char)((b - 10) + 'a'); \
else ch = '?';
```

Function Documentation

int rtClearBit (OSOCKET * *pBits*, int *numbits*, int *bitIndex*)

This function clears the specified bit in the bit string.

Parameters:

pBits Pointer to octets of bit string.

numbits Number of bits in the bit string.

bitIndex Index of bit to be cleared. The bit with index 0 is a most significant bit in the octet with index 0.

Returns:

If successful, returns the previous state of bit. If bit was set the return value is positive, if bit was not set the return value is zero. Otherwise, return value is an error code:

- ASN_E_OUTOFBND = bitIndex is out of bound (see errno)

int rtFileReadBinary (ASN1CTXT * *pctxt*, const char * *filePath*, OSOCKET ** *ppMsgBuf*, size_t * *pLength*)

This function reads the entire contents of a binary file into memory. A memory buffer is allocated for the file contents using the ASN1C run-time memory management functions.

Parameters:

pctxt Pointer to context block structure.

filePath Complete file path name of file to read.

ppMsgBuf Pointer to message buffer to receive allocated memory pointer.

pLength Pointer to integer to receive length of data read.

Returns:

Completion status of operation:

- 0 (ASN_OK) = success,
- ASN_E_FILNOTFOU = file not found
- ASN_E_FILEREAD = file read error (see errno)

const char* rtGetLibInfo ()

Returns information string describing the library. The string contains name of library, its version and flags used for building the library.

Returns:

Information string

int rtGetLibVersion ()

Returns numeric version of run-time library. The format of version is as follows: MmP, where: M - major version number; m - minor version number; p - patch release number. For example, the value 581 means the version 5.81.

Returns:

Version of run-time library in numeric format.

int rtSetBit (OSOCKET * *pBits*, int *numbits*, int *bitIndex*)

This function sets specified bit in the bit string.

Parameters:

pBits Pointer to octets of bit string.

numbits Number of bits in the bit string.

bitIndex Index of bit to be set. The bit with index 0 is a most significant bit in the octet with index 0.

Returns:

If successful, returns the previous state of bit. If bit was set the return value is positive, if bit was not set the return value is zero. Otherwise, return value is an error code:

- ASN_E_OUTOFBND = bitIndex is out of bound (see errno)

OSBOOL rtTestBit (const OSOCKET * *pBits*, int *numbits*, int *bitIndex*)

This function tests specified bit in the bit string.

Parameters:

pBits Pointer to octets of bit string.

numbits Number of bits in the bit string.

bitIndex Index of bit to be tested. The bit with index 0 is a most significant bit in the octet with index 0.

Returns:

True if bit set or false if not set or array index is beyond range of number of bits in the string.

Memory Allocation Macros and Functions

Detailed Description

Memory allocation functions and macros handle memory management for the ASN1C run-time. Special algorithms are used for allocation and deallocation of memory to improve the run-time performance.

Defines

- #define **ALLOC_ASN1ARRAY**(pctx, pseqof, type)
- #define **ALLOC_ASN1ARRAY1**(pctx, pseqof, type)
- #define **ALLOC_ASN1ARRAY2**(pctx, n, type)
- #define **ALLOC_ASN1ELEM**(pctx, type) (type*) rtMemHeapAllocZ (&(pctx)->pTypeMemHeap, sizeof(type))
- #define **ASN1ARRAYSIZE**(x) (sizeof(x)/sizeof(x[0]))
- #define **ASN1MALLOC**(pctx, nbytes) rtMemHeapAlloc(&(pctx)->pTypeMemHeap, nbytes)
- #define **ASN1REALLOC**(pctx, pmem, nbytes) rtMemHeapRealloc(&(pctx)->pTypeMemHeap, pmem, nbytes)
- #define **REALLOC_ASN1ARRAY**(pctx, pseqof, type)
- #define **ASN1MEMFREE**(pctx) rtMemHeapFreeAll(&(pctx)->pTypeMemHeap)
- #define **ASN1MEMFREEPTR**(pctx, pmem) rtMemHeapFreePtr(&(pctx)->pTypeMemHeap, (void*)pmem)
- #define **ASN1MEMRESET**(pctx) rtMemHeapReset(&(pctx)->pTypeMemHeap)
- #define **rtMemAlloc**(pctx, nbytes) rtMemHeapAlloc(&(pctx)->pTypeMemHeap, nbytes)
- #define **rtMemAllocType**(pctx, ctype) (ctype*)rtMemHeapAlloc(&(pctx)->pTypeMemHeap, sizeof(ctype))
- #define **rtMemAllocZ**(pctx, nbytes) rtMemHeapAllocZ(&(pctx)->pTypeMemHeap, nbytes)
- #define **rtMemAllocTypeZ**(pctx, ctype) (ctype*)rtMemHeapAllocZ(&(pctx)->pTypeMemHeap, sizeof(ctype))
- #define **rtMemRealloc**(pctx, mem_p, nbytes) rtMemHeapRealloc(&(pctx)->pTypeMemHeap, (void*)mem_p, nbytes)
- #define **rtMemFreePtr**(pctx, mem_p)
- #define **rtMemFree**(pctx) rtMemHeapFreeAll(&(pctx)->pTypeMemHeap)
- #define **rtMemReset**(pctx) rtMemHeapReset(&(pctx)->pTypeMemHeap)
- #define **OSCDECL**

Typedefs

- typedef void *OSCDECL * **OSMallocFunc** (size_t size)
- typedef void *OSCDECL * **OSReallocFunc** (void *ptr, size_t size)

Functions

- typedef void (OSCDECL *OSFreeFunc)(void *ptr)
- void **rtMemHeapAddRef** (void **ppvMemHeap)
- void * **rtMemHeapAlloc** (void **ppvMemHeap, size_t nbytes)
- void * **rtMemHeapAllocZ** (void **ppvMemHeap, size_t nbytes)
- int **rtMemHeapCheckPtr** (void **ppvMemHeap, void *mem_p)
- int **rtMemHeapCreate** (void **ppvMemHeap)
- void **rtMemHeapFreeAll** (void **ppvMemHeap)
- void **rtMemHeapFreePtr** (void **ppvMemHeap, void *mem_p)
- void * **rtMemHeapMarkSaved** (void **ppvMemHeap, ASN1ConstVoidPtr mem_p, ASN1BOOL saved)
- void * **rtMemHeapRealloc** (void **ppvMemHeap, void *mem_p, size_t nbytes)
- void **rtMemHeapRelease** (void **ppvMemHeap)
- void **rtMemHeapReset** (void **ppvMemHeap)
- void **rtMemHeapSetProperty** (void **ppvMemHeap, ASN1UINT propId, void *pProp)
- void **rtMemSetAllocFuncs** (OSMallocFunc malloc_func, OSReallocFunc realloc_func, OSFreeFunc free_func)
- void **rtMemFreeOpenSeqExt** (ASN1CTXT *pctx, Asn1RTDList *pElemList)
- void **rtMemHeapSetFlags** (ASN1CTXT *pctx, ASN1UINT flags)
- void **rtMemHeapClearFlags** (ASN1CTXT *pctx, ASN1UINT flags)

- void **rtMemHeapSetDefBlkSize** (ASN1CTXT *pctxt, ASN1UINT blkSize)
- ASN1UINT **rtMemHeapGetDefBlkSize** (ASN1CTXT *pctxt)

Define Documentation

#define ALLOC_ASN1ARRAY(pctxt, pseqof, type)

```
Value:do {\
if (sizeof(type)*(pseqof)->n < (pseqof)->n) return ASN_E_NOMEM; \
if (((pseqof)->elem = (type*) rtMemHeapAlloc \
(&(pctxt)->pTypeMemHeap, sizeof(type)*(pseqof)->n)) == 0) return ASN_E_NOMEM; \
} while (0)
```

Allocate a dynamic array. This macro allocates a dynamic array of records of the given type. This version of the macro will return the ASN_E_NOMEM error status if the memory request cannot be fulfilled.

Parameters:

pctxt - Pointer to a context block

pseqof - Pointer to a generated SEQUENCE OF array structure. The *n* member variable must be set to the number of records to allocate.

type - Data type of an array record

#define ALLOC_ASN1ARRAY1(pctxt, pseqof, type)

```
Value:do {\
if (sizeof(type)*(pseqof)->n < (pseqof)->n) (pseqof)->elem = 0; \
else (pseqof)->elem = (type*) rtMemHeapAlloc \
(&(pctxt)->pTypeMemHeap, sizeof(type)*(pseqof)->n); \
} while (0)
```

Allocate a dynamic array. This macro allocates a dynamic array of records of the given type. This version of the macro will set the internal parameters of the SEQUENCE OF structure to NULL if the memory request cannot be fulfilled.

Parameters:

pctxt - Pointer to a context block

pseqof - Pointer to a generated SEQUENCE OF array structure. The *n* member variable must be set to the number of records to allocate.

type - Data type of an array record

#define ALLOC_ASN1ARRAY2(pctxt, n, type)

```
Value:((type*) ((sizeof(type)*n < n) ? 0 : \
rtMemHeapAlloc (&(pctxt)->pTypeMemHeap, sizeof(type)*n))
```

Allocate a dynamic array. This macro allocates a dynamic array of records of the given type. This version returns the pointer to the allocated array directly to the caller. It does not use a generated SEQUENCE OF structure.

Parameters:

pctxt - Pointer to a context block

n - Number of records to allocate

type - Data type of an array record

#define ALLOC_ASN1ELEM(pctxt, type) (type*) rtMemHeapAllocZ (&(pctxt)->pTypeMemHeap, sizeof(type))

Allocate and zero an ASN.1 element. This macro allocates and zeros a single element of the given type. (Note: this macro is deprecated. Users should now use `rtMemAllocTypeZ` instead).

Parameters:

pctxt - Pointer to a context block
type - Data type of record to allocate

#define ASN1ARRAYSIZE(x) (sizeof(x)/sizeof(x[0]))

Get array size. This macro returns the number of elements in an array.

Parameters:

x - Array variable

#define ASN1MALLOC(pctxt, nbytes) rtMemHeapAlloc(&(pctxt)->pTypeMemHeap, nbytes)

Allocate memory. This macro allocates the given number of bytes. It is similar to the C `malloc` run-time function. (Note: this macro is deprecated. Users should now use `rtMemAlloc` instead).

Parameters:

pctxt - Pointer to a context block
nbytes - Number of bytes of memory to allocate

Returns:

- Void pointer to allocated memory or NULL if insufficient memory was available to fulfill the request.

#define ASN1MEMFREE(pctxt) rtMemHeapFreeAll(&(pctxt)->pTypeMemHeap)

Free memory associated with a context. This macro frees all memory held within a context. This is all memory allocated using the `ASN1MALLOC` (and similar macros) and the `rtMem` memory allocation functions using the given context variable.

Parameters:

pctxt - Pointer to a context block

#define ASN1MEMFREEPTR(pctxt, pmem) rtMemHeapFreePtr(&(pctxt)->pTypeMemHeap, (void*)pmem)

Free memory pointer. This macro frees memory at the given pointer. The memory must have been allocated using the `ASN1MALLOC` (or similar) macros or the `rtMem` memory allocation functions. This macro is similar to the C `free` function.

Parameters:

pctxt - Pointer to a context block
pmem - Pointer to memory block to free. This must have been allocated using the `ASN1MALLOC` macro or the `rtMemHeapAlloc` function.

#define ASN1MEMRESET(pctxt) rtMemHeapReset(&(pctxt)->pTypeMemHeap)

Reset memory associated with a context. This macro resets all memory held within a context. This is all memory allocated using the `ASN1MALLOC` (and similar macros) and the `rtMem` memory allocation functions using the given context variable.

The difference between this and the `ASN1MEMFREE` macro is that the memory blocks held within the context are not actually freed. Internal pointers are reset so the existing blocks can be reused. This can provide a performance improvement for repetitive tasks such as decoding messages in a loop.

Parameters:

pctxt - Pointer to a context block

#define ASN1REALLOC(pctxt, pmem, nbytes) rtMemHeapRealloc(&(pctxt)->pTypeMemHeap, pmem, nbytes)

Reallocate memory. This macro reallocates a memory block (either expands or contracts) to the given number of bytes. It is similar to the C `realloc` run-time function. (Note: this macro is deprecated. Users should now use `rtMemRealloc` instead).

Parameters:

pctxt - Pointer to a context block
pmem - Pointer to memory block to reallocate. This must have been allocated using the ASN1ALLOC macro or the `rtMemHeapAlloc` function.
nbytes - Number of bytes of memory to which the block is to be resized.

Returns:

- Void pointer to allocated memory or NULL if insufficient memory was available to fulfill the request. This may be the same as the *pmem* pointer that was passed in if the block did not need to be relocated.

#define REALLOC_ASN1ARRAY(pctxt, pseqof, type)

```
Value:do {\nif (sizeof(type)*(pseqof)->n < (pseqof)->n) return ASN E NOMEM; \nif (((pseqof)->elem = (type*) rtMemHeapRealloc \n(&(pctxt)->pTypeMemHeap, (void*)(pseqof)->elem, sizeof(type)*(pseqof)->n)) == 0) \nreturn ASN E NOMEM; \n} while (0)
```

Reallocate an array. This macro reallocates an array (either expands or contracts) to hold the given number of elements. The number of elements is specified in the *n* member variable of the *pseqof* argument.

Parameters:

pctxt - Pointer to a context block
pseqof - Pointer to a generated SEQUENCE OF array structure. The *n* member variable must be set to the number of records to allocate.
type - Data type of an array record

#define rtMemAlloc(pctxt, nbytes) rtMemHeapAlloc(&(pctxt)->pTypeMemHeap,nbytes)

Allocate memory. This macro allocates the given number of bytes. It is similar to the C `malloc` run-time function.

Parameters:

pctxt - Pointer to a context block
nbytes - Number of bytes of memory to allocate

Returns:

- Void pointer to allocated memory or NULL if insufficient memory was available to fulfill the request.

#define rtMemAllocType(pctxt, ctype) (ctype*)rtMemHeapAlloc(&(pctxt)->pTypeMemHeap,sizeof(ctype))

Allocate type. This macro allocates memory to hold a variable of the given type.

Parameters:

pctxt - Pointer to a context block
ctype - Name of C typedef

Returns:

- Pointer to allocated memory or NULL if insufficient memory was available to fulfill the request.

#define rtMemAllocTypeZ(pctxt, ctype) (ctype*)rtMemHeapAllocZ(&(pctxt)->pTypeMemHeap,sizeof(ctype))

Allocate type and zero memory. This macro allocates memory to hold a variable of the given type and initializes the allocated memory to zero.

Parameters:

pctxt - Pointer to a context block
ctype - Name of C typedef

Returns:

- Pointer to allocated memory or NULL if insufficient memory was available to fulfill the request.

#define rtMemAllocZ(pctxt, nbytes) rtMemHeapAllocZ(&(pctxt)->pTypeMemHeap,nbytes)

Allocate and zero memory. This macro allocates the given number of bytes and then initializes the memory block to zero.

Parameters:

pctxt - Pointer to a context block
nbytes - Number of bytes of memory to allocate

Returns:

- Void pointer to allocated memory or NULL if insufficient memory was available to fulfill the request.

#define rtMemFree(pctxt) rtMemHeapFreeAll(&(pctxt)->pTypeMemHeap)

Free memory associated with a context. This macro frees all memory held within a context. This is all memory allocated using the rtMem memory allocation functions and macros using the given context variable.

Parameters:

pctxt - Pointer to a context block

#define rtMemFreePtr(pctxt, mem_p)

```
Value: if (rtMemHeapCheckPtr (&(pctxt)->pTypeMemHeap, (void*)mem_p)) \
rtMemHeapFreePtr (&(pctxt)->pTypeMemHeap, (void*)mem_p)
```

Free memory pointer. This macro frees memory at the given pointer. The memory must have been allocated using the ASN1MALLOC (or similar) macros or the rtMem memory allocation macros. This macro is similar to the C `free` function.

Parameters:

pctxt - Pointer to a context block
mem_p - Pointer to memory block to free. This must have been allocated using the ASN1MALLOC or rtMemAlloc macro or the rtMemHeapAlloc function.

#define rtMemRealloc(pctxt, mem_p, nbytes) rtMemHeapRealloc(&(pctxt)->pTypeMemHeap, (void*)mem_p, nbytes)

Reallocate memory. This macro reallocates a memory block (either expands or contracts) to the given number of bytes. It is similar to the C `realloc` run-time function.

Parameters:

pctxt - Pointer to a context block
mem_p - Pointer to memory block to reallocate. This must have been allocated using the ASN1MALLOC macro or the rtMemHeapAlloc function.
nbytes - Number of bytes of memory to which the block is to be resized.

Returns:

- Void pointer to allocated memory or NULL if insufficient memory was available to fulfill the request. This may be the same as the pmem pointer that was passed in if the block did not need to be relocated.

#define rtMemReset(pctxt) rtMemHeapReset(&(pctxt)->pTypeMemHeap)

Reset memory associated with a context. This macro resets all memory held within a context. This is all memory allocated using the ASN1MALLOC (and similar macros) and the rtMem memory allocation functions using the given context variable.

The difference between this and the rtMemFree macro is that the memory blocks held within the context are not actually freed. Internal pointers are reset so the existing blocks can be reused. This can provide a performance improvement for repetitive tasks such as decoding messages in a loop.

Parameters:

pctxt - Pointer to a context block

Function Documentation**ASN1UINT rtMemHeapGetDefBlkSize (ASN1CTXT * pctxt)**

This function returns the actual granularity of memory blocks.

Parameters:

pctxt Pointer to a context block.

void rtMemHeapSetDefBlkSize (ASN1CTXT * pctxt, ASN1UINT blkSize)

This function sets the pointer to standard allocation functions. These functions are used to allocate/reallocate/free the memory blocks. By default, standard C functions - malloc, realloc, and free - are used. But if some platforms do not support these functions or some other reasons exist) they can be overloaded. The functions being overloaded should have the same prototypes as standard ones.

Parameters:

pctxt Pointer to a context block.

blkSize The currently used minimum size and the granularity of memory blocks.

void rtMemSetAllocFuncs (OSMallocFunc malloc_func, OSReallocFunc realloc_func, OSFreeFunc free_func)

This function sets the pointers to standard allocation functions. These functions are used to allocate/reallocate/free the memory blocks. By default, standard C functions - 'malloc', 'realloc' and 'free' - are used. But if some platforms do not support these functions (or some other reasons exist) they can be overloaded. The functions being overloaded should have the same prototypes as standard ones.

Parameters:

malloc_func Pointer to the memory allocation function ('malloc' by default).

realloc_func Pointer to the memory reallocation function ('realloc' by default).

free_func Pointer to the memory deallocation function ('free' by default).

Context Management Functions

Detailed Description

Context initialization functions handle the allocation, initialization, and destruction of ASN.1 context variables (variables of type ASN1CTXT). These variables hold all of the working data used during the process of encoding or decoding a message. The context provides thread safe operation by isolating what would otherwise be global variables within this structure. The context variable is passed from function to function as a message is encoded or decoded and maintains state information on the encoding or decoding process.

Functions

- `int rtInitContextBuffer (ASN1CTXT *ctxt_p, const OSOCTET *bufaddr, size_t bufsiz)`
- `int rtInitContextUsingKey (ASN1CTXT *ctxt_p, const OSOCTET *key, size_t keylen)`
- `int rtInitContext (ASN1CTXT *pctxt)`
- `int rtInitSubContext (ASN1CTXT *ctxt_p, ASN1CTXT *psrc)`
- `int rtPreInitContext (ASN1CTXT *ctxt_p)`
- `ASN1CTXT * rtNewContext (void)`
- `ASN1CTXT * rtNewContextUsingKey (const OSOCTET *key, size_t keylen)`
- `void rtFreeContext (ASN1CTXT *ctxt_p)`
- `int rtReadNextByte (ASN1CTXT *pCtxt)`
- `void rtSetCopyValues (ASN1CTXT *ctxt_p, OSBOOL value)`
- `void rtSetFastCopy (ASN1CTXT *ctxt_p, OSBOOL value)`

Function Documentation

void rtFreeContext (ASN1CTXT * *ctxt_p*)

This function frees all dynamic memory associated with a context. This includes all memory inside the block (in particular, the list of memory blocks used by the `rtMem` functions) as well as the block itself if allocated with the `rtNewContext` function

Parameters:

ctxt_p A pointer to a context structure.

int rtInitContext (ASN1CTXT * *pctxt*)

This function initializes an ASN1CTXT block. It makes sure that if the block was not previously initialized, that all key working parameters are set to their correct initial state values (i.e. declared within a function as a normal working variable). A user must call this function or `rtNewContext` before making any other run-time library calls.

Parameters:

pctxt Pointer to context structure variable to be initialized.

int rtInitContextBuffer (ASN1CTXT * *ctxt_p*, const OSOCTET * *bufaddr*, size_t *bufsiz*)

This function assigns a buffer to an ASN1CTXT block. The block should have been previously initialized by `rtInitContext`.

Parameters:

ctxt_p The pointer to the context structure variable to be initialized.

bufaddr For encoding, the address of a memory buffer to receive and encode a message. For decoding the address of a buffer that contains the message data to be decoded. This address will be stored within the context structure. For encoding it might be zero, the dynamic buffer will be used in this case.

bufsiz The size of the memory buffer. For encoding, it might be zero; the dynamic buffer will be used in this case. If *bufaddr* is NULL, this parameter specifies the initial size of the dynamic buffer; if 0 - the default size will be used.

Returns:

Completion status of operation:

- 0 (ASN_OK) = success,
- negative return value is error.

int rtInitContextUsingKey (ASN1CTXT * ctxt_p, const OSOCTET * key, size_t keylen)

This function initializes a context using a run-time key. This form is required for evaluation and limited distribution software. The compiler will generate a macro for `rtInitContext` in the `rtkey.h` file that will invoke this function with the generated run-time key.

Parameters:

ctxt_p The pointer to the context structure variable to be initialized.

key Key data generated by ASN1C compiler.

keylen Key data field length.

Returns:

Completion status of operation:

- 0 (ASN_OK) = success,
- negative return value is error.

ASN1CTXT* rtNewContext (void)

This function allocates a new ASN1CTXT block and initializes it. Although the block is allocated from the standard heap, it should not be freed using `free`. The `rtFreeContext` function should be used because this frees items allocated within the block before freeing the block itself.

This is the preferred way of setting up a new encode or decode context because it ensures the block is properly initialized before using it. If a context variable is declared on the stack, the user must first remember to invoke the `rtInitContext` function on it. This can be a source of errors. This function can be called directly when setting up BER context or it will be invoked from within the `pu_newContext` call for PER.

ASN1CTXT* rtNewContextUsingKey (const OSOCTET * key, size_t keylen)

This function allocates a new ASN1CTXT block and initializes it. Although the block is allocated from the standard heap, it should not be freed using `free`. The `rtFreeContext` function should be used because this frees items allocated within the block before freeing the block itself.

This is the preferred way of setting up a new encode or decode context because it ensures the block is properly initialized before using it. If a context variable is declared on the stack, the user must first remember to invoke the `rtInitContext` function on it. This can be a source of errors. This function can be called directly when setting up BER context or it will be invoked from within the `pu_newContext` call for PER.

Parameters:

key Key data generated by ASN1C compiler.
keylen Key data field length.

Returns:

Newly allocated context if successful, NULL if failure.

void rtSetCopyValues (ASN1CTXT * *ctxt_p*, OSBOOL *value*)

This function is deprecated, use **rtSetFastCopy**.

void rtSetFastCopy (ASN1CTXT * *ctxt_p*, OSBOOL *value*)

This function sets the ASN1FASTCOPY flag in the specified context. This flag has effect only if decoding is being performed. If this flag is set then some decoded data (BIT STRINGS, OCTET STRINGs or OPEN TYPEs) will not be copied. Instead, a pointer to the location of the data in the decode buffer will be stored in the generated structure. This may improve decoding performance, but should be used carefully. If the decode buffer is destroyed, all uncopied data will be lost. Use only if you are sure the decode buffer will be available when data in the decoded structure is processed. By default this flag is not set.

Parameters:

ctxt_p Pointer to context block structure.
value Boolean value of the flag to set.

Linked List Utility Functions

Detailed Description

Linked list and stack utility functions are used to maintain linked lists and stacks used within the ASN.1 run-time library functions.

Defines

- #define **rtDListFastInit**(pList)

Functions

- Asn1RTDListNode * **rtDListAppend** (ASN1CTXT **ctxt_p*, Asn1RTDList **pList*, void **pData*)
- void **rtDListInit** (Asn1RTDList **pList*)
- Asn1RTDListNode * **rtDListInsert** (ASN1CTXT **ctxt_p*, Asn1RTDList **pList*, int *index*, void **pData*)
- Asn1RTDListNode * **rtDListInsertBefore** (ASN1CTXT **ctxt_p*, Asn1RTDList **pList*, Asn1RTDListNode **node*, void **pData*)
- Asn1RTDListNode * **rtDListInsertAfter** (ASN1CTXT **ctxt_p*, Asn1RTDList **pList*, Asn1RTDListNode **node*, void **pData*)
- Asn1RTDListNode * **rtDListFindByIndex** (Asn1RTDList **pList*, int *index*)
- Asn1RTDListNode * **rtDListFindByData** (Asn1RTDList **pList*, void **data*)
- int **rtDListFindIndexByData** (Asn1RTDList **pList*, void **data*)
- void **rtDListRemove** (Asn1RTDList **pList*, Asn1RTDListNode **node*)
- void **rtDListFreeNodes** (ASN1CTXT **ctxt_p*, Asn1RTDList **pList*)
- void **rtDListFreeAll** (ASN1CTXT **ctxt_p*, Asn1RTDList **pList*)

- int **rtDListToArray** (ASN1CTXT *ctxt_p, Asn1RTDList *pList, void **ppArray, OSUINT32 *pElements, int elemSize)
- int **rtDListAppendArray** (ASN1CTXT *ctxt_p, Asn1RTDList *pList, const void *pArray, OSUINT32 numElements, int elemSize)
- int **rtDListAppendArrayCopy** (ASN1CTXT *ctxt_p, Asn1RTDList *pList, const void *pArray, OSUINT32 numElements, int elemSize)
- Asn1RTDListNode * **rtDListAppendNode** (ASN1CTXT *ctxt_p, Asn1RTDList *pList, void *pData)

Define Documentation

#define rtDListFastInit(pList)

```
Value:do {\
if ((pList) != 0) { \
(pList)->head = (pList)->tail = (Asn1RTDListNode*) 0; \
(pList)->count = 0; } \
} while (0)
```

Function Documentation

Asn1RTDListNode* rtDListAppend (ASN1CTXT * ctxt_p, Asn1RTDList * pList, void * pData)

This function appends an item to the linked list structure. The data item is passed into the function as a void pointer that can point to any object of any type. The rtMemAlloc function is used to allocated the memory for the list node structure; therefore, all internal list memory will be released whenever rtMemFree is called. The pointer to the data item itself is stored in the node structure - a copy is not made.

Parameters:

ctxt_p A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.

pList A pointer to a linked list structure onto which the data item is to be appended. A pointer to an updated linked list structure.

pData A pointer to a data item to be appended to the list.

Returns:

A pointer to an allocated node structure used to link the given data value into the list.

int rtDListAppendArray (ASN1CTXT * ctxt_p, Asn1RTDList * pList, const void * pArray, OSUINT32 numElements, int elemSize)

This function appends an array items' pointers into a doubly linked list structure.

Parameters:

ctxt_p A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.

pList A pointer to a linked list structure. A pointer to an updated linked list structure. The elements from the array will be appended to the end of the list.

pArray A pointer to the source array to be converted.

numElements The number of elements in the array.

elemSize The size of one element in the array. Use sizeof() operator to pass this parameter.

Returns:

Completion status of operation:

- 0 (ASN_OK) = success,
- negative return value is error.

int rtDListAppendArrayCopy (ASN1CTXT * *ctxt_p*, Asn1RTDList * *pList*, const void * *pArray*, OSUINT32 *numElements*, int *elemSize*)

This function appends a copy of an array's items into a doubly linked list structure. This memory for copies is allocated via calls to `rtMemAlloc`.

Parameters:

ctxt_p A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pList A pointer to a linked list structure. A pointer to an updated linked list structure. The elements from the array will be appended to the end of the list.
pArray A pointer to the source array to be converted.
numElements The number of elements in the array.
elemSize The size of one element in the array. Use `sizeof()` operator to pass this parameter.

Returns:

Completion status of operation:

- 0 (ASN_OK) = success,
- negative return value is error.

Asn1RTDListNode* rtDListFindByData (Asn1RTDList * *pList*, void * *data*)

This function gets a node from a linked list structure that contains a specific data value. The comparison is done on the pointer values, not the data objects themselves.

Parameters:

pList A pointer to a linked list structure onto which the data item is to be appended.
data A pointer to the data item to be found in the list.

Returns:

A pointer to the found node structure. NULL is returned if a matching node could be found.

Asn1RTDListNode* rtDListFindByIndex (Asn1RTDList * *pList*, int *index*)

This function gets a node from a linked list structure at a specified index.

Parameters:

pList A pointer to a linked list structure onto which the data item is to be appended.
index A relative zero-based index of the node in the list to be returned.

Returns:

The pointer to the found node structure. NULL, if the node is not found.

int rtDListFindIndexByData (Asn1RTDList * *pList*, void * *data*)

This function gets a node's index from a linked list structure that contains a specific data value. The comparison is done on the pointer values, not the data objects themselves.

Parameters:

pList A pointer to a linked list structure onto which the data item is to be appended.
data A pointer to the data item to be found in the list.

Returns:

The index of the node that contains specified data. Negative one (-1) is returned if matching node could not be found.

void rtDListFreeAll (ASN1CTXT * *ctxt_p*, Asn1RTDList * *pList*)

This function removes all nodes from the linked list structure and releases the memory that was allocated for storing nodes' structures (`Asn1RTDListNode`) and for data. The memory for

data in each node must have been previously allocated with calls to `rtMemAlloc`, `rtMemAllocZ`, or `rtMemRealloc` functions.

Parameters:

ctxt_p A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.

pList A pointer to a linked list structure onto which the data item is to be appended. A pointer to an updated linked list structure. A pointer to an empty linked list structure.

void rtDListFreeNodes (ASN1CTXT * *ctxt_p*, Asn1RTDList * *pList*)

This function removes all nodes from the linked list and releases the memory that was allocated for storing nodes' structures (`Asn1RTDListNode`). The data will not be released.

Parameters:

ctxt_p A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.

pList A pointer to a linked list structure onto which the data item is to be appended. A pointer to an updated linked list structure.

void rtDListInit (Asn1RTDList * *pList*)

This function initializes a doubly linked list structure. It sets the number of elements to zero and sets all internal pointer values to NULL. A doubly linked-list structure is described by the `Asn1RTDList` type defined in **asn1type.h**. Nodes of the list are of type `Asn1RTDListNode`.

Memory for the structures is allocated using the `rtMemAlloc` run-time function and is maintained within the context structure that is a required parameter to all `rtDList` functions. This memory is released when `rtMemFree` is called or the Context is released. Unless otherwise noted, all data passed into the list functions is simply stored on the list by value (i.e. a deep-copy of the data is not done).

Parameters:

pList A pointer to a linked list structure to be initialized.

Asn1RTDListNode* rtDListInsert (ASN1CTXT * *ctxt_p*, Asn1RTDList * *pList*, int *index*, void * *pData*)

This function inserts an item into the linked list structure. The data item is passed into the function as a void pointer that can point to an object of any type. The `rtMemAlloc` function is used to allocate the memory for the list node structure; therefore, all internal list memory will be released whenever the `rtMemFree` is called.

Parameters:

ctxt_p A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.

pList A pointer to a linked list structure onto which the data item is to be appended. A pointer to an updated linked list structure.

index An index at which the specified item is to be inserted.

pData A pointer to a data item to be inserted to the list.

Returns:

A pointer to an allocated node structure used to link the given data value into the list.

**Asn1RTDListNode* rtDListInsertAfter (ASN1CTXT * *ctxt_p*, Asn1RTDList * *pList*,
Asn1RTDListNode * *node*, void * *pData*)**

This function inserts an item into a linked list structure after a specified node. The data item is passed into the function as a void pointer that can point to any type. The `rtMemAlloc` function is used to allocate the memory for the list structure; therefore, all internal list memory will be released whenever the `rtMemFree` is called.

Parameters:

ctxt_p A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pList A pointer to a linked list structure onto which the data item is to be appended. A pointer to an updated linked list structure.
node A pointer to the node after which the specified item is to be inserted. It should already be in the linked list structure.
pData A pointer to a data item to be inserted to the list.

Returns:

A pointer to an allocated node structure used to link the given data value into the list

**Asn1RTDListNode* rtDListInsertBefore (ASN1CTXT * *ctxt_p*, Asn1RTDList * *pList*,
Asn1RTDListNode * *node*, void * *pData*)**

This function inserts an item into a linked list structure before a specified node. The data item is passed into the function as a void pointer that can point to an object of any type. The `rtMemAlloc` function is used to allocate the memory for the list node structure; therefore, all internal list memory will be released whenever the `rtMemFree` is called.

Parameters:

ctxt_p A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pList A pointer to a linked list structure onto which the data item is to be appended. A pointer to an updated linked list structure.
node A pointer to the node before which the specified item is to be inserted. It should already be in the linked list structure.
pData A pointer to a data item to be inserted to the list.

Returns:

A pointer to an allocated node structure used to link the given data value into the list.

void rtDListRemove (Asn1RTDList * *pList*, Asn1RTDListNode * *node*)

This function removes a node from the linked list structure. The `rtMemAlloc` function was used to allocate the memory for the list node structure, therefore, all internal list memory will be released whenever `rtMemFree` or `rtMemFreePtr` is called.

Parameters:

pList A pointer to a linked list structure onto which the data item is to be appended. A pointer to an updated linked list structure.
node A pointer to the node that is to be removed. It should already be in the linked list structure.

**int rtDListToArray (ASN1CTXT * *ctxt_p*, Asn1RTDList * *pList*, void ** *ppArray*, OSUINT32 *
pElements, int *elemSize*)**

This function converts a doubly linked list to an array.

Parameters:

ctxt_p A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.

pList A pointer to a linked list structure. A pointer to an updated linked list structure.
ppArray A pointer to a pointer to a destination array. A pointer to filled destination array.
pElements A pointer to the number of elements already allocated in *ppArray*. If value of *pElements* is less than the number of nodes in the list, then a new array is allocated and pointer is stored into *ppArray*. Memory is allocated via calls to the *rtMemAlloc* function.
elemSize The size of one element in the array. Use *sizeof()* operator to pass this parameter.

Returns:

The number of elements in the returned array.

Diagnostic Trace Functions

Detailed Description

Diagnostic trace functions allow the output of trace messages to *stdout* that trace the execution of compiler generated functions. The primary function is *rtdiag*, a *printf*-like function that checks a global trace flag before writing to the standard output.

Functions

- void **rtdiag** (const char **fmtsSpec*,...)
- void **rtDiagStream** (ASN1CTXT **pctx*, const char **fmtsSpec*,...)
- void **rtDiagStreamHexDump** (ASN1CTXT **pctx*, const OSOCTET **data*, OSUINT32 *numocts*)
- void **rtDiagHexDump** (const OSOCTET **data*, OSUINT32 *numocts*)
- void **rtSetDiag** (int *value*)
- void **rtHexDump** (const OSOCTET **data*, OSUINT32 *numocts*)
- void **rtHexDumpEx** (const OSOCTET **data*, OSUINT32 *numocts*, int *bytesPerUnit*)
- void **rtHexDumpToFile** (FILE **fp*, const OSOCTET **data*, OSUINT32 *numocts*)
- void **rtHexDumpToFileEx** (FILE **fp*, const OSOCTET **data*, OSUINT32 *numocts*, int *bytesPerUnit*)
- int **rtHexDumpToString** (const OSOCTET **data*, OSUINT32 *numocts*, char **buffer*, int *bufferIndex*, int *bufferSize*)
- int **rtHexDumpToStringEx** (const OSOCTET **data*, OSUINT32 *numocts*, char **buffer*, int *bufferIndex*, int *bufferSize*, int *bytesPerUnit*)
- int **rtHexDumpToStream** (ASN1CTXT **pctx*, const OSOCTET **data*, OSUINT32 *numocts*)
- int **rtHexDumpToStreamEx** (ASN1CTXT **pctx*, const OSOCTET **data*, OSUINT32 *numocts*, int *bytesPerUnit*)

Function Documentation

void rtdiag (const char * *fmtsSpec*, ...)

This function conditionally outputs diagnostic trace messages to *stdout*. The ASN1C compiler embeds calls to this function into the generated source code when the *-trace* option is specified on the command line (note: it may embed the macro version of these calls - *RTDIAGx* - so that these calls can be compiled out of the final code).

Parameters:

fmtsSpec A *printf*-like format specification string describing the message to be printed (for example, "string s, ivalue d").

... A variable list of arguments.

void rtDiagStream (ASN1CTXT * *pctxt*, const char * *fmts*pec, ...)

This function conditionally outputs diagnostic trace messages to output stream for the context. The ASN1C compiler embeds calls to this function into the generated source code when the -trace option is specified on the command line (note: it may embed the macro version of these calls - RTDIAGSTREAMx - so that these calls can be compiled out of the final code).

Parameters:

pctxt Pointer to ASN1CTXT pointing to the output stream.
*fmts*pec A printf-like format specification string describing the message to be printed (for example, "string s, ivalue d").
... A variable list of arguments.

void rtHexDump (const OSOCTET * *data*, OSUIN32 *numocts*)

The rtHexDump function outputs a hexadecimal dump of the current buffer contents to stdout. There is also a macro associated with rtHexDump. This macro is RTHEXDUMP and is defined in the **asn1type.h** header file. It allows a dump function to be compiled in or out of the object code generated by the C/C++ compiler by using a preprocessor variable (_TRACE).

Parameters:

data The pointer to a buffer to be displayed.
numocts The number of octets to be displayed.

void rtHexDumpEx (const OSOCTET * *data*, OSUIN32 *numocts*, int *bytesPerUnit*)

This function outputs a hexadecimal dump of the current buffer contents to stdout, but it may display the dump as an array or bytes, words, or double words.

Parameters:

data The pointer to a buffer to be displayed.
numocts The number of octets to be displayed.
bytesPerUnit The number of bytes in one unit. May be 1 (byte), 2 (word), or 4 (double word).

void rtHexDumpToFile (FILE * *fp*, const OSOCTET * *data*, OSUIN32 *numocts*)

This function outputs a hexadecimal dump of the current buffer contents to a file.

Parameters:

fp A pointer to FILE structure. The file should be opened for writing.
data The pointer to a buffer to be displayed.
numocts The number of octets to be displayed

void rtHexDumpToFileEx (FILE * *fp*, const OSOCTET * *data*, OSUIN32 *numocts*, int *bytesPerUnit*)

This function outputs a hexadecimal dump of the current buffer to a file, but it may output the dump as an array of bytes, words, or double words.

Parameters:

fp A pointer to FILE structure. The file should be opened for writing.
data The pointer to a buffer to be displayed.
numocts The number of octets to be displayed.

bytesPerUnit The number of bytes in one unit. May be 1 (byte), 2 (word), or 4 (double word).

int rtHexDumpToStream (ASN1CTXT * *pctxt*, const OSOCTET * *data*, OSUINT32 *numocts*)

This function outputs a hexadecimal dump of the current buffer contents to a stream.

Parameters:

pctxt A pointer to ASN1CTXT structure which owns the print stream
data The pointer to a buffer to be displayed.
numocts The number of octets to be displayed

Returns:

0 on success, negative on error

int rtHexDumpToStreamEx (ASN1CTXT * *pctxt*, const OSOCTET * *data*, OSUINT32 *numocts*, int *bytesPerUnit*)

This function outputs a hexadecimal dump of the current buffer to a print stream, but it may output the dump as an array of bytes, words, or double words.

Parameters:

pctxt A pointer to ASN1CTXT structure which owns the print stream.
data The pointer to a buffer to be displayed.
numocts The number of octets to be displayed.
bytesPerUnit The number of bytes in one unit. May be 1 (byte), 2 (word), or 4 (double word).

Returns:

0 on success, -ve on error.

int rtHexDumpToString (const OSOCTET * *data*, OSUINT32 *numocts*, char * *buffer*, int *bufferIndex*, int *bufferSize*)

This function formats a hexadecimal dump of the current buffer contents to a string.

Parameters:

data The pointer to a buffer to be displayed.
numocts The number of octets to be displayed.
buffer The destination string buffer.
bufferIndex The starting position in the destination buffer. The formatting of the dump will begin at this position.
bufferSize The total size of the destination buffer.

Returns:

The length of the final string.

int rtHexDumpToStringEx (const OSOCTET * *data*, OSUINT32 *numocts*, char * *buffer*, int *bufferIndex*, int *bufferSize*, int *bytesPerUnit*)

This function formats a hexadecimal dump of the current buffer contents to a string, but it may output the dump as an array of bytes, words, or double words.

Parameters:

data The pointer to a buffer to be displayed.
numocts The number of octets to be displayed.
buffer The destination string buffer.
bufferIndex The starting position in the destination buffer. The formatting of the dump will begin at this position.
bufferSize The total size of the destination buffer.
bytesPerUnit The number of bytes in one unit. May be 1 (byte), 2 (word), or 4 (double word).

Returns:

The length of the final string.

void rtSetDiag (int value)

This function turns diagnostic tracing on or off.

Parameters:

value - Value indicating whether to enable or disable tracing.

Error Formatting and Print Functions

Detailed Description

Error formatting and print functions allow information about the encode/decode errors to be added to a context block structure and then printed out when the error is propagated to the top level.

Functions

- int **rtErrAddIntParm** (ASN1ErrInfo *pErrInfo, int errParm)
 - int **rtErrAddStrParm** (ASN1ErrInfo *pErrInfo, const char *errprm_p)
 - int **rtErrAddTagParm** (ASN1ErrInfo *pErrInfo, ASN1TAG errParm)
 - int **rtErrAddUIntParm** (ASN1ErrInfo *pErrInfo, unsigned int errParm)
 - int **rtErrCopyData** (ASN1ErrInfo *pSrcErrInfo, ASN1ErrInfo *pDestErrInfo)
 - void **rtErrFreeParms** (ASN1ErrInfo *pErrInfo)
 - char * **rtErrFmtMsg** (ASN1ErrInfo *pErrInfo, char *bufp)
 - char * **rtErrGetText** (ASN1CTXT *ctxt_p)
 - void **rtErrLogUsingCB** (ASN1ErrInfo *pErrInfo, ASN1DumpCbFunc cb, void *cbArg)
 - void **rtErrPrint** (ASN1ErrInfo *pErrInfo)
 - int **rtErrReset** (ASN1ErrInfo *pErrInfo)
 - int **rtErrSetData** (ASN1ErrInfo *pErrInfo, int status, const char *module, int lno)
-

Function Documentation**int rtErrAddIntParm (ASN1ErrInfo * pErrInfo, int errParm)**

This function adds an integer parameter to an error information structure. Parameter substitution is done in much the same way as it is done in C printf statments. The base error message specification that goes along with a particular status code may have variable fields built in using " modifiers. These would be replaced with actual parameter data.

Parameters:

pErrInfo A pointer to a structure containing information on the error to be printed. Typically, the error info structure referred to is the one inside the ASN1CTXT structure. (i.e. &ctxt_p->errInfo).
errParm The typed error parameter.

Returns:

The status of the operation.

int rtErrAddStrParm (ASN1ErrInfo * pErrInfo, const char * errprm_p)

This function adds an string parameter to an error information structure. Parameter substitution is done in much the same way as it is done in C printf statments. The base error message specification that goes along with a particular status code may have variable fields built in using " modifiers. These would be replaced with actual parameter data.

Parameters:

pErrInfo A pointer to a structure containing information on the error to be printed. Typically, the error info structure referred to is the one inside the ASN1CTXT structure. (i.e. &ctxt_p->errInfo).
errprm_p The typed error parameter.

Returns:

The status of the operation.

int rtErrAddTagParm (ASN1ErrInfo * pErrInfo, ASN1TAG errParm)

This function adds an tag parameter to an error information structure. Parameter substitution is done in much the same way as it is done in C printf statments. The base error message specification that goes along with a particular status code may have variable fields built in using " modifiers. These would be replaced with actual parameter data.

Parameters:

pErrInfo A pointer to a structure containing information on the error to be printed. Typically, the error info structure referred to is the one inside the ASN1CTXT structure. (i.e. &ctxt_p->errInfo).
errParm The typed error parameter.

Returns:

The status of the operation.

int rtErrAddUIntParm (ASN1ErrInfo * pErrInfo, unsigned int errParm)

This function adds an unsigned integer parameter to an error information structure. Parameter substitution is done in much the same way as it is done in C printf statments. The base error message specification that goes along with a particular status code may have variable fields built in using " modifiers. These would be replaced with actual parameter data.

Parameters:

pErrInfo A pointer to a structure containing information on the error to be printed. Typically, the error info structure referred to is the one inside the ASN1CTXT structure. (i.e. &ctxt_p->errInfo).
errParm The typed error parameter.

Returns:

The status of the operation.

void rtErrFreeParms (ASN1ErrInfo * pErrInfo)

This function frees memory associated with the storage of parameters associated with an error message. These parameters are maintained on an internal linked list maintained within the error information structure. The list memory must be freed when error processing is complete. This function is called from within rtErrPrint after teh error has been printed out. It is also called in teh pu_freeContext function.

Parameters:

pErrInfo A pointer to a structure containing information on the error to be printed. Typically, the error info structure referred to is the one inside the ASN1CTXT structure. (i.e. &ctxt_p->errInfo).

char* rtErrGetText (ASN1CTXT * ctxt_p)

This function gets the text of the error

Parameters:

ctxt_p A pointer to a context structure.

void rtErrLogUsingCB (ASN1ErrInfo * pErrInfo, ASN1DumpCbFunc cb, void * cbArg)

This function logs error information using a callback function provided by the user. In many situations, it is not sufficient to write error information to stdout to debug problems. Examples are back-end server applications that run in the background and write diagnostic information to system log files and front-end applications that log error information to widow displays. This function allows different error output methods to be accommodated.

This function is invoked with a line of text from the formatted error output provided in the *txt_p* argument. The *cbArg_p* argument is used to pass in a user-defined parameter (specified in the *cbArg* argument of this function). The integer return status is not used at this time. The callback function is invoked once for each formatted line of information in the error holding structure.

Parameters:

pErrInfo A pointer to a structure containing information on the error to be printed. Typically, the error info structure referred to is the one inside the ASN1CTXT structure (i.e. *&ctxt_p->errInfo*).

cb A callback function to be invoked for each line of formatted error output in the error information structure.

cbArg User defined callback argument to be passed as a parameter to the callback function.

void rtErrPrint (ASN1ErrInfo * pErrInfo)

This function prints error information to the standard output device. The error information is stored in a structure of type ASN1ErrInfo. A structure of the this type is part of the ASN1CTXT structure. This is where error information is stored within the ASN1C generated and low-level encode/decode functions.

Parameters:

pErrInfo A pointer to a structure containing information on the error to be printed. Typically, the error info structure referred to is the one inside the ASN1CTXT structure. (i.e. *&ctxt_p->errInfo*).

int rtErrReset (ASN1ErrInfo * pErrInfo)

This function resets the error information in the error information structure.

Parameters:

pErrInfo A pointer to a structure containing information on the error to be printed. Typically, the error info structure referred to is the one inside the ASN1CTXT structure. (i.e. *&ctxt_p->errInfo*).

int rtErrSetData (ASN1ErrInfo * pErrInfo, int status, const char * module, int lno)

This function sets error information in an error information structure. The information set includes status code, module name, and line number. Location information (i.e. module name and line number) is pushed onto a stack within the error information structure to provide a complete stack trace when the information is printed out.

Parameters:

pErrInfo A pointer to a structure containing information on the error to be printed. Typically, the error info structure referred to is the one inside the ASN1CTXT structure. (i.e. *&ctxt_p->errInfo*).

status The error status code. This is one of the negative error status codes.

module The name of the module (C or C++ source file) in which the module occurred. This is typically obtained by using the `_FILE_` macro.

lno The line number at which the error occurred. This is typically obtained by using the `_LINE_` macro.

Returns:

The status value passed to the operation in the third argument. This makes it possible to set the error information and return the status value in one line of code.

Memory Buffer Management Functions

Detailed Description

Memory buffer management functions handle the allocation, expansion, and deallocation of dynamic memory buffers used by some encode/decode functions. Dynamic memory buffers are buffers that can grow or shrink to hold variable sized amounts of data. This group of functions allows data to be appended to buffers, to be set within buffers, and to be retrieved from buffers. Currently, these functions are used within the generated XER SAX decode routines to collect data as it is parsed by an XML parser.

Functions

- void **rtMemBufInit** (ASN1CTXT *pCtxt, ASN1MemBuf *pMemBuf, OSUINT32 segsize)
- void **rtMemBufInitBuffer** (ASN1CTXT *pCtxt, ASN1MemBuf *pMemBuf, OSOCTET *buf, OSUINT32 bufsize, OSUINT32 segsize)
- void **rtMemBufFree** (ASN1MemBuf *pMemBuf)
- int **rtMemBufAppend** (ASN1MemBuf *pMemBuf, OSOCTET *pdata, OSUINT32 nbytes)
- int **rtMemBufCut** (ASN1MemBuf *pMemBuf, OSUINT32 fromOffset, OSUINT32 nbytes)
- void **rtMemBufReset** (ASN1MemBuf *pMemBuf)
- int **rtMemBufTrimW** (ASN1MemBuf *pMemBuf)
- int **rtMemBufSet** (ASN1MemBuf *pMemBuf, OSOCTET value, OSUINT32 nbytes)
- OSOCTET * **rtMemBufGetData** (ASN1MemBuf *pMemBuf, int *length)
- int **rtMemBufGetDataLen** (ASN1MemBuf *pMemBuf)
- int **rtMemBufPreAllocate** (ASN1MemBuf *pMemBuf, OSUINT32 nbytes)
- OSBOOL **rtMemBufSetExpandable** (ASN1MemBuf *pMemBuf, OSBOOL isExpandable)

Function Documentation

int rtMemBufAppend (ASN1MemBuf * pMemBuf, OSOCTET * pdata, OSUINT32 nbytes)

This function appends the data to the end of a memory buffer. If the buffer was dynamic and full then the buffer will be reallocated. If it is static (the static buffer was assigned by a call to `rtMemBufInitBuffer`) or it is empty (no memory previously allocated) then a new buffer will be allocated.

Parameters:

pMemBuf A pointer to a memory buffer structure.

pdata The pointer to the buffer to be appended. The data will be copied at the end of the memory buffer.

nbytes The number of bytes to be copied from `pData`.

Returns:

Completion status of operation:

- 0 (ASN_OK) = success,
- negative return value is error.

int rtMemBufCut (ASN1MemBuf * *pMemBuf*, OSUINT32 *fromOffset*, OSUINT32 *nbytes*)

This function cuts off the part of memory buffer. The beginning of the cutting area is specified by offset "fromOffset" and the length is specified by "nbytes". All data in this part will be lost. The data from the offset "fromOffset + nbytes" will be moved to "fromOffset" offset.

Parameters:

pMemBuf A pointer to a memory buffer structure.

fromOffset The offset of the beginning part, being cut off.

nbytes The number of bytes to be cut off from the memory buffer.

Returns:

Completion status of operation:

- 0 (ASN_OK) = success,
- negative return value is error.

void rtMemBufFree (ASN1MemBuf * *pMemBuf*)

This function frees the memory buffer. If memory was allocated then it will be freed. Do not use the memory buffer structure after this function is called.

Parameters:

pMemBuf A pointer to a memory buffer structure.

OSOCKET* rtMemBufGetData (ASN1MemBuf * *pMemBuf*, int * *length*)

This function returns the pointer to the used part of a memory buffer.

Parameters:

pMemBuf A pointer to a memory buffer structure.

length The pointer to the length of the used part of the memory buffer.

Returns:

The pointer to the used part of the memory buffer.

int rtMemBufGetDataLen (ASN1MemBuf * *pMemBuf*)

This function returns the length of the used part of a memory buffer.

Parameters:

pMemBuf A pointer to a memory buffer structure.

Returns:

The length of the used part of the buffer.

void rtMemBufInit (ASN1Ctxt * *pCtxt*, ASN1MemBuf * *pMemBuf*, OSUINT32 *segsz*)

This function initializes a memory buffer structure. It does not allocate memory; it sets the fields of the structure to the proper states. This function must be called before any operations with the memory buffer.

Parameters:

pCtxt A provides a storage area for the function to store all working variables that must be maintained between function calls.

pMemBuf A pointer to the initialized memory buffer structure.
segsz The number of bytes in which the memory buffer will be expanded in case it is full.

void rtMemBufInitBuffer (ASN1CTXT * *pCtxt*, ASN1MemBuf * *pMemBuf*, OSOCTET * *buf*, OSUINT32 *bufsize*, OSUINT32 *segsz*)

This function assigns a static buffer to the memory buffer structure. It does not allocate memory; it sets the pointer to the passed buffer. If additional memory is required (for example, additional data is appended to the buffer using `rtMemBufAppend`), a dynamic buffer will be allocated and all data copied to the new buffer.

Parameters:

pCtxt A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pMemBuf A pointer to a memory buffer structure.
buf A pointer to the buffer to be assigned.
bufsize The size of the buffer.
segsz The number of bytes on which the memory buffer will be expanded in case it is full.

int rtMemBufPreAllocate (ASN1MemBuf * *pMemBuf*, OSUINT32 *nbytes*)

This function allocates a buffer with a predetermined amount of space.

Parameters:

pMemBuf A pointer to a memory buffer structure.
nbytes The number of bytes to be copied from `pData`.

Returns:

Completion status of operation:

- 0 (ASN_OK) = success,
- negative return value is error.

void rtMemBufReset (ASN1MemBuf * *pMemBuf*)

This function resets the memory buffer structure. It does not free memory, just sets the pointer to the beginning and the used length to zero.

Parameters:

pMemBuf A pointer to a memory buffer structure.

int rtMemBufSet (ASN1MemBuf * *pMemBuf*, OSOCTET *value*, OSUINT32 *nbytes*)

This function sets part of a memory buffer to a specified octet value. The filling is started from the end of the memory buffer. If the buffer is dynamic and full, then the buffer will be reallocated. If it is static (a static buffer was assigned by a call to `rtMemBufInitBuffer`) or it is empty (no memory previously was allocated) then a new buffer will be allocated.

Parameters:

pMemBuf A pointer to a memory buffer structure.
value The pointer to the buffer to be appended. The data will be copied at the end of the memory buffer.
nbytes The number of bytes to be copied from `pData`.

Returns:

Completion status of operation:

- 0 (ASN_OK) = success,
- negative return value is error.

OSBOOL rtMemBufSetExpandable (ASN1MemBuf * pMemBuf, OSBOOL isExpandable)

This function sets "isExpandable" flag for the memory buffer object. By default, this flag is set to TRUE, thus, memory buffer could be expanded, even if it was initialized by static buffer (see `rtMemBufInitBuffer`). If flag is cleared and buffer is full the `rtMemBufAppend`/`rtMemBufPreAllocate` functions will return error status.

Parameters:

pMemBuf A pointer to a memory buffer structure.
isExpandable TRUE, if buffer should be expandable.

Returns:

Previous state of "isExpandable" flag.

int rtMemBufTrimW (ASN1MemBuf * pMemBuf)

This function trims white space of the memory buffer.

Parameters:

pMemBuf A pointer to a memory buffer structure.

Object Identifier Helper Functions

Detailed Description

Object identifier helper functions provide assistance in working with the object identifier ASN.1 type. Three functions are provided: two to populate an Object Identifier structure and one to print the contents.

Functions

- void **rtSetOID** (ASN1OBJID *ptarget, ASN1OBJID *psource)
 - void **rtAddOID** (ASN1OBJID *ptarget, ASN1OBJID *psource)
-

Function Documentation

void rtAddOID (ASN1OBJID * ptarget, ASN1OBJID * psource)

This function appends one object identifier to another one. It copies the data from a source variable to the end of a target variable. Typically, the source variable is a compiler-generated object identifier constant that resulted from an object identifier value specification within an ASN.1 specification.

Parameters:

ptarget A pointer to a target object identifier variable to receive object identifier data. Typically, this is a variable within a compiler-generated C structure.
psource A pointer to a source object identifier variable to copy to a target. Typically, this is a compiler-generated variable corresponding to an ASN.1 value specification in the ASN.1 source file.

void rtSetOID (ASN1OBJID * *ptarget*, ASN1OBJID * *psource*)

This function populates an object identifier variable with data. It copies data from a source variable to a target variable. Typically, the source variable is a compiler-generated object identifier constant that resulted from an object identifier value specification within an ASN.1 specification.

Parameters:

ptarget A pointer to a target object identifier variable to receive object * identifier data. Typically, this is a variable within a compiler-generated C structure.

psource A pointer to a source object identifier variable to copy to a target. Typically, this is a compiler-generated variable corresponding to an ASN.1 value specification in the ASN.1 source file.

Linked List and Stack Utility Functions

Detailed Description

Linked list and stack utility functions are used to maintain linked lists and stacks used within the ASN.1 run-time library functions.

Functions

- void **rtSListInit** (Asn1RTSList *pList)
- void **rtSListInitEx** (ASN1CTXT *pctxt, Asn1RTSList *pList)
- void **rtSListFree** (Asn1RTSList *pList)
- Asn1RTSList * **rtSListCreate** (void)
- Asn1RTSList * **rtSListCreateEx** (ASN1CTXT *pctxt)
- Asn1RTSListNode * **rtSListAppend** (Asn1RTSList *pList, void *pData)
- OSBOOL **rtSListFind** (Asn1RTSList *pList, void *pData)
- void **rtSListRemove** (Asn1RTSList *pList, void *pData)
- Asn1RTStack * **rtStackCreate** (void)
- Asn1RTStack * **rtStackCreateEx** (ASN1CTXT *pctxt)
- void **rtStackInit** (Asn1RTStack *pStack)
- void * **rtStackPop** (Asn1RTStack *pStack)
- int **rtStackPush** (Asn1RTStack *pStack, void *pData)

Function Documentation

Asn1RTSListNode* rtSListAppend (Asn1RTSList * *pList*, void * *pData*)

This function appends an item to a linked list structure. The data item is passed into the function as a void parameter that can point to an object of any type.

Parameters:

pList A pointer to a linked list onto which the data item is to be appended.

pData A pointer to a data item to be appended to the list.

Returns:

A pointer to the allocated linked list structure.

Asn1RTSList* rtSListCreate (void)

This function creates a new linked list structure. It allocates memory for the structure and calls `rtSListInit` to initialize the structure.

Returns:

A pointer to the allocated linked list structure.

Asn1RTSList* rtSListCreateEx (ASN1CTX * *pctx*)

The `rtSListAppend` function appends an item to linked list structure. The data is passed into the function as a void that can point to an object of any type.

Parameters:

pctx A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.

Returns:

A pointer to the allocated linked list structure.

OSBOOL rtSListFind (Asn1RTSList * *pList*, void * *pData*)

This function finds an item in the linked list structure. The data item is passed into the function as a void pointer that can point to an object of any type. If the appropriate node is found in the list this function returns TRUE, otherwise FALSE.

Parameters:

pList A pointer to a linked list onto which the data item is to be appended.

pData A pointer to a data item to be appended to the list.

Returns:

TRUE, if the node is found, otherwise FALSE.

void rtSListFree (Asn1RTSList * *pList*)

This function removes all nodes from the linked list structure and releases memory that was allocated for storing nodes' structures (`Asn1RTSListNode`). The data will not be freed.

Parameters:

pList A pointer to a linked list onto which the data item is to be appended.

void rtSListInit (Asn1RTSList * *pList*)

Singly linked list structures have only a single link pointer and can therefore only be traversed in a single direction (forward). The node structures consume less memory than those of a doubly linked list.

Another difference between the singly linked list implementation and doubly linked lists is that the singly linked list uses conventional memory allocation functions (C `malloc` and `free`) for less storage instead of the `rtMem` functions. Therefore, it is not a requirement to have an initialized context structure to work with these lists. However, performance may suffer if the lists become large due to the use of non-optimized memory management.

This function initializes a singly linked list structure. It sets the number of elements to zero and sets all internal pointer values to NULL.

Parameters:

pList A pointer to a linked list structure to be initialized.

void rtSListInitEx (ASN1CTXT * *pctxt*, Asn1RTSList * *pList*)

This function is similar to `rtSListInit` but it also sets the `pctxt` (pointer to a context structure member of `Asn1RTSList` structure. This context will be used for further memory allocations; otherwise the standard `malloc` is used.

Parameters:

pctxt A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.

pList A pointer to a linked list structure to be initialized.

void rtSListRemove (Asn1RTSList * *pList*, void * *pData*)

This function finds an item in the linked list structure and removes it from the list. The data item is passed into the function as a void pointer that can point to an object of any type.

Parameters:

pList A pointer to a linked list onto which the data item is to be appended.

pData A pointer to a data item to be appended to the list.

Asn1RTStack* rtStackCreate (void)

This function creates a new stack structure. It allocates memory for the structure and calls `rtStackInit` to initialize the structure.

Returns:

A pointer to an allocated stack structure.

Asn1RTStack* rtStackCreateEx (ASN1CTXT * *pctxt*)

This function creates a new stack structure. The `pctxt` will be used for memory management.

Parameters:

pctxt A pointer to a context structure.

Returns:

A pointer to an allocated stack structure.

void rtStackInit (Asn1RTStack * *pStack*)

This function initializes a stack structure. It sets the number of elements to zero and sets all internal pointer values to `NULL`.

Parameters:

pStack A pointer to a stack structure to be initialized.

void* rtStackPop (Asn1RTStack * *pStack*)

This function pops an item off the stack.

Parameters:

pStack The pointer to the stack structure from which the value is to be popped. Pointer to the updated stack structure.

Returns:

The pointer to the item popped from the stack

int rtStackPush (Asn1RTStack * *pStack*, void * *pData*)

This function pushes an item onto the stack.

Parameters:

pStack A pointer to the structure onto which the data item is to be pushed. The pointer updated to the stack structure

pData A pointer to the data item to be pushed on the stack.

Returns:

Completion status of operation:

- 0 (ASN_OK) = success,
- negative return value is error.

REAL Helper Functions

Detailed Description

REAL helper functions provide assistance in working with the REAL ASN.1 type. Two functions are provided to obtain the plus and minus infinity special values.

Functions

- OSREAL **rtGetPlusInfinity** (void)
 - OSREAL **rtGetMinusInfinity** (void)
 - OSREAL **rtGetMinusZero** (void)
 - OSREAL **rtGetNaN** (void)
 - OSBOOL **rtIsMinusInfinity** (OSREAL value)
 - OSBOOL **rtIsMinusZero** (OSREAL value)
 - OSBOOL **rtIsNaN** (OSREAL value)
 - OSBOOL **rtIsPlusInfinity** (OSREAL value)
-

Function Documentation**OSREAL rtGetMinusInfinity (void)**

This function returns the MINUS-INFINITY special value. The return value may be used for comparing with other OSREAL values or for encoding the REAL MINUS_INFINITY value.

Returns:

MINUS-INFINITY special value.

OSREAL rtGetPlusInfinity (void)

This function returns the PLUS-INFINITY special value. The return value may be used for comparing with other OSREAL values or for encoding the REAL PLUS_INFINITY value.

Returns:

PLUS-INFINITY special value.

Formatted Printing Functions

Detailed Description

This group of functions allows raw ASN.1 data fields to be formatted and printed to stdout and other output devices.

Functions

- `const char * rtBitStrToString` (OSUINT32 numbits, const OSOCTET *data, char *buffer, size_t bufsiz)
 - `const char * rtBoolToString` (OSBOOL value)
 - `const char * rtIntToString` (OSINT32 value, char *buffer, size_t bufsiz)
 - `const char * rtInt64ToString` (OSINT64 value, char *buffer, size_t bufsiz)
 - `const char * rtUIntToString` (OSUINT32 value, char *buffer, size_t bufsiz)
 - `const char * rtUInt64ToString` (OSUINT64 value, char *buffer, size_t bufsiz)
 - `const char * rtOIDToString` (OSUINT32 numids, OSUINT32 *data, char *buffer, size_t bufsiz)
 - `const char * rtOID64ToString` (OSUINT32 numids, OSUINT64 *data, char *buffer, size_t bufsiz)
 - `const char * rtOctStrToString` (OSUINT32 numocts, const OSOCTET *data, char *buffer, size_t bufsiz)
 - `const char * rtTagToString` (ASN1TAG tag, char *buffer, size_t bufsiz)
-

Function Documentation

const char* rtBitStrToString (OSUINT32 numbits, const OSOCTET * data, char * buffer, size_t bufsiz)

This function converts an ASN.1 bit string to a string. The output is an ASN.1 value notation for a binary string (for example, '10010'B).

Parameters:

numbits The number of bits in the data argument to format.

data The buffer containing the bit string to be formatted (note : in the case of BER/DER, this refers to the actual point in the string where the data starts, not where the contents field starts. The contents field contains an extra byte at the beginning that specifies the number of unused bits in the last byte).

buffer A pointer to a buffer to receive the stringified value.

bufsiz The size of the buffer to receive the stringified value.

const char* rtBoolToString (OSBOOL value)

This function converts an ASN.1 Boolean value to a string. The string value is one of the keywords TRUE or FALSE.

Parameters:

value The value to convert.

Returns:

The converted value. This will be a string literal set to either "TRUE" or "FALSE".

const char* rtInt64ToString (OSINT64 value, char * buffer, size_t bufsiz)

This function converts a 64-bit ASN.1 integer value to a string.

Parameters:

value The value to convert.

buffer The pointer to a buffer to receive the stringified value.

bufsiz The size of the buffer to receive the stringified value.

Returns:

The converted value. This pointer will be equal to the buffer argument that was passed in.

const char* rtIntToString (OSINT32 *value*, char * *buffer*, size_t *bufsiz*)

This function converts an ASN.1 integer value to a string.

Parameters:

value The value to convert.

bufsiz The size of the buffer to receive the stringified value.

buffer The pointer to a buffer to receive the stringified value.

Returns:

The converted value. This pointer will be equal to the buffer argument that was passed in.

const char* rtOctStrToString (OSUINT32 *numocts*, const OSOCTET * *data*, char * *buffer*, size_t *bufsiz*)

This function converts an ASN.1 octet string value to a string. The output format is ASN.1 value notation for a hexadecimal string (for example, '1F8A'H).

Parameters:

numocts The number of octets (bytes) in the data argument to format.

data The buffer containing the octet string to be formatted.

bufsiz The size of the buffer to receive the stringified value.

buffer The pointer to a buffer to receive the stringified value.

Returns:

The converted value. This pointer will be equal to the buffer argument that was passed in.

const char* rtOID64ToString (OSUINT32 *numids*, OSUINT64 * *data*, char * *buffer*, size_t *bufsiz*)

This function converts a 64-bit ASN.1 OBJECT IDENTIFIER or RELATED-OID value to a string. The output format is the ASN.1 value notation for an object identifier (ex. { 0 1 222 333 }). All subidentifiers are shown as unsigned integer numbers - no attempt is made to map the identifiers to symbolic names.

Parameters:

numids The number of subidentifiers in the OID value.

data The buffer containing the 64-bit OID subidentifiers to be formatted.

bufsiz The size of the buffer to receive the stringified value.

buffer The pointer to a buffer to receive the stringified value.

Returns:

The converted value. This pointer will be equal to the buffer argument that was passed in.

const char* rtOIDToString (OSUINT32 *numids*, OSUINT32 * *data*, char * *buffer*, size_t *bufsiz*)

This function converts an ASN.1 OBJECT IDENTIFIER or RELATED-OID value to a string. The output format is the ASN.1 value notation for an object identifier (ex. { 0 1 222 333 }). All subidentifiers are shown as unsigned integer numbers - no attempt is made to map the identifiers to symbolic names.

Parameters:

numids The number of subidentifiers in the OID value.

data The buffer containing the OID subidentifiers to be formatted.

bufsiz The size of the buffer to receive the stringified value.

buffer The pointer to a buffer to receive the stringified value.

Returns:

The converted value. This pointer will be equal to the buffer argument that was passed in.

const char* rtTagToString (ASN1TAG tag, char * buffer, size_t bufsiz)

This function converts an ASN.1 tag to a string. The tag is represented using the ASN1C internal ASN1CTAG structure. The output is standard ASN.1 notation for a tag (for example, [0] = context 0 tag).

Parameters:

tag The tag value to be converted.

buffer The pointer to a buffer to receive the stringified value.

bufsiz The size of buffer to receive the stringified value

Returns:

The converted value. This pointer will be equal to the buffer argument that was passed in.

const char* rtUInt64ToString (OSUINT64 value, char * buffer, size_t bufsiz)

This function converts a 64-bit ASN.1 value to a string.

Parameters:

value The value to convert.

buffer The pointer to a buffer to receive the stringified value.

bufsiz The size of the buffer to receive the stringified value.

Returns:

The converted value. This pointer will be equal to the buffer argument that was passed in.

const char* rtUIntToString (OSUINT32 value, char * buffer, size_t bufsiz)

This function converts an ASN.1 integer value to a string. In this case, the ASN.1 value was represented in the C/C++ code as an unsigned integer based on a constraint.

Parameters:

value The value to convert.

buffer The pointer to a buffer to receive the stringified value.

bufsiz The size of the buffer to receive the stringified value.

Returns:

The converted value. This pointer will be equal to the buffer argument that was passed in.

Binary Coded Decimal (BCD) Helper Functions

Detailed Description

Binary Coded Decimal (BCD) helper functions provide assistance in working with BCD numbers. Two functions are provided: one to convert to a BCD value to a string and one to perform the opposite conversion.

Functions

- `const char * rtBCDToString` (OSUINT32 numocts, const OSOCTET *data, char *buffer, size_t bufsiz)
 - `int rtStringToBCD` (const char *str, OSOCTET *bcdStr, size_t bufsiz)
 - `int rtStringToDynBCD` (ASN1CTXT *pctxt, const char *str, ASN1DynOctStr *pocstr)
-

Function Documentation

const char* rtBCDToString (OSUINT32 numocts, const OSOCTET * data, char * buffer, size_t bufsiz)

This function converts a packed BCD value to a standard null-terminated string. Octet value can contain the filler digit to represent the odd number of BCD digits. BCD digits can be 0(0000) to 9(1001). Filler digits can be A(1010), B(1011), C(1100), D(1101), E(1110) or F(1111)

Parameters:

numocts The number of octets in the BCD value.

data The pointer to the BCD value.

buffer The destination buffer. Should not be less than bufsiz argument is.

bufsiz The size of the destination buffer (in octets). The buffer size should be atleast ((numocts * 2) + 1) to hold the BCD to String conversion.

Returns:

The converted null-terminated string. NULL, if error has occurred or destination buffer is not enough.

int rtStringToBCD (const char * str, OSOCTET * bcdStr, size_t bufsiz)

This function converts a standard null-terminated string into a BCD value. The source string should contain only characters '0' - '9', 'A' - 'F', or 'a' - 'f'. Otherwise, an error will occur.

Parameters:

str The source standard null-terminated string.

bcdStr The destination buffer. Should not be less than bufsiz is.

bufsiz The size of the destination buffer (in octets).

Returns:

The number of octets in the resulting BCD value or a negative value if an error occurs.

int rtStringToDynBCD (ASN1CTXT * pctxt, const char * str, ASN1DynOctStr * pocstr)

This function converts a standard null-terminated string into a BCD value. The source string should contain only characters '0' - '9', 'A' - 'F', or 'a' - 'f'. Otherwise, an error will occur.

Parameters:

str The source standard null-terminated string.

pctxt Pointer to a context structure block.

pocstr Pointer to a dynamic octet string variable. Memory will be allocated for the data member of this structure with rtMemAlloc.

Returns:

The number of octets in the resulting BCD value or a negative value if an error occurs.

Character String Conversion Functions

Detailed Description

Common utility functions are provided to convert between standard null-terminated C strings and different ASN.1 string types.

Functions

- `const char * rtBMPToCString` (ASN1BMPString *pBMPString, char *cstring, OSUINT32 cstrsize)
- `const char * rtBMPToNewCString` (ASN1BMPString *pBMPString)
- `const char * rtBMPToNewCStringEx` (ASN1CTXT *pctxt, ASN1BMPString *pBMPString)
- `ASN1BMPString * rtCToBMPString` (ASN1CTXT *ctxt_p, const char *cstring, ASN1BMPString *pBMPString, Asn16BitCharSet *pCharSet)
- `OSBOOL rtIsIn16BitCharSet` (OSUNICHAR ch, Asn16BitCharSet *pCharSet)
- `const char * rtUCSToCString` (ASN1UniversalString *pUCSString, char *cstring, OSUINT32 cstrsize)
- `const char * rtUCSToNewCString` (ASN1UniversalString *pUCSString)
- `const char * rtUCSToNewCStringEx` (ASN1CTXT *pctxt, ASN1UniversalString *pUCSString)
- `ASN1UniversalString * rtCToUCSString` (ASN1CTXT *ctxt_p, const char *cstring, ASN1UniversalString *pUCSString, Asn132BitCharSet *pCharSet)
- `OSBOOL rtIsIn32BitCharSet` (OS32BITCHAR ch, Asn132BitCharSet *pCharSet)
- `wchar_t * rtUCSToWCSSString` (ASN1UniversalString *pUCSString, wchar_t *wcstring, OSUINT32 wcstrsize)
- `ASN1UniversalString * rtWCSToUCSString` (ASN1CTXT *ctxt_p, wchar_t *wcstring, ASN1UniversalString *pUCSString, Asn132BitCharSet *pCharSet)
- `int rtUTF8ToWCS` (ASN1CTXT *pCtxt, ASN1UTF8String inbuf, wchar_t *outbuf, size_t outbufsiz)
- `long rtWCSToUTF8` (ASN1CTXT *pCtxt, wchar_t *inbuf, size_t inlen, OSOCTET *outbuf, size_t outbufsiz)
- `int rtValidateUTF8` (ASN1CTXT *pCtxt, ASN1UTF8String inbuf)
- `int rtUTF8Len` (ASN1UTF8String inbuf)
- `int rtUTF8LenBytes` (ASN1UTF8String inbuf)
- `int rtUTF8CharSize` (OS32BITCHAR wc)
- `int rtUTF8EncodeChar` (OS32BITCHAR wc, OSOCTET *buf, int bufisz)
- `int rtUTF8DecodeChar` (ASN1CTXT *pCtxt, ASN1UTF8String pinbuf, int *pInsize)
- `char * rtUTF8Strdup` (ASN1CTXT *pctxt, ASN1UTF8String utf8str)
- `char * rtUTF8Strndup` (ASN1CTXT *pctxt, ASN1UTF8String utf8str, int nbytes)

Function Documentation

`const char* rtBMPToCString` (ASN1BMPString * *pBMPString*, char * *cstring*, OSUINT32 *cstrsize*)

This function converts a BMP string into a null-terminated C string. Any characters that are not 8-bit characters are discarded.

Parameters:

- pBMPString* A pointer to a BMP string structure to be converted.
- cstring* A pointer to a buffer to receive the converted string.

cstrsize The size of the buffer to receive the converted string.

Returns:

A pointer to the returned string structure. This is the *cstring* argument parameter value.

const char* rtBMPToNewCString (ASN1BMPString * *pBMPString*)

This function converts a BMP string into a null-terminated C string. Any characters that are not 8-bit characters are discarded. This function allocates dynamic memory to hold the converted string using the standard C run-time malloc function. The user is responsible for freeing this memory.

Parameters:

pBMPString A pointer to a BMP string structure to be converted.

Returns:

A pointer to the returned string structure. This is the *cstring* argument parameter value.

const char* rtBMPToNewCStringEx (ASN1CTXT * *pctxt*, ASN1BMPString * *pBMPString*)

This function converts a BMP string into a null-terminated C string. Any characters that are not 8-bit characters are discarded. In contrast to *rtBMPToNewCString*, this function allocates dynamic memory to hold the converted string using the *rtMemAlloc* function. The *rtMemFreePtr* should be called to release the allocated memory or the *rtmemFree* function should be called to release all memory allocated using the specified context block.

Parameters:

pctxt A pointer to a context structure.

pBMPString A pointer to a BMP string structure to be converted.

Returns:

A pointer to the returned string structure. This is the *cstring* argument parameter value.

ASN1BMPString* rtCToBMPString (ASN1CTXT * *ctxt_p*, const char * *cstring*, ASN1BMPString * *pBMPString*, Asn116BitCharSet * *pCharSet*)

This function converts a null-terminated C string into a 16-bit BMP string structure.

Parameters:

ctxt_p A pointer to a context string.

cstring A pointer to a null-terminated C string to be converted into a BMP string.

pBMPString A pointer to a BMP string structure to receive the converted string.

pCharSet A pointer to a character set structure describing the character set currently associated with the BMP character string type.

Returns:

A pointer to BMP string structure. This is the *pBMPString* argument parameter value.

ASN1UniversalString* rtCToUCSString (ASN1CTXT * *ctxt_p*, const char * *cstring*, ASN1UniversalString * *pUCSString*, Asn132BitCharSet * *pCharSet*)

This function converts a null-terminated C string into a 32-bit UCS-4 (Universal Character Set, 4 bytes) string structure.

Parameters:

ctxt_p A pointer to a context structure.

cstring A pointer to a null-terminated C string to be converted into a Universal string.

pUCSString A pointer to a Universal string structure to receive the converted string

pCharSet A pointer to a character structure describing the character set currently associated with the Universal character string type.

Returns:

A pointer to a Universal string structure. This is the `pUCSString` argument parameter value.

const char* rtUCSToCString (ASN1UniversalString * *pUCSString*, char * *cstring*, OSUINT32 *cstrsize*)

This function converts a Universal 32-bit string into a null-terminated C string. Any characters that are not 8-bit characters are discarded.

Parameters:

pUCSString A pointer to a Universal string structure to be converted.
cstring A pointer to a buffer to receive a converted string.
cstrsize The size of the buffer to receive the converted string.

Returns:

The pointer to the returned string. This is the `cstring` argument parameter value.

const char* rtUCSToNewCString (ASN1UniversalString * *pUCSString*)

This function converts a Universal 32-bit string into a null-terminated C string. Any characters that are not 8-bit characters are discarded. This function allocates dynamic memory to hold the converted string using the standard C run-time `malloc` function. The user is responsible for freeing this memory.

Parameters:

pUCSString A pointer to a Universal 32-bit string structure to be converted.

Returns:

A pointer to allocated null-terminated string. The user is responsible for freeing this memory.

const char* rtUCSToNewCStringEx (ASN1CTXT * *pctxt*, ASN1UniversalString * *pUCSString*)

This function converts a Universal 32-bit string into a null-terminated C string. Any characters that are not 8-bit characters are discarded. In contrast to `rtUCSToNewCString` this function allocates dynamic memory to hold the converted string using the `rtMemAlloc` function. The `rtMemFreePtr` should be called to release the allocated memory or the `rtMemFree` function should be called to release all memory allocated using the specified context block.

Parameters:

pctxt A pointer to a context block.
pUCSString A pointer to a Universal 32-bit string structure to be converted.

Returns:

A pointer to allocated null-terminated string. The user is responsible for freeing this memory.

wchar_t* rtUCSToWCSSString (ASN1UniversalString * *pUCSString*, wchar_t * *wcstring*, OSUINT32 *wcstrsize*)

This function converts a 32-bits encoded string to a wide character string.

Parameters:

pUCSString A pointer to a Universal string structure.
wcstring The pointer to the buffer to receive the converted string.
wcstrsize The number of wide characters (`wchar_t`) the outbuffer can hold.

Returns:

A character count or error status. This will be negative if the conversion fails. If the result is positive, the number of characters was written to *sctrsize*.

int rtUTF8CharSize (OS32BITCHAR *wc*)

This function will return the number of bytes needed to encode the given 32-bit universal character value as a UTF-8 character.

Parameters:

wc 32-bit wide character value.

Returns:

Number of bytes needed to encode as UTF-8.

int rtUTF8EncodeChar (OS32BITCHAR *wc*, OSOCTET * *buf*, int *bufsiz*)

This function will convert a wide character into an encoded UTF-8 character byte string.

Parameters:

wc 32-bit wide character value.

buf Buffer to receive encoded UTF-8 character value.

bufsiz Size of the buffer to receive the encoded value.

Returns:

Completion status of operation:

- 0 (ASN_OK) = success,
- negative return value is error.

int rtUTF8Len (ASN1UTF8String *inbuf*)

This function will return the length (in characters) of a null-terminated UTF-8 encoded string.

Parameters:

inbuf A pointer to the null-terminated UTF-8 encoded string.

Returns:

Number of characters in string. Note that this may be different than the number of bytes as UTF-8 characters can span multiple-bytes.

int rtUTF8LenBytes (ASN1UTF8String *inbuf*)

This function will return the length (in bytes) of a null-terminated UTF-8 encoded string.

Parameters:

inbuf A pointer to the null-terminated UTF-8 encoded string.

Returns:

Number of bytes in the string.

char* rtUTF8Strdup (ASN1CTXT * *pctxt*, ASN1UTF8String *utf8str*)

This function creates a duplicate copy of the given UTF-8 character string. It is similar to the C `strdup` function. Memory for the duplicated string is allocated using the `rtxMemAlloc` function.

Parameters:

pctxt A pointer to a context structure.

utf8str Null-terminated UTF-8 string to be duplicated.

Returns:

Pointer to duplicated string value.

char* rtUTF8Strdup (ASN1CTXT * *pctxt*, ASN1UTF8String *utf8str*, int *nbytes*)

This function creates a duplicate copy of the given UTF-8 character string. It is similar to the `rtUTF8Strdup` function except that it allows the number of bytes to convert to be specified. Memory for the duplicated string is allocated using the `rtMemAlloc` function.

Parameters:

pctxt A pointer to a context structure.
utf8str UTF-8 string to be duplicated.
nbytes Number of bytes from `utf8str` to duplicate.

Returns:

Pointer to duplicated string value.

int rtUTF8ToWCS (ASN1CTXT * *pCtxt*, ASN1UTF8String *inbuf*, wchar_t * *outbuf*, size_t *outbufsiz*)

This function converts a UTF-8 encoded string to a wide-character string.

Parameters:

pCtxt A pointer to a context structure.
inbuf A pointer to a null-terminated UTF-8 encoded string.
outbuf A pointer to a buffer to receive a converted string.
outbufsiz The number of wide characters (`wchar_t`) the outbuffer can hold.

Returns:

The character count or an error status. The status will be negative if the conversion fails, and positive indicating the number of characters written to `outbuf`.

int rtValidateUTF8 (ASN1CTXT * *pCtxt*, ASN1UTF8String *inbuf*)

This function will validate a UTF-8 encoded string to ensure that it is encoded correctly.

Parameters:

pCtxt A pointer to a context structure.
inbuf A pointer to the null-terminated UTF-8 encoded string.

Returns:

Completion status of operation:

- 0 (ASN_OK) = success,
- negative return value is error.

ASN1UniversalString* rtWCSToUCSString (ASN1CTXT * *ctxt_p*, wchar_t * *wcstring*, ASN1UniversalString * *pUCSString*, Asn132BitCharSet * *pCharSet*)

This function converts a wide-character string to a Universal 32-bits encoded string.

Parameters:

ctxt_p A pointer to a context structure.
wcstring The pointer to the wide-character (Unicode) string to convert
pUCSString The pointer to the Universal String structure to receive the converted string.
pCharSet The pointer to the character set structure describing the character set currently associated with the Universal character string type.

Returns:

If the conversion of the WCS to the UTF-8 was successful, the number of bytes in the converted string is returned. If the encoding fails, a negative status value is returned.

long rtWCSToUTF8 (ASN1CTXT * pCtxt, wchar_t * inbuf, size_t inlen, OSOCTET * outbuf, size_t outbufsiz)

This function converts a wide-character string to a UTF-8 encoded string.

Parameters:

- pCtxt* A pointer to a context structure.
- inbuf* The pointer to a wide-character (Unicode) string to convert.
- inlen* The number of character in the Unicode string.
- outbuf* The pointer to a buffer to receive the converted string.
- outbufsiz* The size (in bytes) or the output buffer to receive the encoded string.

Returns:

If the conversion of WCS to UTF-8 is successful, the number of bytes in the converted string is returned. If the encoding fails, a negative status value is returned.

Big Integer Helper Functions

Detailed Description

Arbitrary-precision integer manipulating functions are used to maintain big integers used within the ASN.1 run-time functions. Big integer variables are passed in and out of ASN.1 C encode and decode functions as character strings. Internally, a special structure is used for performing an arbitrary-precision match.

Functions

- void **rtBigIntInit** (ASN1BigInt *pInt)
- int **rtBigIntSetStr** (ASN1CTXT *pCtxt, ASN1BigInt *pInt, const char *value, int radix)
- int **rtBigIntSetInt64** (ASN1CTXT *pCtxt, ASN1BigInt *pInt, OSINT64 value)
- int **rtBigIntSetBytes** (ASN1CTXT *pCtxt, ASN1BigInt *pInt, OSOCTET *value, size_t vallen)
- size_t **rtBigIntGetDataLen** (ASN1BigInt *pInt)
- long **rtBigIntGetData** (ASN1CTXT *pCtxt, ASN1BigInt *pInt, OSOCTET *buffer, size_t bufSize)
- int **rtBigIntDigitsNum** (ASN1BigInt *pInt, int radix)
- int **rtBigIntCopy** (ASN1CTXT *pCtxt, const ASN1BigInt *pSrc, ASN1BigInt *pDst)
- int **rtBigIntFastCopy** (ASN1CTXT *pCtxt, ASN1BigInt *pSrc, ASN1BigInt *pDst)
- int **rtBigIntToString** (ASN1CTXT *pCtxt, ASN1BigInt *pInt, int radix, char *str, size_t strSize)
- int **rtBigIntPrint** (const char *name, ASN1BigInt *bigint, int radix)
- int **rtBigIntCompare** (const ASN1BigInt *arg1, const ASN1BigInt *arg2)
- int **rtBigIntStrCompare** (ASN1CTXT *pCtxt, const char *arg1, const char *arg2)
- void **rtBigIntFree** (ASN1CTXT *pCtxt, ASN1BigInt *pInt)

Function Documentation

int rtBigIntCompare (const ASN1BigInt * arg1, const ASN1BigInt * arg2)

This function compares two big integer values. The result of comparison can be -1 (first value less than the second one), 0 (both values are equal), and 1 (the first value is greater than the second one).

Parameters:

arg1 The first big integer value to compare.
arg2 The second big integer value to compare.

Returns:

The result of the comparison (-1, 0, 1).

int rtBigIntCopy (ASN1CTXT * *pCtxt*, const ASN1BigInt * *pSrc*, ASN1BigInt * *pDst*)

This function copies one big integer structure into another one. The destination big integer structure should not be initialized yet.

Parameters:

pCtxt A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pSrc The source big integer value to copy.
pDst The destination big integer value to receive the copied value. This should not be initialized.

Returns:

The status value of the operation.

int rtBigIntDigitsNum (ASN1BigInt * *pInt*, int *radix*)

This function returns the approximate number of digits of the big integer value according to the specified radix. This function may be used to calculate the size of the buffer for the `rtBigIntToString` function. The number of digits might be slightly greater than really necessary, but never less.

Parameters:

pInt A pointer to a big integer structure.
radix The base of value. Must be 2, 8, 10, or 16.

Returns:

The approximated number of digits of the big integer.

int rtBigIntFastCopy (ASN1CTXT * *pCtxt*, ASN1BigInt * *pSrc*, ASN1BigInt * *pDst*)

This function copies one big structure into another one. The destination big integer structure should be initialized before use of this function. This function might be faster than `rtBigIntCopy` because if the destination big integer already has enough allocated memory then memory will be reused without allocation.

Parameters:

pCtxt A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pSrc The source big integer value to copy.
pDst The destination big integer value to receive the copied value. This should be initialized.

Returns:

The status value of the operation.

void rtBigIntFree (ASN1CTXT * *pCtxt*, ASN1BigInt * *pInt*)

This function releases memory if it was called for the current big integer. If the memory was not allocated or static memory was used, this function does nothing. It is recommended to not use the big integer structure after the `rtBigIntFree` is invoked for it.

Parameters:

pCtxt A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pInt A pointer to a big integer structure that will be freed.

long rtBigIntGetData (ASN1CTXT * *pCtxt*, ASN1BigInt * *pInt*, OSOCTET * *buffer*, size_t *bufSize*)

This function copies the two's complement binary representation of a big integer into an octet string. The output will be in big-endian octet-order: the most significant octet will be in the zeroth element.

Parameters:

pCtxt A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pInt A pointer to the big integer structure.
buffer A pointer to an octet array to receive the two's complement binary representation of a arbitrary-precision integer.
bufSize The length of an octet array.

Returns:

If positive, number of octets actually copied to the buffer. If negative - the status value of the operation.

size_t rtBigIntGetDataLen (ASN1BigInt * *pInt*)

This function returns the number of octets necessary for storing a big integer value in the octet array. This function may be used for calculating the necessary buffer size for the `rtBigIntGetData`.

Parameters:

pInt A pointer to the big integer structure onto which the value is to be stored.

Returns:

The number of octets, necessary for storing a big integer value in the octet string.

void rtBigIntInit (ASN1BigInt * *pInt*)

This function initializes a big integer structure. This function should be called before the first use of the big integer.

Parameters:

pInt A pointer to the big integer structure to be initialized.

int rtBigIntPrint (const char * *name*, ASN1BigInt * *bigint*, int *radix*)

This function prints a big integer value to standard output (stdout), according to the specified radix. In general, this function is very similar to the `rtBigIntToString` function. It simply prints the output value to standard output in a "name = value" format.

Parameters:

name The name of the variable to print.
bigint The big integer value to print.
radix The base value for conversion to print. It must be 2, 8, 10, or 16.

int rtBigIntSetBytes (ASN1CTXT * pCtxt, ASN1BigInt * pInt, OSOCTET * value, size_t vallen)

This function translates an octet array containing the two's complement binary representation of an arbitrary-precision integer into a bigger structure. The input array is assumed to be in big-endian octet-order: the most significant octet is in the zeroth element.

Parameters:

pCtxt A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pInt A pointer to the big integer structure onto which the value is to be stored. A pointer to an updated big integer structure.
vallen The length of the array.
value The pointer to an octet array with two's complement binary representation of an arbitrary-precision integer.

Returns:

The status value of the operation.

int rtBigIntSetInt64 (ASN1CTXT * pCtxt, ASN1BigInt * pInt, OSINT64 value)

This function converts the ASN1INT64 value to a big integer structure. An ASN1INT64 type is a 64-bit integer if the platform supports 64-bit integers. In the other case, it will be a 32-bit integer.

Parameters:

pCtxt A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pInt A pointer to the big integer structure onto which the value is to be stored. A pointer to an updated big integer structure.
value A pointer to a character string to be converted.

Returns:

The status value of the operation.

int rtBigIntSetStr (ASN1CTXT * pCtxt, ASN1BigInt * pInt, const char * value, int radix)

This function converts the character string to a big integer structure. The string might contain the prefix to indicate the radix ('0x' - 16, '0o' - 8, '0b' - 2, otherwise 10). In this case the radix argument should be set to 0.

Parameters:

pCtxt A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pInt A pointer to the big integer structure onto which the value is to be stored. A pointer to an updated big integer structure.
value A pointer to a character string to be converted.
radix The base value of pString. Must be 0, 2, 8, 10 or 16. If radix is 0 then the radix will be set according to string prefix ('0x' - 16, '0o' - 8, '0b' - 2, otherwise 10).

Returns:

The status value of the operation.

int rtBigIntStrCompare (ASN1CTXT * pCtxt, const char * arg1, const char * arg2)

This function compares two big integer values, which are represented as strings. The strings might contain prefixes to indicate the actual radix of the big integer value ('0x' - 16, '0o' - 8, '0b' - 2, otherwise 10). The result of comparison can be -1 (first value less than the second one), 0 (both values are equal), and 1 (the first value is greater than the second one).

Parameters:

pCtxt A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.

arg1 The first big integer value to compare.

arg2 The second big integer value to compare.

Returns:

The result of the comparison (-1, 0, 1).

int rtBigIntToString (ASN1CTXT * pCtxt, ASN1BigInt * pInt, int radix, char * str, size_t strSize)

This function converts a big integer to a string according to the specified radix.

Parameters:

pCtxt A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.

pInt A pointer to a big integer structure to be converted into a string.

radix A base value for conversion. It must be 2, 8, 10, or 16.

strSize The size of str.

str A pointer to the buffer to receive the converted character string.

Returns:

The status value of the operation.

Time Helper Functions

Detailed Description

These are helper functions for work with time strings in different formats. *rtMake** functions create time strings, *rtParse** ones parse time string into the C structure *OSDATETIME*. There are implementations of these functions for ASN.1 GeneralizedTime, UTCTime and XML dateTime formats.

Functions

- int **rtSetLocalTime** (*OSDATETIME* *dateTime, time_t time, OSBOOL diffTime)
 - int **rtMakeGeneralizedTime** (*ASN1CTXT* *pctx, const *OSDATETIME* *dateTime, char **outdata, int outdataSize)
 - int **rtMakeUTCTime** (*ASN1CTXT* *pctx, const *OSDATETIME* *dateTime, char **outdata, int outdataSize)
 - int **rtMakeXmlDateTime** (*ASN1CTXT* *pctx, const *OSDATETIME* *dateTime, char **outdata, int outdataSize)
 - int **rtParseGeneralizedTime** (*ASN1CTXT* *pctx, const char *value, *OSDATETIME* *dateTime)
 - int **rtParseUTCTime** (*ASN1CTXT* *pctx, const char *value, *OSDATETIME* *dateTime)
 - int **rtParseXmlDateTime** (*ASN1CTXT* *pctx, const char *value, *OSDATETIME* *dateTime)
-

Function Documentation

int rtMakeGeneralizedTime (ASN1CTXT * *pctxt*, const OSDATETIME * *dateTime*, char ** *outdata*, int *outdataSize*)

This function creates a time string in ASN.1 GeneralizedTime format as specified in the X.680 ITU-T standard.

Parameters:

pctxt A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.

dateTime A pointer to the OSDATETIME structure, that contains components of the date and time.

outdata A pointer to a pointer to a destination string. If *outdataSize* is non-zero, it should be a pointer to a pointer to an actual array. Otherwise, the memory will be allocated and the pointer will be stored in the *outdata*.

outdataSize A size of *outdata* (in octets). If zero, the memory for the *outdata* will be allocated. If not, the *outdata*'s size should be big enough to receive the generated time string. Otherwise, error code will be returned.

Returns:

Completion status of operation:

- 0 (ASN_OK) = success,
- negative return value is error.

int rtMakeUTCTime (ASN1CTXT * *pctxt*, const OSDATETIME * *dateTime*, char ** *outdata*, int *outdataSize*)

This function creates a time string in ASN.1 UTCTime format as specified in the X.680 ITU-T standard.

Parameters:

pctxt A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.

dateTime A pointer to the OSDATETIME structure, that contains components of the date and time.

outdata A pointer to a pointer to a destination string. If *outdataSize* is non-zero, it should be a pointer to a pointer to an actual array. Otherwise, the memory will be allocated and the pointer will be stored in the *outdata*.

outdataSize A size of *outdata* (in octets). If zero, the memory for the *outdata* will be allocated. If not, the *outdata*'s size should be big enough to receive the generated time string. Otherwise, error code will be returned.

Returns:

Completion status of operation:

- 0 (ASN_OK) = success,
- negative return value is error.

int rtMakeXmlDateTime (ASN1CTXT * *pctxt*, const OSDATETIME * *dateTime*, char ** *outdata*, int *outdataSize*)

This function creates a time string in XML *dateTime* format as specified in the XML schema W3C Recommendation.

Parameters:

pctxt A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.

dateTime A pointer to the OSDATETIME structure, that contains components of the date and time.

outdata A pointer to a pointer to a destination string. If *outdataSize* is non-zero, it should be a pointer to a pointer to an actual array. Otherwise, the memory will be allocated and the pointer will be stored in the *outdata*.

outdataSize A size of *outdata* (in octets). If zero, the memory for the *outdata* will be allocated. If not, the *outdata*'s size should be big enough to receive the generated time string. Otherwise, error code will be returned.

Returns:

Completion status of operation:

- 0 (ASN_OK) = success,
- negative return value is error.

int rtParseGeneralizedTime (ASN1CTXT * *pctxt*, const char * *value*, OSDATETIME * *dateTime*)

This function parses a time string that is represented in ASN.1 GeneralizedTime format as specified in the X.680 ITU-T standard. It stores the parsing result in OSDATETIME C structure.

Parameters:

pctxt A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.

value A pointer to the time string to be parsed.

dateTime A pointer to the destination OSDATETIME structure.

Returns:

Completion status of operation:

- 0 (ASN_OK) = success,
- negative return value is error.

int rtParseUTCTime (ASN1CTXT * *pctxt*, const char * *value*, OSDATETIME * *dateTime*)

This function parses a time string that is represented in ASN.1 UTCTime format as specified in the X.680 ITU-T standard. It stores the parsing result in OSDATETIME C structure.

Parameters:

pctxt A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.

value A pointer to the time string to be parsed.

dateTime A pointer to the destination OSDATETIME structure.

Returns:

Completion status of operation:

- 0 (ASN_OK) = success,
- negative return value is error.

int rtParseXmlDateTime (ASN1CTXT * *pctxt*, const char * *value*, OSDATETIME * *dateTime*)

This function parses a time string that is represented in XML dateTime format as specified in the XML schema W3C Recommendation. It stores the parsing result in OSDATETIME C structure.

Parameters:

pctxt A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.

value A pointer to the time string to be parsed.

dateTime A pointer to the destination OSDATETIME structure.

Returns:

Completion status of operation:

- 0 (ASN_OK) = success,
- negative return value is error.

int rtSetLocalTime (OSDATETIME * *dateTime*, time_t *time*, OSBOOL *diffTime*)

This function converts *time_t* value to the OSDATETIME structure and corrects for the local time zone

Parameters:

dateTime A pointer to the OSDATETIME structure, that contains components of the date and time.

time A time_t type value. That will be converted to the OSDATETIME structure.

diffTime TRUE means the difference between local time and UTC will be calculated; FALSE means only local time will be stored.

Returns:

Completion status of operation:

- 0 (ASN_OK) = success,
- negative return value is error.

Comparison Functions

Detailed Description

The group of functions allows comparing the values of primitive ASN.1 types. These functions are used in the comparison routines that are generated by the ASN1C compiler when the *-gencompare* option is used.

Information on elements that do not match is written to the given error buffer for each function. This makes it possible to compare complex structures and get back specific information as to what elements within those structures are different.

Functions

- ASN1BOOL **rtCmpBoolean** (ASN1ConstCharPtr name, ASN1BOOL value, ASN1BOOL compValue, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpInt8** (ASN1ConstCharPtr name, ASN1INT8 value, ASN1INT8 compValue, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpSInt** (ASN1ConstCharPtr name, ASN1SINT value, ASN1SINT compValue, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpUInt8** (ASN1ConstCharPtr name, ASN1UINT8 value, ASN1UINT8 compValue, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpUSInt** (ASN1ConstCharPtr name, ASN1USINT value, ASN1USINT compValue, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpInteger** (ASN1ConstCharPtr name, ASN1INT value, ASN1INT compValue, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpUnsigned** (ASN1ConstCharPtr name, ASN1UINT value, ASN1UINT compValue, char *errBuff, int errBuffSize)

- ASN1BOOL **rtCmpInt64** (ASN1ConstCharPtr name, ASN1INT64 value, ASN1INT64 compValue, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpUInt64** (ASN1ConstCharPtr name, ASN1UINT64 value, ASN1UINT64 compValue, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpBitStr** (ASN1ConstCharPtr name, ASN1UINT numbits, ASN1ConstOctetPtr data, ASN1UINT compNumbits, ASN1ConstOctetPtr compData, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpOctStr** (ASN1ConstCharPtr name, ASN1UINT numocts, ASN1ConstOctetPtr data, ASN1UINT compNumocts, ASN1ConstOctetPtr compData, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpCharStr** (ASN1ConstCharPtr name, ASN1ConstCharPtr cstring, ASN1ConstCharPtr compCstring, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmp16BitCharStr** (ASN1ConstCharPtr name, Asn116BitCharString *bstring, Asn116BitCharString *compBstring, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmp32BitCharStr** (ASN1ConstCharPtr name, Asn132BitCharString *bstring, Asn132BitCharString *compBstring, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpReal** (ASN1ConstCharPtr name, ASN1REAL value, ASN1REAL compValue, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpOID** (ASN1ConstCharPtr name, ASN1OBJID *pOID, ASN1OBJID *pcompOID, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpOIDValue** (ASN1ConstCharPtr name, ASN1OBJID *pOID, ASN1OBJID *pcompOID, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpOID64** (ASN1ConstCharPtr name, ASN1OID64 *pOID, ASN1OID64 *pcompOID, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpOID64Value** (ASN1ConstCharPtr name, ASN1OID64 *pOID, ASN1OID64 *pcompOID, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpOpenType** (ASN1ConstCharPtr name, ASN1UINT numocts, ASN1ConstOctetPtr data, ASN1UINT compNumocts, ASN1ConstOctetPtr compData, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpOpenTypeExt** (ASN1ConstCharPtr name, Asn1RTDList *pElemList, Asn1RTDList *pCompElemList, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpTag** (ASN1ConstCharPtr name, int tag, int compTag, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpSeqOfElements** (ASN1ConstCharPtr name, int noOfElems, int compNoOfElems, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpOptional** (ASN1ConstCharPtr name, unsigned presentBit, unsigned compPresentBit, char *errBuff, int errBuffSize)

Function Documentation

ASN1BOOL rtCmp16BitCharStr (ASN1ConstCharPtr name, Asn116BitCharString * bstring, Asn116BitCharString * compBstring, char * errBuff, int errBuffSize)

The `rtCmp16BitCharStr` function compares two ASN.1 16-bit character string values (including BMPString).

Parameters:

- name* The name of value. Used for constructing `errBuff` if values are not matching.
- bstring* The pointer to the first 16-bit character string structure to compare.
- compBstring* The pointer to the second 16-bit character string structure to compare.
- errBuff* The pointer to the buffer to receive the null-terminated string if the comparison fails. If the comparison failed, this buffer will contain the null-terminated string that explains the reason of comparison failure.
- errBuffSize* The size of the `errBuff` buffer.

Returns:

The comparison result. TRUE, if values matched, otherwise FALSE.

ASN1BOOL rtCmp32BitCharStr (ASN1ConstCharPtr name, Asn132BitCharString * bstring, Asn132BitCharString * compBstring, char * errBuff, int errBuffSize)

The rtCmp32BitCharStr function compares two 32-bit character string values (including UniversalString).

Parameters:

name The name of value. Used for constructing errBuff if values are not matching.

bstring The pointer to the first 32-bit character string structure to compare.

compBstring The pointer to the second 32-bit character string structure to compare.

errBuff The pointer to the buffer to receive the null-terminated string if the comparison fails. If the comparison failed, this buffer will contain the null-terminated string that explains the reason of comparison failure.

errBuffSize The size of the errBuff buffer.

Returns:

The comparison result. TRUE, if values matched, otherwise FALSE.

ASN1BOOL rtCmpBitStr (ASN1ConstCharPtr name, ASN1UINT numbits, ASN1ConstOctetPtr data, ASN1UINT compNumbits, ASN1ConstOctetPtr compData, char * errBuff, int errBuffSize)

The rtCmpBitStr function compares two ASN.1 BIT STRING values.

Parameters:

name The name of value. Used for constructing errBuff if values are not matching.

numbits The number of bits in the first value to compare.

data The pointer to the data of the first value to compare.

compNumbits The number of bits in the second value to compare.

compData The pointer to the data of the second value to compare.

errBuff The pointer to the buffer to receive the null-terminated string if the comparison fails. If the comparison failed, this buffer will contain the null-terminated string that explains the reason of comparison failure.

errBuffSize The size of the errBuff buffer.

Returns:

The comparison result. TRUE, if values matched, otherwise FALSE.

ASN1BOOL rtCmpBoolean (ASN1ConstCharPtr name, ASN1BOOL value, ASN1BOOL compValue, char * errBuff, int errBuffSize)

The rtCmpBoolean function compares two ASN.1 Boolean values. The return value is TRUE (matched) or FALSE (unmatched).

Parameters:

name The name of value. Used for constructing errBuff if values are not matching.

value First value to compare.

compValue Second value to compare.

errBuff The pointer to the buffer to receive the null-terminated string if the comparison fails. If the comparison failed, this buffer will contain the null-terminated string that explains the reason of comparison failure.

errBuffSize The size of the errBuff buffer.

Returns:

The comparison result. TRUE, if values matched, otherwise FALSE.

ASN1BOOL rtCmpCharStr (ASN1ConstCharPtr name, ASN1ConstCharPtr cstring, ASN1ConstCharPtr compCstring, char * errBuff, int errBuffSize)

The rtCmpCharStr function compares two ASN.1 8-bit character string values (including IA5String, VisibleString, PrintableString, NumericString, etc.)

Parameters:

name The name of value. Used for constructing errBuff if values are not matching.
cstring The first standard null-terminated string to compare.
compCstring The second standard null-terminated string to compare.
errBuff The pointer to the buffer to receive the null-terminated string if the comparison fails. If the comparison failed, this buffer will contain the null-terminated string that explains the reason of comparison failure.
errBuffSize The size of the errBuff buffer.

Returns:

The comparison result. TRUE, if values matched, otherwise FALSE.

ASN1BOOL rtCmpInt64 (ASN1ConstCharPtr name, ASN1INT64 value, ASN1INT64 compValue, char * errBuff, int errBuffSize)

The rtCmpInt64 function compares two 64-bit ASN.1 INTEGER values.

Parameters:

name The name of value. Used for constructing errBuff if values are not matching.
value First value to compare.
compValue Second value to compare.
errBuff The pointer to the buffer to receive the null-terminated string if the comparison fails. If the comparison failed, this buffer will contain the null-terminated string that explains the reason of comparison failure.
errBuffSize The size of the errBuff buffer.

Returns:

The comparison result. TRUE, if values matched, otherwise FALSE.

ASN1BOOL rtCmpInteger (ASN1ConstCharPtr name, ASN1INT value, ASN1INT compValue, char * errBuff, int errBuffSize)

The rtCmpInteger function compares two ASN.1 INTEGER values.

Parameters:

name The name of value. Used for constructing errBuff if values are not matching.
value First value to compare.
compValue Second value to compare.
errBuff The pointer to the buffer to receive the null-terminated string if the comparison fails. If the comparison failed, this buffer will contain the null-terminated string that explains the reason of comparison failure.
errBuffSize The size of the errBuff buffer.

Returns:

The comparison result. TRUE, if values matched, otherwise FALSE.

ASN1BOOL rtCmpOctStr (ASN1ConstCharPtr name, ASN1UINT numocts, ASN1ConstOctetPtr data, ASN1UINT compNumocts, ASN1ConstOctetPtr compData, char * errBuff, int errBuffSize)

The rtCmpOctStr function compares two ASN.1 OCTET STRING values.

Parameters:

name The name of value. Used for constructing errBuff if values are not matching.

numocts The number of the octets in the first value to compare.
data The pointer to the data of the first value to compare.
compNumocts The number of the octets in the second value to compare.
compData The pointer to the data of the second value to compare.
errBuff The pointer to the buffer to receive the null-terminated string if the comparison fails. If the comparison failed, this buffer will contain the null-terminated string that explains the reason of comparison failure.
errBuffSize The size of the *errBuff* buffer.

Returns:

The comparison result. TRUE, if values matched, otherwise FALSE.

ASN1BOOL rtCmpOID (ASN1ConstCharPtr name, ASN1OBJID * pOID, ASN1OBJID * pcompOID, char * errBuff, int errBuffSize)

The *rtCmpOID* function compares two ASN.1 OBJECT IDENTIFIER or RELATIVE-OID values.

Parameters:

name The name of value. Used for constructing *errBuff* if values are not matching.
pOID The pointer to the first value to compare.
pcompOID The pointer to the second value to compare.
errBuff The pointer to the buffer to receive the null-terminated string if the comparison fails. If the comparison failed, this buffer will contain the null-terminated string that explains the reason of comparison failure.
errBuffSize The size of the *errBuff* buffer.

Returns:

The comparison result. TRUE, if values matched, otherwise FALSE.

ASN1BOOL rtCmpOID64 (ASN1ConstCharPtr name, ASN1OID64 * pOID, ASN1OID64 * pcompOID, char * errBuff, int errBuffSize)

The *rtCmpOID64* function compares two 64-bit ASN.1 OBJECT IDENTIFIER or RELATIVE-OID values.

Parameters:

name The name of value. Used for constructing *errBuff* if values are not matching.
pOID The pointer to the first value to compare.
pcompOID The pointer to the second value to compare.
errBuff The pointer to the buffer to receive the null-terminated string if the comparison fails. If the comparison failed, this buffer will contain the null-terminated string that explains the reason of comparison failure.
errBuffSize The size of the *errBuff* buffer.

Returns:

The comparison result. TRUE, if values matched, otherwise FALSE.

ASN1BOOL rtCmpOpenType (ASN1ConstCharPtr name, ASN1UINT numocts, ASN1ConstOctetPtr data, ASN1UINT compNumocts, ASN1ConstOctetPtr compData, char * errBuff, int errBuffSize)

The *rtCmpOpenType* function compares two ASN.1 values of the old (pre- 1994) ASN.1 ANY type or other elements defined in the later standards to be Open Types (for example, a variable type declaration in a CLASS construct defined in X.681).

Parameters:

name The name of value. Used for constructing *errBuff* if values are not matching.
numocts The number of octets in the first value to compare.

data The pointer to the data of the first value to compare.
compNumocts The number of octets in the second value to compare.
compData The pointer to the data of the second value to compare.
errBuff The pointer to the buffer to receive the null-terminated string if the comparison fails. If the comparison failed, this buffer will contain the null-terminated string that explains the reason of comparison failure.
errBuffSize The size of the *errBuff* buffer.

Returns:

The comparison result. TRUE, if values matched, otherwise FALSE.

ASN1BOOL rtCmpOpenTypeExt (ASN1ConstCharPtr name, Asn1RTDList * pElemList, Asn1RTDList * pCompElemList, char * errBuff, int errBuffSize)

The *rtCmpOpenTypeExt* function compares two ASN.1 open type extension values.

An open type extension is defined as an extensibility marker on a constructed type without any extension elements defined (for example, SEQUENCE {a INTEGER, ...}). The difference is that this is an implicit field that can span one or more elements whereas the standard Open Type is assumed to be a single tagged field.

Parameters:

name The name of value. Used for constructing *errBuff* if values are not matching.
pElemList The first pointer to a linked list structure. The list should consist of ASN1OpenType elements.
pCompElemList The second pointer to a linked list structure. The list should consist of ASN1OpenType elements.
errBuff The pointer to the buffer to receive the null-terminated string if the comparison fails. If the comparison failed, this buffer will contain the null-terminated string that explains the reason of comparison failure.
errBuffSize The size of the *errBuff* buffer.

Returns:

The comparison result. TRUE, if values matched, otherwise FALSE.

ASN1BOOL rtCmpReal (ASN1ConstCharPtr name, ASN1REAL value, ASN1REAL compValue, char * errBuff, int errBuffSize)

The *rtCmpReal* function compares two ASN.1 REAL values.

Parameters:

name The name of value. Used for constructing *errBuff* if values are not matching.
value First value to compare.
compValue Second value to compare.
errBuff The pointer to the buffer to receive the null-terminated string if the comparison fails. If the comparison failed, this buffer will contain the null-terminated string that explains the reason of comparison failure.
errBuffSize The size of the *errBuff* buffer.

Returns:

The comparison result. TRUE, if values matched, otherwise FALSE.

ASN1BOOL rtCmpUInt64 (ASN1ConstCharPtr name, ASN1UINT64 value, ASN1UINT64 compValue, char * errBuff, int errBuffSize)

The *rtCmpUInt64* function compares two 64-bit ASN.1 unsigned INTEGER values.

Parameters:

name The name of value. Used for constructing *errBuff* if values are not matching.
value First value to compare.

compValue Second value to compare.

errBuff The pointer to the buffer to receive the null-terminated string if the comparison fails. If the comparison failed, this buffer will contain the null-terminated string that explains the reason of comparison failure.

errBuffSize The size of the *errBuff* buffer.

Returns:

The comparison result. TRUE, if values matched, otherwise FALSE.

ASN1BOOL rtCmpUnsigned (ASN1ConstCharPtr name, ASN1UINT value, ASN1UINT compValue, char * errBuff, int errBuffSize)

The *rtCmpUnsigned* function compares two ASN.1 unsigned INTEGER values.

Parameters:

name The name of value. Used for constructing *errBuff* if values are not matching.

value First value to compare.

compValue Second value to compare.

errBuff The pointer to the buffer to receive the null-terminated string if the comparison fails. If the comparison failed, this buffer will contain the null-terminated string that explains the reason of comparison failure.

errBuffSize The size of the *errBuff* buffer.

Returns:

The comparison result. TRUE, if values matched, otherwise FALSE.

Comparison to Standard Output Functions

Detailed Description

The *rtCmpToStdout<type>* functions do the same actions as the *rtCmp<type>* ones, but they print the comparison results to *stdout* instead of putting it into the buffer. The prototypes of these functions are almost the same as for the *rtCmp<type>* except the last two parameters - they are absent in the *rtCmpToStdout<type>* functions.

Functions

- ASN1BOOL **rtCmpToStdoutBoolean** (ASN1ConstCharPtr name, ASN1BOOL value, ASN1BOOL compValue)
- ASN1BOOL **rtCmpToStdoutInteger** (ASN1ConstCharPtr name, ASN1INT value, ASN1INT compValue)
- ASN1BOOL **rtCmpToStdoutInt64** (ASN1ConstCharPtr name, ASN1INT64 value, ASN1INT64 compValue)
- ASN1BOOL **rtCmpToStdoutUnsigned** (ASN1ConstCharPtr name, ASN1UINT value, ASN1UINT compValue)
- ASN1BOOL **rtCmpToStdoutUInt64** (ASN1ConstCharPtr name, ASN1UINT64 value, ASN1UINT64 compValue)
- ASN1BOOL **rtCmpToStdoutBitStr** (ASN1ConstCharPtr name, ASN1UINT numbits, ASN1ConstOctetPtr data, ASN1UINT compNumbits, ASN1ConstOctetPtr compData)
- ASN1BOOL **rtCmpToStdoutOctStr** (ASN1ConstCharPtr name, ASN1UINT numocts, ASN1ConstOctetPtr data, ASN1UINT compNumocts, ASN1ConstOctetPtr compData)

- ASN1BOOL **rtCmpToStdoutCharStr** (ASN1ConstCharPtr name, ASN1ConstCharPtr cstring, ASN1ConstCharPtr compCstring)
- ASN1BOOL **rtCmpToStdout16BitCharStr** (ASN1ConstCharPtr name, Asn116BitCharString *bstring, Asn116BitCharString *compBstring)
- ASN1BOOL **rtCmpToStdout32BitCharStr** (ASN1ConstCharPtr name, Asn132BitCharString *bstring, Asn132BitCharString *compBstring)
- ASN1BOOL **rtCmpToStdoutReal** (ASN1ConstCharPtr name, ASN1REAL value, ASN1REAL compValue)
- ASN1BOOL **rtCmpToStdoutOID** (ASN1ConstCharPtr name, ASN1OBJID *pOID, ASN1OBJID *pcompOID)
- ASN1BOOL **rtCmpToStdoutOIDValue** (ASN1ConstCharPtr name, ASN1OBJID *pOID, ASN1OBJID *pcompOID)
- ASN1BOOL **rtCmpToStdoutOID64** (ASN1ConstCharPtr name, ASN1OID64 *pOID, ASN1OID64 *pcompOID)
- ASN1BOOL **rtCmpToStdoutOID64Value** (ASN1ConstCharPtr name, ASN1OID64 *pOID, ASN1OID64 *pcompOID)
- ASN1BOOL **rtCmpToStdoutOpenType** (ASN1ConstCharPtr name, ASN1UINT numocts, ASN1ConstOctetPtr data, ASN1UINT compNumocts, ASN1ConstOctetPtr compData)
- ASN1BOOL **rtCmpToStdoutOpenTypeExt** (ASN1ConstCharPtr name, Asn1RTDList *pElemList, Asn1RTDList *pCompElemList)
- ASN1BOOL **rtCmpToStdoutTag** (ASN1ConstCharPtr name, int tag, int compTag)
- ASN1BOOL **rtCmpToStdoutSeqOfElements** (ASN1ConstCharPtr name, int noOfElems, int compNoOfElems)
- ASN1BOOL **rtCmpToStdoutOptional** (ASN1ConstCharPtr name, unsigned presentBit, unsigned compPresentBit)

Function Documentation

ASN1BOOL rtCmpToStdout16BitCharStr (ASN1ConstCharPtr name, Asn116BitCharString * bstring, Asn116BitCharString * compBstring)

Parameters:

- name* The name of value.
- bstring* The first value to compare.
- compBstring* The second value to compare.

ASN1BOOL rtCmpToStdout32BitCharStr (ASN1ConstCharPtr name, Asn132BitCharString * bstring, Asn132BitCharString * compBstring)

Parameters:

- name* The name of value.
- bstring* The first value to compare.
- compBstring* The second value to compare.

ASN1BOOL rtCmpToStdoutBitStr (ASN1ConstCharPtr name, ASN1UINT numbits, ASN1ConstOctetPtr data, ASN1UINT compNumbits, ASN1ConstOctetPtr compData)

Parameters:

- name* The name of value.
- numbits* The first value to compare.

data The pointer to the first value.
compNumbits The second value to compare.
compData The pointer to the second value.

ASN1BOOL rtCmpToStdoutBoolean (ASN1ConstCharPtr *name*, ASN1BOOL *value*, ASN1BOOL *compValue*)

Parameters:

name The name of value.
value The first value to compare.
compValue The second value to compare.

ASN1BOOL rtCmpToStdoutCharStr (ASN1ConstCharPtr *name*, ASN1ConstCharPtr *cstring*, ASN1ConstCharPtr *compCString*)

Parameters:

name The name of value.
cstring The first value to compare.
compCString The second value to compare.

ASN1BOOL rtCmpToStdoutInt64 (ASN1ConstCharPtr *name*, ASN1INT64 *value*, ASN1INT64 *compValue*)

Parameters:

name The name of value.
value The first value to compare.
compValue The second value to compare.

ASN1BOOL rtCmpToStdoutInteger (ASN1ConstCharPtr *name*, ASN1INT *value*, ASN1INT *compValue*)

Parameters:

name The name of value.
value The first value to compare.
compValue The second value to compare.

ASN1BOOL rtCmpToStdoutOctStr (ASN1ConstCharPtr *name*, ASN1UINT *numocts*, ASN1ConstOctetPtr *data*, ASN1UINT *compNumocts*, ASN1ConstOctetPtr *compData*)

Parameters:

name The name of value.
numocts The first value to compare.
data The pointer to the data of the first value.
compNumocts The second value to compare.
compData The pointer to the data of the second value.

ASN1BOOL rtCmpToStdoutOID (ASN1ConstCharPtr *name*, ASN1OBJID * *pOID*, ASN1OBJID * *pcompOID*)

Parameters:

name The name of value.
pOID The first value to compare.
pcompOID The second value to compare.

ASN1BOOL rtCmpToStdoutOID64 (ASN1ConstCharPtr *name*, ASN1OID64 * *pOID*, ASN1OID64 * *pcompOID*)

Parameters:

name The name of value.
pOID The first value to compare.
pcompOID The second value to compare.

ASN1BOOL rtCmpToStdoutOID64Value (ASN1ConstCharPtr *name*, ASN1OID64 * *pOID*, ASN1OID64 * *pcompOID*)

Parameters:

name The name of value.
pOID The first value to compare.
pcompOID The second value to compare.

ASN1BOOL rtCmpToStdoutOIDValue (ASN1ConstCharPtr *name*, ASN1OBJID * *pOID*, ASN1OBJID * *pcompOID*)

Parameters:

name The name of value.
pOID The first value to compare.
pcompOID The second value to compare.

ASN1BOOL rtCmpToStdoutOpenType (ASN1ConstCharPtr *name*, ASN1UINT *numocts*, ASN1ConstOctetPtr *data*, ASN1UINT *compNumocts*, ASN1ConstOctetPtr *compData*)

Parameters:

name The name of value.
numocts The number of octets in the first value to compare.
data The pointer to the data in the first value to compare.
compNumocts The number of octets in the second value to compare.
compData The pointer to the data in the second value to compare.

ASN1BOOL rtCmpToStdoutOpenTypeExt (ASN1ConstCharPtr *name*, Asn1RTDList * *pElemList*, Asn1RTDList * *pCompElemList*)

Parameters:

name The name of value.
pElemList The first value to compare.
pCompElemList The second value to compare.

ASN1BOOL rtCmpToStdoutOptional (ASN1ConstCharPtr *name*, unsigned *presentBit*, unsigned *compPresentBit*)

Parameters:

name The name of value.
presentBit The first value to compare.
compPresentBit The second value to compare.

ASN1BOOL rtCmpToStdoutReal (ASN1ConstCharPtr *name*, ASN1REAL *value*, ASN1REAL *compValue*)

Parameters:

name The name of value.
value The first value to compare.
compValue The second value to compare.

ASN1BOOL rtCmpToStdoutSeqOfElements (ASN1ConstCharPtr *name*, int *noOfElems*, int *compNoOfElems*)

Parameters:

name The name of value.
noOfElems The first value to compare.
compNoOfElems The second value to compare.

ASN1BOOL rtCmpToStdoutTag (ASN1ConstCharPtr *name*, int *tag*, int *compTag*)

Parameters:

name The name of value.
tag The first value to compare.
compTag The second value to compare.

ASN1BOOL rtCmpToStdoutUInt64 (ASN1ConstCharPtr *name*, ASN1UINT64 *value*, ASN1UINT64 *compValue*)

Parameters:

name The name of value.
value The first value to compare.
compValue The second value to compare.

ASN1BOOL rtCmpToStdoutUnsigned (ASN1ConstCharPtr *name*, ASN1UINT *value*, ASN1UINT *compValue*)

Parameters:

name The name of value.
value The first value to compare.
compValue The second value to compare.

Copy Functions

Detailed Description

This group of functions allows copying values of primitive ASN.1 types.

These functions are used in copy routines that are generated by the ASN.1 compiler when *gencopy* option is used. Some primitive types that are mapped onto C standard primitive types (such as BOOLEAN, INTEGER REAL) do not need copy functions. The standard assignment operator can be used to copy these types.

Functions

- ASN1BOOL **rtCopyBitStr** (ASN1CTXT *pctx, ASN1UINT srcNumbits, ASN1ConstOctetPtr pSrcData, ASN1UINT *pDstNumbits, ASN1OCTET *pDstData)
- ASN1BOOL **rtCopyDynBitStr** (ASN1CTXT *pctx, ASN1DynBitStr *pSrcData, ASN1DynBitStr *pDstData)
- ASN1BOOL **rtCopyOctStr** (ASN1CTXT *pctx, ASN1UINT srcNumocts, ASN1ConstOctetPtr pSrcData, ASN1UINT *pDstNumocts, ASN1OCTET *pDstData)
- ASN1BOOL **rtCopyDynOctStr** (ASN1CTXT *pctx, ASN1DynOctStr *pSrcData, ASN1DynOctStr *pDstData)
- ASN1BOOL **rtCopyCharStr** (ASN1CTXT *pctx, ASN1ConstCharPtr srcStr, char **dstStr)
- ASN1BOOL **rtCopy16BitCharStr** (ASN1CTXT *pctx, Asn116BitCharString *srcStr, Asn116BitCharString *dstStr)
- ASN1BOOL **rtCopy32BitCharStr** (ASN1CTXT *pctx, Asn132BitCharString *srcStr, Asn132BitCharString *dstStr)
- ASN1BOOL **rtCopyOID** (ASN1CTXT *pctx, ASN1OBJID *srcOID, ASN1OBJID *dstOID)
- ASN1BOOL **rtCopyOID64** (ASN1CTXT *pctx, ASN1OID64 *srcOID, ASN1OID64 *dstOID)
- ASN1BOOL **rtCopyOpenType** (ASN1CTXT *pctx, ASN1OpenType *srcOT, ASN1OpenType *dstOT)
- ASN1BOOL **rtCopyOpenTypeExt** (ASN1CTXT *pctx, Asn1RTDList *srcList, Asn1RTDList *dstList)

Function Documentation

ASN1BOOL rtCopy16BitCharStr (ASN1CTXT * pctx, Asn116BitCharString * srcStr, Asn116BitCharString * dstStr)

The `rtCopy16BitCharStr` function copies one ASN.1 16-bit character string value to another (generally BMPString).

The return value is one of the TRUE (copied successfully) or FALSE (error has occurred).

Parameters:

pctx Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.

srcStr The pointer to the source 16-bit character string structure to copy.

dstStr The pointer to destination 16-bit character string structure to receive the copied string. The memory will be allocated dynamically via call to `rtMemAlloc` function.

Returns:

The copying result. TRUE, if success, otherwise FALSE.

ASN1BOOL rtCopy32BitCharStr (ASN1CTXT * *pctxt*, Asn132BitCharString * *srcStr*, Asn132BitCharString * *dstStr*)

The rtCopy32BitCharStr function copies one ASN.1 32-bit character string value to another (generally UniversalString).

The return value is one of the TRUE (copied successfully) or FALSE (error has occurred).

Parameters:

pctxt Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.

srcStr The pointer to the source 32-bit character string structure to copy.

dstStr The pointer to destination 32-bit character string structure to receive the copied string. The memory will be allocated dynamically via call to rtMemAlloc function.

Returns:

The copying result. TRUE, if success, otherwise FALSE.

ASN1BOOL rtCopyBitStr (ASN1CTXT * *pctxt*, ASN1UINT *srcNumbits*, ASN1ConstOctetPtr *pSrcData*, ASN1UINT * *pDstNumbits*, ASN1OCTET * *pDstData*)

The rtCopyBitStr function copies one ASN.1 BIT STRING value to another.

The return value is one of the TRUE (copied successfully) or FALSE (error has occurred).

Parameters:

pctxt Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.

srcNumbits The number of bits in the source value to copy.

pSrcData The pointer to data of the source value to copy.

pDstNumbits The pointer to destination number of bits. The srcNumbits argument will be copied into it.

pDstData The pointer to the destination buffer to receive the copied data. The buffer is assumed to be already allocated or static and should be enough to receive the copying data.

Returns:

The copying result. TRUE, if success, otherwise FALSE.

ASN1BOOL rtCopyCharStr (ASN1CTXT * *pctxt*, ASN1ConstCharPtr *srcStr*, char ** *dstStr*)

The rtCopyCharStr function copies one ASN.1 8-bit character string value to another (including IA5String, VisibleString, PrintableString, NumericString, etc).

The return value is one of the TRUE (copied successfully) or FALSE (error has occurred).

Parameters:

pctxt Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.

srcStr The pointer to the source standard null-terminated string to copy.

dstStr The pointer to pointer destination string to receive the copied string. The memory will be allocated dynamically via a call to rtMemAlloc function.

Returns:

The copying result. TRUE, if success, otherwise FALSE.

ASN1BOOL rtCopyDynBitStr (ASN1CTXT * *pctxt*, ASN1DynBitStr * *pSrcData*, ASN1DynBitStr * *pDstData*)

The rtCopyDynBitStr function is similar to the rtCopyBitStr, but it copies a dynamic ASN.1 BIT STRING value.

The return value is one of the TRUE (copied successfully) or FALSE (error has occurred).

Parameters:

pctxt Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.

pSrcData The pointer to data of the source value to copy.

pDstData The pointer to the destination dynamic bit string structure to receive the copied data. The memory will be allocated dynamically via call to rtMemAlloc function.

Returns:

The copying result. TRUE, if success, otherwise FALSE.

ASN1BOOL rtCopyDynOctStr (ASN1CTXT * *pctxt*, ASN1DynOctStr * *pSrcData*, ASN1DynOctStr * *pDstData*)

The rtCopyDynOctStr function is similar to rtCopyOctStr, but it copies a dynamic ASN.1 OCTET STRING value.

The return value is one of the TRUE (copied successfully) or FALSE (error has occurred).

Parameters:

pctxt Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.

pSrcData The pointer to the source dynamic octet string structure to copy.

pDstData The pointer to destination dynamic octet string structure to receive the copied data. The memory will be allocated dynamically via a call to rtMemAlloc function.

Returns:

The copying result. TRUE, if success, otherwise FALSE.

ASN1BOOL rtCopyOctStr (ASN1CTXT * *pctxt*, ASN1UINT *srcNumocts*, ASN1ConstOctetPtr *pSrcData*, ASN1UINT * *pDstNumocts*, ASN1OCTET * *pDstData*)

The rtCopyOctStr function copies one ASN.1 OCTET STRING value to another.

The return value is one of the TRUE (copied successfully) or FALSE (error has occurred).

Parameters:

pctxt Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.

srcNumocts The number of octets in the source value to copy.

pSrcData The pointer to data of the source value to copy.

pDstNumocts The pointer to the destination number of octets. The srcNumocts argument will be copied into it.

pDstData The pointer to the destination buffer to receive the copied data. The buffer is assumed to be already allocated or static and should be enough to receive the copying data.

Returns:

The copying result. TRUE, if success, otherwise FALSE.

ASN1BOOL rtCopyOID (ASN1CTXT * *pctxt*, ASN1OBJID * *srcOID*, ASN1OBJID * *dstOID*)

The rtCopyOID function copies one ASN.1 OBJECT IDENTIFIER or RELATED-IOD value to another.

The return value is one of the TRUE (copied successfully) or FALSE (error has occurred).

Parameters:

pctxt Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
srcOID The pointer to the source object identifier structure to copy.
dstOID The pointer to destination structure to receive the copied string.

Returns:

The copying result. TRUE, if success, otherwise FALSE.

ASN1BOOL rtCopyOID64 (ASN1CTXT * *pctxt*, ASN1OID64 * *srcOID*, ASN1OID64 * *dstOID*)

The rtCopyOID64 function copies one 64-bit ASN.1 OBJECT IDENTIFIER or RELATIVE-OID value to another.

The return value is one of the TRUE (copied successfully) or FALSE (error has occurred).

Parameters:

pctxt Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
srcOID The pointer to the source object identifier structure to copy.
dstOID The pointer to destination structure to receive the copied string.

Returns:

The copying result. TRUE, if success, otherwise FALSE.

ASN1BOOL rtCopyOpenType (ASN1CTXT * *pctxt*, ASN1OpenType * *srcOT*, ASN1OpenType * *dstOT*)

The rtCopyOpenType copies ASN.1 value of the old (pre- 1994) ASN.1 ANY type or other elements defined in the later standards to be Open Types (for example, a variable type declaration in a CLASS construct as defined in X.681).

The return value is one of the TRUE (copied successfully) or FALSE (error has occurred).

Parameters:

pctxt Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
srcOT The pointer to the source Open Type structure to copy.
dstOT The pointer to the destination Open Type structure to receive the copied data. The memory will be allocated dynamically via call to the rtMemAlloc function.

Returns:

The copying result. TRUE, if success, otherwise FALSE.

ASN1BOOL rtCopyOpenTypeExt (ASN1CTXT * *pctxt*, Asn1RTDList * *srcList*, Asn1RTDList * *dstList*)

The rtCopyOpenTypeExt function copies an ASN.1 open type extension value.

The return value is one of the TRUE (copied successfully) or FALSE (error has occurred). An open type extension is defined as extensibility marker on a constructed type without any extension elements defined (for example, SEQUENCE { a INTEGER, ... }). The difference is that this is an implicit field that can span more elements whereas the standard Open Type is assumed to be a single tagged field.

Parameters:

pctxt Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
srcList The pointer to the source linked list structure to copy. The list should consist of ASN1OpenType elements.

dstList The pointer to destination linked list structure to receive the copied data. The memory for list nodes and data will be allocated dynamically via call to the `rtMemAlloc` function. The list's nodes will contain the data of `ASN1OpenType` type.

Print Values to Standard Output

Detailed Description

These functions print the output in a "name=value" format. The value format is obtained by calling one of the `Tostring` functions with the given value.

Functions

- void **rtPrintBoolean** (const char *name, ASN1BOOL value)
 - void **rtPrintInteger** (const char *name, ASN1INT value)
 - void **rtPrintInt64** (const char *name, ASN1INT64 value)
 - void **rtPrintUnsigned** (const char *name, ASN1UINT value)
 - void **rtPrintUInt64** (const char *name, ASN1UINT64 value)
 - void **rtPrintBitStr** (const char *name, ASN1UINT numbits, ASN1ConstOctetPtr data, const char *conn)
 - void **rtPrintBitStrBraceText** (const char *name, OSUINT32 numbits, const OSOCTET *data)
 - void **rtPrintOctStr** (const char *name, ASN1UINT numocts, ASN1ConstOctetPtr data, const char *conn)
 - void **rtPrintHexStr** (const char *name, OSUINT32 numocts, const OSOCTET *data)
 - void **rtPrintCharStr** (const char *name, const char *cstring)
 - void **rtPrint16BitCharStr** (const char *name, Asn116BitCharString *bstring, const char *conn)
 - void **rtPrintUnicodeCharStr** (const char *name, Asn116BitCharString *bstring)
 - void **rtPrint32BitCharStr** (const char *name, Asn132BitCharString *bstring, const char *conn)
 - void **rtPrintUnivCharStr** (const char *name, Asn132BitCharString *bstring)
 - void **rtPrintReal** (const char *name, ASN1REAL value)
 - void **rtPrintOID** (const char *name, ASN1OBJID *pOID)
 - void **rtPrintOIDValue** (ASN1OBJID *pOID)
 - void **rtPrintOID64** (const char *name, ASN1OID64 *pOID)
 - void **rtPrintOID64Value** (ASN1OID64 *pOID)
 - void **rtPrintOpenType** (const char *name, ASN1UINT numocts, ASN1ConstOctetPtr data, const char *conn)
 - void **rtPrintOpenTypeExt** (const char *name, Asn1RTDList *pElemList)
 - void **rtPrintOpenTypeExtBraceText** (const char *name, Asn1RTDList *pElemList)
 - void **rtPrintIndent** ()
 - void **rtPrintIncrIndent** ()
 - void **rtPrintDecrIndent** ()
 - void **rtPrintCloseBrace** ()
 - void **rtPrintOpenBrace** (const char *)
-

Function Documentation

void rtPrint16BitCharStr (const char * name, Asn116BitCharString * bstring, const char * conn)

This function prints the value of a 16-bit character string to stdout.

Parameters:

name The name of the variable to print.

bstring A pointer to the bit string to be printed.

conn A pointer to the connector between the name and the value. This points to either -> or .

void rtPrint32BitCharStr (const char * name, Asn132BitCharString * bstring, const char * conn)

This function prints the value of a 32-bit character string to stdout.

Parameters:

name The name of the variable to print.

bstring A pointer to the bit string to be printed.

conn A pointer to the connector between the name and the value. This points to either -> or .

void rtPrintBitStr (const char * name, ASN1UINT numbits, ASN1ConstOctetPtr data, const char * conn)

This function prints the value of a bit string to stdout.

Parameters:

name The name of the variable to print.

numbits The number of bits to be printed.

data A pointer to the data to be printed.

conn A pointer to the connector between the name and the value. This points to either -> or .

void rtPrintBitStrBraceText (const char * name, OSUINT32 numbits, const OSOCTET * data)

This function prints the value of a bit string to stdout in brace text format.

Parameters:

name The name of the variable to print.

numbits The number of bits to be printed.

data A pointer to the data to be printed.

void rtPrintBoolean (const char * name, ASN1BOOL value)

This function prints the value of a boolean type to stdout.

Parameters:

name The name of the variable to print.

value ASN.1 value to print (note: multiple arguments may be used to represent the value - for example a bit string would be represented by a numbits argument).

void rtPrintCharStr (const char * name, const char * cstring)

This function prints the value of a character string to stdout.

Parameters:

name The name of the variable to print.

cstring A pointer to the character string to be printed.

void rtPrintCloseBrace ()

This function closes a braced region by decreasing the indent level, printing indent spaces, and printing the closing brace.

void rtPrintDecrIndent ()

This function decrements the current indentation level.

void rtPrintHexStr (const char * name, OSUINT32 numocts, const OSOCTET * data)

This function prints the value of a binary string in hex format to standard output. If the string is 32 bytes or less, it is printed on a single line with a '0x' prefix. If longer, a formatted hex dump showing both hex and ascii codes is done.

Parameters:

name The name of the variable to print.

numocts The number of octets to be printed.

data A pointer to the data to be printed.

void rtPrintIncrIndent ()

This function increments the current indentation level.

void rtPrintIndent ()

This function prints indentation spaces to stdout.

void rtPrintInt64 (const char * name, ASN1INT64 value)

This function prints the value of a 64-bit signed integer to stdout.

Parameters:

name The name of the variable to print.

value ASN.1 value to print (note: multiple arguments may be used to represent the value - for example a bit string would be represented by a numbits argument).

void rtPrintInteger (const char * name, ASN1INT value)

This function prints the value of a 32-bit signed integer to stdout.

Parameters:

name The name of the variable to print.

value ASN.1 value to print (note: multiple arguments may be used to represent the value - for example a bit string would be represented by a numbits argument).

void rtPrintOctStr (const char * name, ASN1UINT numocts, ASN1ConstOctetPtr data, const char * conn)

This function prints the value of an octet string to stdout.

Parameters:

name The name of the variable to print.

numocts The number of octets to be printed.

data A pointer to the data to be printed.

conn A pointer to the connector between the name and the value. This points to either -> or .

void rtPrintOID (const char * name, ASN1OBJID * pOID)

This function prints the value of an object identifier to stdout.

Parameters:

name The name of the variable to print.
pOID A pointer to the OID to be printed.

void rtPrintOID64 (const char * name, ASN1OID64 * pOID)

This function prints the value of an object identifier with 64-bit arc values to stdout.

Parameters:

name The name of the variable to print.
pOID A pointer to the OID to be printed.

void rtPrintOID64Value (ASN1OID64 * pOID)

This function prints the value of an object identifier with 64-bit arc values to stdout. Only the value is printed, not the name.

Parameters:

pOID A pointer to the OID to be printed.

void rtPrintOIDValue (ASN1OBJID * pOID)

This function prints the value of an object identifier to stdout. Only the value is printed, not the name.

Parameters:

pOID A pointer to the OID to be printed.

void rtPrintOpenBrace (const char *)

This function opens a braced region by printing indent spaces, printing the name and opening brace, and increasing the indent level.

void rtPrintOpenType (const char * name, ASN1UINT numocts, ASN1ConstOctetPtr data, const char * conn)

This function prints the value of an open type to stdout.

Parameters:

name The name of the variable to print.
numocts The number of octets to be printed.
data The pointer to the data to be printed.
conn A pointer to the connector between the name and the value. This points to either -> or .

void rtPrintOpenTypeExt (const char * name, Asn1RTDList * pElemList)

This function prints the value of an open type extension to stdout.

Parameters:

name The name of the variable to print.
pElemList A pointer to an element of a list.

void rtPrintOpenTypeExtBraceText (const char * name, Asn1RTDList * pElemList)

This function prints the value of an open type extension in brace text format to stdout.

Parameters:

name The name of the variable to print.
pElemList A pointer to an element of a list.

void rtPrintReal (const char * *name*, ASN1REAL *value*)

This function prints the value of a floating-point number to stdout.

Parameters:

name The name of the variable to print.
value ASN.1 value to print

void rtPrintUInt64 (const char * *name*, ASN1UINT64 *value*)

This function prints the value of a 64-bit unsigned integer to stdout.

Parameters:

name The name of the variable to print.
value ASN.1 value to print (note: multiple arguments may be used to represent the value - for example a bit string would be represented by a numbits argument).

void rtPrintUnicodeCharStr (const char * *name*, Asn116BitCharString * *bstring*)

This function prints a Unicode string to standard output. Characters in the string that are within the normal Ascii range are printed as single characters. Characters outside the Ascii range are printed as 4-byte hex codes (0xnxxx).

Parameters:

name The name of the variable to print.
bstring A pointer to the bit string to be printed.

void rtPrintUnivCharStr (const char * *name*, Asn132BitCharString * *bstring*)

This function prints a Universal string to standard output. Characters in the string that are within the normal Ascii range are printed as single characters. Characters outside the Ascii range are printed as 8-byte hex codes (0xxxxxxxx).

Parameters:

name The name of the variable to print.
bstring A pointer to the bit string to be printed.

void rtPrintUnsigned (const char * *name*, ASN1UINT *value*)

This function prints the value of a 32-bit unsigned integer to stdout.

Parameters:

name The name of the variable to print.
value ASN.1 value to print (note: multiple arguments may be used to represent the value - for example a bit string would be represented by a numbits argument).

Print stream structure and function definitions.

Classes

- struct **OSRTPrintStream**

Typedefs

- typedef void(* **rtPrintCallback**)(void *pPrntStrmInfo, ASN1ConstCharPtr fmtsSpec, va_list arglist)
- typedef **OSRTPrintStream OSRTPrintStream**

Functions

- `int rtSetPrintStream (ASN1CTXT *pctx, rtPrintCallback myCallback, void *pStrmInfo)`
- `int rtSetGlobalPrintStream (rtPrintCallback myCallback, void *pStrmInfo)`
- `int rtPrintToStream (ASN1CTXT *pctx, ASN1ConstCharPtr fmtspec,...)`
- `int rtDiagToStream (ASN1CTXT *pctx, ASN1ConstCharPtr fmtspec, va_list arglist)`
- `int rtPrintStreamRelease (ASN1CTXT *pctx)`

Variables

- `OSRTPrintStream g_PrintStream`

Typedef Documentation

`typedef struct OSRTPrintStream OSRTPrintStream`

Structure to hold information about global PrintStream

Function Documentation

`int rtDiagToStream (ASN1CTXT * pctx, ASN1ConstCharPtr fmtspec, va_list arglist)`

Print to stream function which in turn calls the user registered callback function of the context for printing. If no callback function is registered it prints to standard output by default.

Parameters:

pctx Pointer to an ASN1 context used.

fmtspec A printf-like format specification string describing the message to be printed (for example, "string s, ivalue d").

arglist A variable list of arguments passed as *va_list*

Returns:

Completion status, 0 on success, -ve on failure

`int rtPrintStreamRelease (ASN1CTXT * pctx)`

This function releases the memory held by PrintStream in the context

Parameters:

pctx Pointer to a ASN1CONTEXT for which the memory has to be released.

Returns:

Completion status, 0 on success, -ve on failure

`int rtPrintToStream (ASN1CTXT * pctx, ASN1ConstCharPtr fmtspec, ...)`

Print to stream function which in turn calls the user registered callback function of the context for printing. If no callback function is registered it prints to standard output by default.

Parameters:

pctx Pointer to an ASN1 context used.

fmtspec A printf-like format specification string describing the message to be printed (for example, "string s, ivalue d").

... A variable list of arguments.

Returns:

Completion status, 0 on success, -ve on failure

int rtSetGlobalPrintStream (rtPrintCallback *myCallback*, void * *pStrmInfo*)

This function is for setting the callback function for PrintStream. Once a callback function is set, then all the print and debug outputs are sent to that function.

Parameters:

myCallback Pointer to a callback print function.

pStrmInfo Pointer to user defined PrintInfo structure where users can store information required by the callback function across calls. Ex. An open File handle for callbak function which directs stream to a file.

Returns:

Completion status, 0 on success, -ve on failure

int rtSetPrintStream (ASN1CTXT * *pctxt*, rtPrintCallback *myCallback*, void * *pStrmInfo*)

This function is for setting the callback function for PrintStream. Once a callback function is set, then all the print and debug outputs are sent to that function.

Parameters:

pctxt Pointer to a context for which callback print function has to be set

myCallback Pointer to a callback print function.

pStrmInfo Pointer to user defined PrintInfo structure where users can store information required by the callback function across calls. Ex. An open File handle for callbak function which directs stream to a file.

Returns:

Completion status, 0 on success, -ve on failure

Variable Documentation**OSRTPrintStream g_PrintStream**

Global PrintStream

Print Values to user specified Stream**Detailed Description**

These functions print the output in a "name=value" format.

Functions

- int **rtPrintToStreamBoolean** (ASN1CTXT **pctxt*, ASN1ConstCharPtr name, ASN1BOOL value)
- int **rtPrintToStreamInteger** (ASN1CTXT **pctxt*, ASN1ConstCharPtr name, ASN1INT value)
- int **rtPrintToStreamInt64** (ASN1CTXT **pctxt*, ASN1ConstCharPtr name, ASN1INT64 value)
- int **rtPrintToStreamUnsigned** (ASN1CTXT **pctxt*, ASN1ConstCharPtr name, ASN1UINT value)
- int **rtPrintToStreamUInt64** (ASN1CTXT **pctxt*, ASN1ConstCharPtr name, ASN1UINT64 value)

- int **rtPrintToStreamBitStr** (ASN1CTXT *pctxt, ASN1ConstCharPtr name, ASN1UINT numbits, ASN1ConstOctetPtr data, ASN1ConstCharPtr conn)
- int **rtPrintToStreamBitStrBraceText** (ASN1CTXT *pctxt, const char *name, OSUINT32 numbits, const OSOCTET *data)
- int **rtPrintToStreamOctStr** (ASN1CTXT *pctxt, ASN1ConstCharPtr name, ASN1UINT numocts, ASN1ConstOctetPtr data, ASN1ConstCharPtr conn)
- int **rtPrintToStreamCharStr** (ASN1CTXT *pctxt, ASN1ConstCharPtr name, ASN1ConstCharPtr cstring)
- int **rtPrintToStream16BitCharStr** (ASN1CTXT *pctxt, ASN1ConstCharPtr name, Asn116BitCharString *bstring, ASN1ConstCharPtr conn)
- int **rtPrintToStream32BitCharStr** (ASN1CTXT *pctxt, ASN1ConstCharPtr name, Asn132BitCharString *bstring, ASN1ConstCharPtr conn)
- int **rtPrintToStreamReal** (ASN1CTXT *pctxt, ASN1ConstCharPtr name, ASN1REAL value)
- int **rtPrintToStreamOID** (ASN1CTXT *pctxt, ASN1ConstCharPtr name, ASN1OBJID *pOID)
- int **rtPrintToStreamOIDValue** (ASN1CTXT *pctxt, ASN1OBJID *pOID)
- int **rtPrintToStreamOID64** (ASN1CTXT *pctxt, ASN1ConstCharPtr name, ASN1OID64 *pOID)
- int **rtPrintToStreamOID64Value** (ASN1CTXT *pctxt, ASN1OID64 *pOID)
- int **rtPrintToStreamOpenType** (ASN1CTXT *pctxt, ASN1ConstCharPtr name, ASN1UINT numocts, ASN1ConstOctetPtr data, ASN1ConstCharPtr conn)
- int **rtPrintToStreamOpenTypeExt** (ASN1CTXT *pctxt, ASN1ConstCharPtr name, Asn1RTDList *pElemList)
- int **rtPrintToStreamUnivCharStr** (ASN1CTXT *pctxt, const char *name, Asn132BitCharString *bstring)
- int **rtPrintToStreamUnicodeCharStr** (ASN1CTXT *pctxt, const char *name, Asn116BitCharString *bstring)
- int **rtPrintToStreamHexStr** (ASN1CTXT *pctxt, const char *name, OSUINT32 numocts, const OSOCTET *data)
- int **rtPrintToStreamOpenTypeExtBraceText** (ASN1CTXT *pctxt, const char *name, Asn1RTDList *pElemList)
- int **rtPrintToStreamIndent** (ASN1CTXT *pctxt)
- void **rtPrintToStreamIncrIndent** ()
- void **rtPrintToStreamDecrIndent** ()
- int **rtPrintToStreamCloseBrace** (ASN1CTXT *pctxt)
- int **rtPrintToStreamOpenBrace** (ASN1CTXT *pctxt, const char *name)

Function Documentation

int rtPrintToStream16BitCharStr (ASN1CTXT * pctxt, ASN1ConstCharPtr name, Asn116BitCharString * bstring, ASN1ConstCharPtr conn)

Parameters:

pctxt Pointer to an ASN1 context initialized for printing.

name The name of the variable to print.

bstring A pointer to the bit string to be printed.

conn A pointer to the connector between the name and the value. This points to either -> or .

Returns:

Completion status, 0 on success and -ve on failure

int rtPrintToStream32BitCharStr (ASN1CTXT * pctxt, ASN1ConstCharPtr name, Asn132BitCharString * bstring, ASN1ConstCharPtr conn)

Parameters:

pctxt Pointer to an ASN1 context initialized for printing.
name The name of the variable to print.
bstring A pointer to the bit string to be printed.
conn A pointer to the connector between the name and the value. This points to either -> or .

Returns:

Completion status, 0 on success and -ve on failure

int rtPrintToStreamBitStr (ASN1CTXT * *pctxt*, ASN1ConstCharPtr *name*, ASN1UINT *numbits*, ASN1ConstOctetPtr *data*, ASN1ConstCharPtr *conn*)

Parameters:

pctxt Pointer to an ASN1 context initialized for printing.
name The name of the variable to print.
numbits The number of bits to be printed.
data A pointer to the data to be printed.
conn A pointer to the connector between the name and the value. This points to either -> or .

Returns:

Completion status, 0 on success and -ve on failure

int rtPrintToStreamBitStrBraceText (ASN1CTXT * *pctxt*, const char * *name*, OSUINT32 *numbits*, const OSOCTET * *data*)

This function prints the value of a bit string to a stream in brace text format.

Parameters:

pctxt Pointer to a context initialized for printing.
name The name of the variable to print.
numbits The number of bits to be printed.
data A pointer to the data to be printed.

Returns:

Completion status, 0 on success and -ve on failure

int rtPrintToStreamBoolean (ASN1CTXT * *pctxt*, ASN1ConstCharPtr *name*, ASN1BOOL *value*)

Parameters:

pctxt Pointer to an ASN1 context initialized for printing.
name The name of the variable to print.
value ASN.1 value to print (note: multiple arguments may be used to represent the value - for example a bit string would be represented by a numbits argument).

Returns:

Completion status, 0 on success and -ve on failure

int rtPrintToStreamCharStr (ASN1CTXT * *pctxt*, ASN1ConstCharPtr *name*, ASN1ConstCharPtr *cstring*)

Parameters:

pctxt Pointer to an ASN1 context initialized for printing.
name The name of the variable to print.

cstring A pointer to the character string to be printed.

Returns:

Completion status, 0 on success and -ve on failure

int rtPrintToStreamCloseBrace (ASN1CTXT * *pctxt*)

This function closes a braced region by decreasing the indent level, printing indent spaces, and printing the closing brace.

Parameters:

pctxt Pointer to an ASN1 context initialized for stream printing.

void rtPrintToStreamDecrIndent ()

This function decrements the current indentation level.

int rtPrintToStreamHexStr (ASN1CTXT * *pctxt*, const char * *name*, OSUINT32 *numocts*, const OSOCTET * *data*)

This function prints the value of a binary string in hex format to stream. If the string is 32 bytes or less, it is printed on a single line with a '0x' prefix. If longer, a formatted hex dump showing both hex and ascii codes is done.

Parameters:

pctxt Pointer to an ASN1 context initialized for stream printing.

name The name of the variable to print.

numocts The number of octets to be printed.

data A pointer to the data to be printed.

void rtPrintToStreamIncrIndent ()

This function increments the current indentation level.

int rtPrintToStreamIndent (ASN1CTXT * *pctxt*)

This function prints indentation spaces to stream.

Parameters:

pctxt Pointer to an ASN1 context initialized for stream printing.

int rtPrintToStreamInt64 (ASN1CTXT * *pctxt*, ASN1ConstCharPtr *name*, ASN1INT64 *value*)

Parameters:

pctxt Pointer to an ASN1 context initialized for printing.

name The name of the variable to print.

value ASN.1 value to print (note: multiple arguments may be used to represent the value - for example a bit string would be represented by a numbits argument).

Returns:

Completion status, 0 on success and -ve on failure

int rtPrintToStreamInteger (ASN1CTXT * *pctxt*, ASN1ConstCharPtr *name*, ASN1INT *value*)

Parameters:

pctxt Pointer to an ASN1 context initialized for printing.

name The name of the variable to print.
value ASN.1 value to print (note: multiple arguments may be used to represent the value - for example a bit string would be represented by a numbits argument).

Returns:

Completion status, 0 on success and -ve on failure

int rtPrintToStreamOctStr (ASN1CTXT * *pctxt*, ASN1ConstCharPtr *name*, ASN1UINT *numocts*, ASN1ConstOctetPtr *data*, ASN1ConstCharPtr *conn*)

Parameters:

pctxt Pointer to an ASN1 context initialized for printing.
name The name of the variable to print.
numocts The number of octets to be printed.
data A pointer to the data to be printed.
conn A pointer to the connector between the name and the value. This points to either -> or .

Returns:

Completion status, 0 on success and -ve on failure

int rtPrintToStreamOID (ASN1CTXT * *pctxt*, ASN1ConstCharPtr *name*, ASN1OBJID * *pOID*)

Parameters:

pctxt Pointer to an ASN1 context initialized for printing.
name The name of the variable to print.
pOID A pointer to the OID to be printed.

Returns:

Completion status, 0 on success and -ve on failure

int rtPrintToStreamOID64 (ASN1CTXT * *pctxt*, ASN1ConstCharPtr *name*, ASN1OID64 * *pOID*)

Parameters:

pctxt Pointer to an ASN1 context initialized for printing.
name The name of the variable to print.
pOID A pointer to the OID to be printed.

Returns:

Completion status, 0 on success and -ve on failure

int rtPrintToStreamOID64Value (ASN1CTXT * *pctxt*, ASN1OID64 * *pOID*)

Parameters:

pctxt Pointer to an ASN1 context initialized for printing.
pOID A pointer to the OID to be printed.

Returns:

Completion status, 0 on success and -ve on failure

int rtPrintToStreamOIDValue (ASN1CTXT * *pctxt*, ASN1OBJID * *pOID*)

Parameters:

pctxt Pointer to an ASN1 context initialized for printing.
pOID A pointer to the OID to be printed.

Returns:

Completion status, 0 on success and -ve on failure

int rtPrintToStreamOpenBrace (ASN1CTXT * *pctxt*, const char * *name*)

This function opens a braced region by printing indent spaces, printing the name and opening brace, and increasing the indent level.

Parameters:

pctxt Pointer to an ASN1 context initialized for stream printing.
name Name to print before opening brace.

int rtPrintToStreamOpenType (ASN1CTXT * *pctxt*, ASN1ConstCharPtr *name*, ASN1UINT *numocts*, ASN1ConstOctetPtr *data*, ASN1ConstCharPtr *conn*)

Parameters:

pctxt Pointer to an ASN1 context initialized for printing.
name The name of the variable to print.
numocts The number of octets to be printed.
data The pointer to the data to be printed.
conn A pointer to the connector between the name and the value. This points to either -> or .

Returns:

Completion status, 0 on success and -ve on failure

int rtPrintToStreamOpenTypeExt (ASN1CTXT * *pctxt*, ASN1ConstCharPtr *name*, Asn1RTDList * *pElemList*)

Parameters:

pctxt Pointer to an ASN1 context initialized for printing.
name The name of the variable to print.
pElemList A pointer to an element of a list.

Returns:

Completion status, 0 on success and -ve on failure

int rtPrintToStreamOpenTypeExtBraceText (ASN1CTXT * *pctxt*, const char * *name*, Asn1RTDList * *pElemList*)

This function prints the value of an open type extension in brace text format to stream.

Parameters:

pctxt Pointer to an ASN1 context initialized for stream printing.
name The name of the variable to print.
pElemList A pointer to an element of a list.

int rtPrintToStreamReal (ASN1CTXT * *pctxt*, ASN1ConstCharPtr *name*, ASN1REAL *value*)

Parameters:

pctxt Pointer to an ASN1 context initialized for printing.

name The name of the variable to print.
value ASN.1 value to print (note: multiple arguments may be used to represent the value - for example a bit string would be represented by a numbits argument).

Returns:

Completion status, 0 on success and -ve on failure

int rtPrintToStreamUInt64 (ASN1CTXT * *pctxt*, ASN1ConstCharPtr *name*, ASN1UINT64 *value*)

Parameters:

pctxt Pointer to an ASN1 context initialized for printing.
name The name of the variable to print.
value ASN.1 value to print (note: multiple arguments may be used to represent the value - for example a bit string would be represented by a numbits argument).

Returns:

Completion status, 0 on success and -ve on failure.

int rtPrintToStreamUnicodeCharStr (ASN1CTXT * *pctxt*, const char * *name*, Asn116BitCharString * *bstring*)

This function prints a Unicode string to stream output. Characters in the string that are within the normal Ascii range are printed as single characters. Characters outside the Ascii range are printed as 4-byte hex codes (0xnxxxx).

Parameters:

pctxt Pointer to an ASN1 context initialized for stream printing.
name The name of the variable to print.
bstring A pointer to the bit string to be printed.

int rtPrintToStreamUnivCharStr (ASN1CTXT * *pctxt*, const char * *name*, Asn132BitCharString * *bstring*)

This function prints a Universal string to stream output. Characters in the string that are within the normal Ascii range are printed as single characters. Characters outside the Ascii range are printed as 8-byte hex codes (0xxxxxxxx).

Parameters:

pctxt Pointer to an ASN1 context initialized for stream printing.
name The name of the variable to print.
bstring A pointer to the bit string to be printed.

int rtPrintToStreamUnsigned (ASN1CTXT * *pctxt*, ASN1ConstCharPtr *name*, ASN1UINT *value*)

Parameters:

pctxt Pointer to an ASN1 context initialized for printing.
name The name of the variable to print.
value ASN.1 value to print (note: multiple arguments may be used to represent the value - for example a bit string would be represented by a numbits argument).

Returns:

Completion status, 0 on success and -ve on failure

Print values to text buffer functions.

Detailed Description

Format the output value to string in a "name = value" format. The value format is obtained by calling one of the "ToString" functions with the given value.

Functions

- **int rtPrintToStringBoolean** (ASN1ConstCharPtr name, ASN1BOOL value, char *buffer, int bufferSize)
- **int rtPrintToStringInteger** (ASN1ConstCharPtr name, ASN1INT value, char *buffer, int bufferSize)
- **int rtPrintToStringInt64** (ASN1ConstCharPtr name, ASN1INT64 value, char *buffer, int bufferSize)
- **int rtPrintToStringUnsigned** (ASN1ConstCharPtr name, ASN1UINT value, char *buffer, int bufferSize)
- **int rtPrintToStringUInt64** (ASN1ConstCharPtr name, ASN1UINT64 value, char *buffer, int bufferSize)
- **int rtPrintToStringBitStr** (ASN1ConstCharPtr name, ASN1UINT numbits, ASN1ConstOctetPtr data, ASN1ConstCharPtr conn, char *buffer, int bufferSize)
- **int rtPrintToStringBitStrBraceText** (const char *name, OSUINT32 numbits, const OSOCTET *data, char *buffer, int bufferSize)
- **int rtPrintToStringOctStr** (ASN1ConstCharPtr name, ASN1UINT numocts, ASN1ConstOctetPtr data, ASN1ConstCharPtr conn, char *buffer, int bufferSize)
- **int rtPrintToStringCharStr** (ASN1ConstCharPtr name, ASN1ConstCharPtr cstring, char *buffer, int bufferSize)
- **int rtPrintToString16BitCharStr** (ASN1ConstCharPtr name, Asn116BitCharString *bstring, ASN1ConstCharPtr conn, char *buffer, int bufferSize)
- **int rtPrintToString32BitCharStr** (ASN1ConstCharPtr name, Asn132BitCharString *bstring, ASN1ConstCharPtr conn, char *buffer, int bufferSize)
- **int rtPrintToStringReal** (ASN1ConstCharPtr name, ASN1REAL value, char *buffer, int bufferSize)
- **int rtPrintToStringOID** (ASN1ConstCharPtr name, ASN1OBJID *pOID, char *buffer, int bufferSize)
- **int rtPrintToStringOID64** (ASN1ConstCharPtr name, ASN1OID64 *pOID, char *buffer, int bufferSize)
- **int rtPrintToStringOIDValue** (ASN1OBJID *pOID, char *buffer, int bufferIndex, int bufferSize)
- **int rtPrintToStringOID64Value** (ASN1OID64 *pOID, char *buffer, int bufferIndex, int bufferSize)
- **int rtPrintToStringOpenType** (ASN1ConstCharPtr name, ASN1UINT numocts, ASN1ConstOctetPtr data, ASN1ConstCharPtr conn, char *buffer, int bufferSize)
- **int rtPrintToStringOpenTypeExt** (ASN1ConstCharPtr name, Asn1RTDList *pElemList, char *buffer, int bufferSize)
- **int rtPrintToString** (ASN1ConstCharPtr namebuf, char *buffer, int bufSize)
- **int rtPrintToStringHexStr** (const char *name, OSUINT32 numocts, const OSOCTET *data, char *buffer, int bufSize)
- **int rtPrintToStringUnicodeCharStr** (const char *name, Asn116BitCharString *bstring, char *buffer, int bufSize)
- **int rtPrintToStringUnivCharStr** (const char *name, Asn132BitCharString *bstring, char *buffer, int bufSize)
- **int rtPrintToStringOpenTypeExtBraceText** (const char *name, Asn1RTDList *pElemList, char *buffer, int bufSize)
- **int rtPrintToStringIndent** (char *buffer, int bufSize)

- void **rtPrintToStringIncrIndent** ()
 - void **rtPrintToStringDecrIndent** ()
 - int **rtPrintToStringCloseBrace** (char *buffer, int bufSize)
 - int **rtPrintToStringOpenBrace** (const char *, char *buffer, int bufSize)
-

Function Documentation

int rtPrintToString (ASN1ConstCharPtr namebuf, char * buffer, int bufSize)

Parameters:

namebuf A pointer to the buffer name.
buffer Pointer to a buffer to receive the printed value.
bufSize The size of the buffer to receive the printed value.

int rtPrintToString16BitCharStr (ASN1ConstCharPtr name, Asn16BitCharString * bstring, ASN1ConstCharPtr conn, char * buffer, int bufferSize)

Parameters:

name The name of the variable to print.
bstring A pointer to a 16-bit string to print.
conn A pointer to the connector between the name and the value. This points to either -> or .
bufferSize The size of the buffer to receive the printed value.
buffer Pointer to a buffer to receive the printed value.

int rtPrintToString32BitCharStr (ASN1ConstCharPtr name, Asn32BitCharString * bstring, ASN1ConstCharPtr conn, char * buffer, int bufferSize)

Parameters:

name The name of the variable to print.
bstring A pointer to a 32-bit string to print.
conn A pointer to the connector between the name and the value. This points to either -> or .
bufferSize The size of the buffer to receive the printed value.
buffer Pointer to a buffer to receive the printed value.

int rtPrintToStringBitStr (ASN1ConstCharPtr name, ASN1UINT numbits, ASN1ConstOctetPtr data, ASN1ConstCharPtr conn, char * buffer, int bufferSize)

Parameters:

name The name of the variable to print.
numbits The number of bits to be printed.
data A pointer to the data to be printed.
conn A pointer to the connector between the name and the value. This points to either -> or .
bufferSize The size of the buffer to receive the printed value.
buffer Pointer to a buffer to receive the printed value.

int rtPrintToStringBitStrBraceText (const char * name, OSUINT32 numbits, const OSOCTET * data, char * buffer, int bufferSize)

This function prints the value of a bit string to a text buffer in brace text format.

Parameters:

name The name of the variable to print.
numbits The number of bits to be printed.
data A pointer to the data to be printed.
bufferSize The size of the buffer to receive the printed value.
buffer Pointer to a buffer to receive the printed value.

int rtPrintToStringBoolean (ASN1ConstCharPtr *name*, ASN1BOOL *value*, char * *buffer*, int *bufferSize*)

Parameters:

name The name of the variable to print.
value ASN.1 value to print (Note: multiple arguments may be used to represent the value- for example a bit string would be represented by a numbits and data argument. See the function prototype for the exact calling sequence).
bufferSize The size of the buffer to receive the printed value.
buffer Pointer to a buffer to receive the printed value.

int rtPrintToStringCharStr (ASN1ConstCharPtr *name*, ASN1ConstCharPtr *cstring*, char * *buffer*, int *bufferSize*)

Parameters:

name The name of the variable to print.
cstring A pointer to the character string to be printed.
bufferSize The size of the buffer to receive the printed value.
buffer Pointer to a buffer to receive the printed value.

int rtPrintToStringCloseBrace (char * *buffer*, int *bufSize*)

This function closes a braced region by decreasing the indent level, printing indent spaces, and printing the closing brace.

Parameters:

buffer Pointer to a buffer to receive the printed value.
bufSize The size of the buffer to receive the printed value.

void rtPrintToStringDecrIndent ()

This function decrements the current indentation level.

int rtPrintToStringHexStr (const char * *name*, OSUINT32 *numocts*, const OSOCTET * *data*, char * *buffer*, int *bufSize*)

This function prints the value of a binary string in hex format to string buffer. If the string is 32 bytes or less, it is printed on a single line with a '0x' prefix. If longer, a formatted hex dump showing both hex and ascii codes is done.

Parameters:

name The name of the variable to print.
numocts The number of octets to be printed.
data A pointer to the data to be printed.
buffer Pointer to a buffer to receive the printed value.
bufSize The size of the buffer to receive the printed value.

void rtPrintToStringIncrIndent ()

This function increments the current indentation level.

int rtPrintToStringIndent (char * *buffer*, int *bufSize*)

This function prints indentation spaces to buffer.

Parameters:

buffer Pointer to a buffer to receive the printed value.

bufSize The size of the buffer to receive the printed value.

int rtPrintToStringInt64 (ASN1ConstCharPtr *name*, ASN1INT64 *value*, char * *buffer*, int *bufferSize*)

Parameters:

name The name of the variable to print.

value ASN.1 value to print (Note: multiple arguments may be used to represent the value- for example a bit string would be represented by a numbits and data argument. See the function prototype for the exact calling sequence).

bufferSize The size of the buffer to receive the printed value.

buffer Pointer to a buffer to receive the printed value.

int rtPrintToStringInteger (ASN1ConstCharPtr *name*, ASN1INT *value*, char * *buffer*, int *bufferSize*)

Parameters:

name The name of the variable to print.

value ASN.1 value to print (Note: multiple arguments may be used to represent the value- for example a bit string would be represented by a numbits and data argument. See the function prototype for the exact calling sequence).

bufferSize The size of the buffer to receive the printed value.

buffer Pointer to a buffer to receive the printed value.

int rtPrintToStringOctStr (ASN1ConstCharPtr *name*, ASN1UINT *numocts*, ASN1ConstOctetPtr *data*, ASN1ConstCharPtr *conn*, char * *buffer*, int *bufferSize*)

Parameters:

name The name of the variable to print.

numocts The number of octets to be printed.

data A pointer to the data to be printed.

conn A pointer to the connector between the name and the value. This points to either -> or .

bufferSize The size of the buffer to receive the printed value.

buffer Pointer to a buffer to receive the printed value.

int rtPrintToStringOID (ASN1ConstCharPtr *name*, ASN1OBJID * *pOID*, char * *buffer*, int *bufferSize*)

Parameters:

name The name of the variable to print.

pOID A pointer to a OID to be printed.

bufferSize The size of the buffer to receive the printed value.

buffer Pointer to a buffer to receive the printed value.

int rtPrintToStringOID64 (ASN1ConstCharPtr name, ASN1OID64 * pOID, char * buffer, int bufferSize)

Parameters:

name The name of the variable to print.

pOID A pointer to a OID to be printed.

bufferSize The size of the buffer to receive the printed value.

buffer Pointer to a buffer to receive the printed value.

int rtPrintToStringOID64Value (ASN1OID64 * pOID, char * buffer, int bufferIndex, int bufferSize)

Parameters:

bufferIndex The index to the buffer.

pOID A pointer to a OID to be printed.

bufferSize The size of the buffer to receive the printed value.

buffer Pointer to a buffer to receive the printed value.

int rtPrintToStringOIDValue (ASN1OBJID * pOID, char * buffer, int bufferIndex, int bufferSize)

Parameters:

bufferIndex The index to the buffer.

pOID A pointer to a OID to be printed.

bufferSize The size of the buffer to receive the printed value.

buffer Pointer to a buffer to receive the printed value.

int rtPrintToStringOpenBrace (const char *, char * buffer, int bufSize)

This function opens a braced region by printing indent spaces, printing the name and opening brace, and increasing the indent level.

Parameters:

buffer Pointer to a buffer to receive the printed value.

bufSize The size of the buffer to receive the printed value.

int rtPrintToStringOpenType (ASN1ConstCharPtr name, ASN1UINT numocts, ASN1ConstOctetPtr data, ASN1ConstCharPtr conn, char * buffer, int bufferSize)

Parameters:

name The name of the variable to print.

numocts The number of octets to be printed.

data A pointer to the data to be printed.

conn A pointer to the connector between the name and the value. This points to either -> or .

bufferSize The size of the buffer to receive the printed value.

buffer Pointer to a buffer to receive the printed value.

int rtPrintToStringOpenTypeExt (ASN1ConstCharPtr name, Asn1RTDList * pElemList, char * buffer, int bufferSize)

Parameters:

name The name of the variable to print.
pElemList A pointer to the element to be printed.
bufferSize The size of the buffer to receive the printed value.
buffer Pointer to a buffer to receive the printed value.

int rtPrintToStringOpenTypeExtBraceText (const char * name, Asn1RTDList * pElemList, char * buffer, int bufSize)

This function prints the value of an open type extension in brace text format to buffer.

Parameters:

name The name of the variable to print.
pElemList A pointer to an element of a list.
buffer Pointer to a buffer to receive the printed value.
bufSize The size of the buffer to receive the printed value.

int rtPrintToStringReal (ASN1ConstCharPtr name, ASN1REAL value, char * buffer, int bufferSize)

Parameters:

name The name of the variable to print.
value ASN.1 value to print (Note: multiple arguments may be used to represent the value- for example a bit string would be represented by a numbits and data argument. See the function prototype for the exact calling sequence).
bufferSize The size of the buffer to receive the printed value.
buffer Pointer to a buffer to receive the printed value.

int rtPrintToStringUInt64 (ASN1ConstCharPtr name, ASN1UINT64 value, char * buffer, int bufferSize)

Parameters:

name The name of the variable to print.
value ASN.1 value to print (Note: multiple arguments may be used to represent the value- for example a bit string would be represented by a numbits and data argument. See the function prototype for the exact calling sequence).
bufferSize The size of the buffer to receive the printed value.
buffer Pointer to a buffer to receive the printed value.

int rtPrintToStringUnicodeCharStr (const char * name, Asn116BitCharString * bstring, char * buffer, int bufSize)

This function prints a Unicode string to string buffer. Characters in the string that are within the normal Ascii range are printed as single characters. Characters outside the Ascii range are printed as 4-byte hex codes (0xnmmn).

Parameters:

name The name of the variable to print.
bstring A pointer to the bit string to be printed.
buffer Pointer to a buffer to receive the printed value.

bufSize The size of the buffer to receive the printed value.

int rtPrintToStringUnivCharStr (const char * *name*, Asn132BitCharString * *bstring*, char * *buffer*, int *bufSize*)

This function prints a Universal string to string buffer. Characters in the string that are within the normal Ascii range are printed as single characters. Characters outside the Ascii range are printed as 8-byte hex codes (0xnnnnnnnn).

Parameters:

name The name of the variable to print.
bstring A pointer to the bit string to be printed.
buffer Pointer to a buffer to receive the printed value.
bufSize The size of the buffer to receive the printed value.

int rtPrintToStringUnsigned (ASN1ConstCharPtr *name*, ASN1UINT *value*, char * *buffer*, int *bufferSize*)

Parameters:

name The name of the variable to print.
value ASN.1 value to print (Note: multiple arguments may be used to represent the value- for example a bit string would be represented by a numbits and data argument. See the function prototype for the exact calling sequence).
bufferSize The size of the buffer to receive the printed value.
buffer Pointer to a buffer to receive the printed value.

TCP/IP and UDP socket functions.

Functions

- int **rtSocketAccept** (OSRTSOCKET socket, OSRTSOCKET *pNewSocket, OSIPADDR *destAddr, int *destPort)
- int **rtSocketAddrToStr** (OSIPADDR ipAddr, char *pbuf, int bufsize)
- int **rtSocketBind** (OSRTSOCKET socket, OSIPADDR addr, int port)
- int **rtSocketBlockingRead** (OSRTSOCKET socket, ASN1OCTET *pbuf, size_t readBytes)
- int **rtSocketClose** (OSRTSOCKET socket)
- int **rtSocketConnect** (OSRTSOCKET socket, const char *host, int port)
- int **rtSocketCreate** (OSRTSOCKET *psocket)
- int **rtSocketCreateUDP** (OSRTSOCKET *psocket)
- int **rtSocketsInit** (void)
- int **rtSocketsCleanup** (void)
- int **rtSocketListen** (OSRTSOCKET socket, int maxConnection)
- int **rtSocketRecv** (OSRTSOCKET socket, ASN1OCTET *pbuf, ASN1UINT bufsize)
- int **rtSocketRecvFrom** (OSRTSOCKET socket, ASN1OCTET *pbuf, ASN1UINT bufsize, char *remotehost, int *remoteport)
- int **rtSocketSend** (OSRTSOCKET socket, const ASN1OCTET *pdata, ASN1UINT size)
- int **rtSocketSendTo** (OSRTSOCKET socket, const ASN1OCTET *pdata, ASN1UINT size, const char *remotehost, int remoteport)
- int **rtSocketStrToAddr** (const char *pIPAddrStr, OSIPADDR *pIPAddr)
- int **rtSocketGetLocalIPAddress** (char *pIPAddr, int bufSize)

Function Documentation

int rtSocketAccept (OSRTSOCKET *socket*, OSRTSOCKET * *pNewSocket*, OSIPADDR * *destAddr*, int * *destPort*)

This function permits an incoming connection attempt on a socket. It extracts the first connection on the queue of pending connections on socket. It then creates a new socket and returns a handle to the new socket. The newly created socket is the socket that will handle the actual connection and has the same properties as original socket. See description of 'accept' socket function for further details.

Parameters:

socket The socket handle created by the **rtSocketCreate** function.

pNewSocket The pointer to variable to receive the new socket handle.

destAddr Optional pointer to a buffer that receives the IP address of the connecting entity. It may be NULL.

destPort Optional pointer to a buffer that receives the port of the connecting entity. It may be NULL.

Returns:

Completion status of operation: 0 (ASN_OK) = success, negative return value is error.

int rtSocketAddrToStr (OSIPADDR *ipAddr*, char * *pbuf*, int *bufsize*)

This function converts an IP address to its string representation.

Parameters:

ipAddr The IP address to be converted.

pbuf Pointer to the buffer to receive a string with the IP address.

bufsize Size of the buffer.

Returns:

Completion status of operation: 0 (ASN_OK) = success, negative return value is error.

int rtSocketBind (OSRTSOCKET *socket*, OSIPADDR *addr*, int *port*)

This function associates a local address with a socket. It is used on an unconnected socket before subsequent calls to the **rtSocketConnect** or **rtSocketListen** functions. See description of 'bind' socket function for further details.

Parameters:

socket The socket handle created by the **rtSocketCreate** function.

addr The local IP address to assign to the socket.

port The local port number to assign to the socket.

Returns:

Completion status of operation: 0 (ASN_OK) = success, negative return value is error.

int rtSocketBlockingRead (OSRTSOCKET *socket*, ASN1OCTET * *pbuf*, size_t *readBytes*)

This function receives data from a connected socket. In this case, the connection is blocked until either the requested number of bytes is received or the socket is closed or an error occurs.

Parameters:

socket The socket handle created by the **rtSocketCreate** or **rtSocketAccept** function.

pbuf Pointer to the buffer for the incoming data.

readBytes Number of bytes to receive.

Returns:

If no error occurs, returns the number of bytes received. Otherwise, the negative value is error code.

int rtSocketClose (OSRTOCKET *socket*)

This function closes an existing socket.

Parameters:

socket The socket handle created by the **rtSocketCreate** or **rtSocketAccept** function.

Returns:

Completion status of operation: 0 (ASN_OK) = success, negative return value is error.

int rtSocketConnect (OSRTOCKET *socket*, const char * *host*, int *port*)

This function establishes a connection to a specified socket. It is used to create a connection to the specified destination. When the socket call completes successfully, the socket is ready to send and receive data. See description of 'connect' socket function for further details.

Parameters:

socket The socket handle created by the **rtSocketCreate** function.

host The null-terminated string with the IP address in the following format: "NNN.NNN.NNN.NNN", where NNN is a number in the range (0..255).

port The destination port to connect.

Returns:

Completion status of operation: 0 (ASN_OK) = success, negative return value is error.

int rtSocketCreate (OSRTOCKET * *psocket*)

This function creates a socket. The only streaming TCP/IP sockets are supported at the moment.

Parameters:

psocket The pointer to the socket handle variable to receive the handle of new socket.

Returns:

Completion status of operation: 0 (ASN_OK) = success, negative return value is error.

int rtSocketCreateUDP (OSRTOCKET * *psocket*)

This function creates a UDP datagram socket.

Parameters:

psocket The pointer to the socket handle variable to receive the handle of new socket.

Returns:

Completion status of operation: 0 (ASN_OK) = success, negative return value is error.

int rtSocketGetLocalIPAddress (char * *pIPAddr*, int *bufSize*)

This function retrieves the IP address of the local host.

Parameters:

pIPAddr Pointer to a char buffer in which local IP address will be returned.

bufSize Size of the buffer passed.

Returns:

Completion status of operation: 0 (ASN_OK) = success, negative return value is error.

int rtSocketListen (OSRTOCKET *socket*, int *maxConnection*)

This function places a socket a state where it is listening for an incoming connection. To accept connections, a socket is first created with the **rtSocketCreate** function and bound to a local address with the **rtSocketBind** function, a *maxConnection* for incoming connections is specified with **rtSocketListen**, and then the connections are accepted with the **rtSocketAccept** function. See description of 'listen' socket function for further details.

Parameters:

socket The socket handle created by the **rtSocketCreate** function.
maxConnection Maximum length of the queue of pending connections.

Returns:

Completion status of operation: 0 (ASN_OK) = success, negative return value is error.

int rtSocketRecv (OSRTOCKET *socket*, ASN1OCTET * *pbuf*, ASN1UINT *bufsize*)

This function receives data from a connected socket. It is used to read incoming data on sockets. The socket must be connected before calling this function. See description of 'recv' socket function for further details.

Parameters:

socket The socket handle created by the **rtSocketCreate** or **rtSocketAccept** function.
pbuf Pointer to the buffer for the incoming data.
bufsize Length of the buffer.

Returns:

If no error occurs, returns the number of bytes received. Otherwise, the negative value is error code.

int rtSocketRecvFrom (OSRTOCKET *socket*, ASN1OCTET * *pbuf*, ASN1UINT *bufsize*, char * *remotehost*, int * *remoteport*)

This function receives data from a connected/unconnected socket. It is used to read incoming data on sockets. It populates the *remotehost* and *remoteport* parameters with information of remote host. See description of 'recvfrom' socket function for further details.

Parameters:

socket The socket handle created by the **rtSocketCreate** function.
pbuf Pointer to the buffer for the incoming data.
bufsize Length of the buffer.
remotehost Pointer to a buffer in which remote ip address will be returned.
remoteport Pointer to an int in which remote port number will be returned.

Returns:

If no error occurs, returns the number of bytes received. Otherwise, the negative value is error code.

int rtSocketsCleanup (void)

This function terminates use of sockets by an application. This function must be called after use of all sockets is complete.

Returns:

Completion status of operation: 0 (ASN_OK) = success, negative return value is error.

int rtSocketSend (OSRTOCKET *socket*, const ASN1OCTET * *pdata*, ASN1UINT *size*)

This function sends data on a connected socket. It is used to write outgoing data on a connected socket. See description of 'send' socket function for further details.

Parameters:

socket The socket handle created by the **rtSocketCreate** or **rtSocketAccept** function.
pdata Buffer containing the data to be transmitted.
size Length of the data in *pdata*.

Returns:

Completion status of operation: 0 (ASN_OK) = success, negative return value is error.

int rtSocketSendTo (OSRTSOCKET *socket*, const ASN1OCTET * *pdata*, ASN1UINT *size*, const char * *remotehost*, int *remoteport*)

This function sends data on a connected or unconnected socket. See description of 'sendto' socket function for further details.

Parameters:

socket The socket handle created by the **rtSocketCreate** or **rtSocketAccept** function.
pdata Buffer containing the data to be transmitted.
size Length of the data in *pdata*.
remotehost Remote host ip address to which data has to be sent.
remoteport Remote port ip address to which data has to be sent.

Returns:

Completion status of operation: 0 (ASN_OK) = success, negative return value is error.

int rtSocketsInit (void)

This function initiates use of sockets by an application. This function must be called first before use sockets.

Returns:

Completion status of operation: 0 (ASN_OK) = success, negative return value is error.

int rtSocketStrToAddr (const char * *pIPAddrStr*, OSIPADDR * *pIPAddr*)

This function converts the string with IP address to a double word representation. The converted address may be used with the **rtSocketBind** function.

Parameters:

pIPAddrStr The null-terminated string with the IP address in the following format: "NNN.NNN.NNN.NNN", where NNN is a number in the range (0..255).
pIPAddr Pointer to the converted IP address.

Returns:

Completion status of operation: 0 (ASN_OK) = success, negative return value is error.

Stream structure definitions.

Classes

- struct **OSRTStream**

Defines

- #define **OSRTSTRMF_INPUT** 0x0001
- #define **OSRTSTRMF_OUTPUT** 0x0002
- #define **OSRTSTRMF_REQBUFFER** 0x4000

- `#define OSRTSTRMF_BUFFERED 0x8000`
- `#define OSRTSTRMF_BUF_INPUT (OSRTSTRMF_INPUT|OSRTSTRMF_BUFFERED)`
- `#define OSRTSTRMF_BUF_OUTPUT (OSRTSTRMF_OUTPUT|OSRTSTRMF_BUFFERED)`
- `#define OSRTSTRMID_FILE 1`
- `#define OSRTSTRMID_SOCKET 2`
- `#define OSRTSTRMID_MEMORY 3`
- `#define OSRTSTRMID_BUFFERED 4`
- `#define OSRTSTRMID_DIRECTBUF 5`
- `#define OSRTSTRMID_USER 1000`
- `#define OSRTSTRM_K_BUFSIZE 1024`
- `#define OSRTSTRM_K_INVALIDMARK ((ASN1UINT)-1)`
- `#define OSRTSTREAM_BYTEINDEX(pctx) ((pctx)->pStream->bytesProcessed + (pctx)->buffer.byteIndex)`
- `#define OSRTSTREAM_ID(pctx) ((pctx)->pStream->id)`
- `#define OSRTSTREAM_FLAGS(pctx) ((pctx)->pStream->flags)`

Typedefs

- `typedef long(* OSRTStreamReadProc)(struct OSRTStream *pStream, ASN1OCTET *pbuffer, size_t bufsize)`
- `typedef long(* OSRTStreamBlockingReadProc)(struct OSRTStream *pStream, ASN1OCTET *pbuffer, size_t toReadBytes)`
- `typedef int(* OSRTStreamFlushProc)(struct OSRTStream *pStream)`
- `typedef int(* OSRTStreamCloseProc)(struct OSRTStream *pStream)`
- `typedef long(* OSRTStreamWriteProc)(struct OSRTStream *pStream, const ASN1OCTET *data, size_t numocts)`
- `typedef int(* OSRTStreamSkipProc)(struct OSRTStream *pStream, size_t skipBytes)`
- `typedef int(* OSRTStreamMarkProc)(struct OSRTStream *pStream, size_t readAheadLimit)`
- `typedef int(* OSRTStreamResetProc)(struct OSRTStream *pStream)`
- `typedef OSRTStream OSRTStream`

Typedef Documentation

typedef struct OSRTStream OSRTStream

This structure is used to define a stream control block for keeping track of stream operations. A user may implement his own specific stream operations by defining read, skip, and close functions for input streams and write, flush, and close for output streams.

typedef long(* OSRTStreamBlockingReadProc)(struct OSRTStream *pStream, ASN1OCTET *pbuffer, size_t toReadBytes)

Stream blockingRead function pointer type. A user may implement a customized read function for specific input streams. The blockingRead function is defined in the **OSRTStream** control structure.

typedef int(* OSRTStreamCloseProc)(struct OSRTStream *pStream)

Pointer to stream close function. User may implement his own read function for any specific streams.

typedef int(* OSRTStreamFlushProc)(struct OSRTStream *pStream)

Pointer to stream flush function. User may implement his own read function for any specific output streams.

typedef int(* OSRTStreamMarkProc)(struct OSRTStream *pStream, size_t readAheadLimit)

Stream mark function pointer type. A user may implement a customized function for a specific input stream type. The mark function is defined in the **OSRTStream** control structure.

typedef long(* OSRTStreamReadProc)(struct OSRTStream *pStream, ASN1OCTET *pbuffer, size_t bufsize)

Pointer to stream read function. User may implement his own read function for any specific input streams.

typedef int(* OSRTStreamResetProc)(struct OSRTStream *pStream)

Stream reset function pointer type. A user may implement a customized function for a specific input stream type. The reset function is defined in the **OSRTStream** control structure.

typedef int(* OSRTStreamSkipProc)(struct OSRTStream *pStream, size_t skipBytes)

Stream skip function pointer type. A user may implement a customized function for a specific input stream type. The skip function is defined in the **OSRTStream** control structure.

typedef long(* OSRTStreamWriteProc)(struct OSRTStream *pStream, const ASN1OCTET *data, size_t numocts)

Pointer to stream write function. User may implement his own read function for any specific output streams.

Low-level unbuffered stream functions.

Detailed Description

This group of stream functions is used for unbuffered stream I/O. All operations with streams are performed using a context block structure. The difference between several kind of streams is only in the opening functions (see **rtStreamFileOpen**, **rtStreamFileAttach**, **rtStreamSocketAttach**, **rtStreamMemoryCreate**, **rtStreamMemoryAttach**). These functions may be used for any input/output operations with streams. Each stream should be initialized first by call to **rtStreamInit** function. After initialization, a stream may be opened for reading or writing by a call to one of the following functions: **rtStreamFileOpen**, **rtStreamFileAttach**, **rtStreamSocketAttach**, **rtStreamMemoryCreate**, **rtStreamMemoryAttach**.

Functions

- int **rtStreamRelease** (ASN1CTXT *pctxt)
- int **rtStreamClose** (ASN1CTXT *pctxt)

- int **rtStreamFlush** (ASN1CTXT *pctx)
 - int **rtStreamInit** (ASN1CTXT *pctx)
 - long **rtStreamRead** (ASN1CTXT *pctx, ASN1OCTET *pbuffer, size_t bufsize)
 - long **rtStreamBlockingRead** (ASN1CTXT *pctx, ASN1OCTET *pbuffer, size_t readBytes)
 - int **rtStreamSkip** (ASN1CTXT *pctx, size_t skipBytes)
 - long **rtStreamWrite** (ASN1CTXT *pctx, const ASN1OCTET *data, size_t numocts)
 - int **rtStreamGetIOBytes** (ASN1CTXT *pctx, size_t *pPos)
 - int **rtStreamGetIOBytes56** (ASN1CTXT *pctx)
 - int **rtStreamMark** (ASN1CTXT *pctx, size_t readAheadLimit)
 - int **rtStreamReset** (ASN1CTXT *pctx)
 - ASN1BOOL **rtStreamMarkSupported** (ASN1CTXT *pctx)
 - ASN1BOOL **rtStreamIsOpened** (ASN1CTXT *pctx)
 - ASN1BOOL **rtStreamIsReadable** (ASN1CTXT *pctx)
 - ASN1BOOL **rtStreamIsWritable** (ASN1CTXT *pctx)
-

Function Documentation

long **rtStreamBlockingRead** (ASN1CTXT * *pctx*, ASN1OCTET * *pbuffer*, size_t *readBytes*)

This function reads up to 'bufsize' bytes of data from the input stream into an array of octets. An attempt is made to read as many as bufsize octets, but a smaller number may be read, possibly zero. The number of octets actually read is returned as an integer. This function blocks until input data is available, end of file is detected, or another error is occurred.

Parameters:

pctx Pointer to a context structure variable which has been initialized for stream operations via a call to **rtStreamInit**.
pbuffer Pointer to a buffer to receive data.
readBytes Number of bytes to read.

Returns:

The total number of octets read into the buffer, or negative value with error code if any error is occurred.

int **rtStreamClose** (ASN1CTXT * *pctx*)

This function closes the input or output stream and releases any system resources associated with the stream. For output streams this function also flushes all internal buffers to the stream.

Parameters:

pctx Pointer to context structure variable, which is initialized for streaming operations by call to **rtStreamInit**.

int **rtStreamFlush** (ASN1CTXT * *pctx*)

This function flushes the output stream and forces any buffered output octets to be written out.

Parameters:

pctx Pointer to context structure variable, which is initialized for streaming operations by call to **rtStreamInit**. Also, the stream must be opened as an input stream by call to appropriate function (see **rtStreamFileOpen**, **rtStreamFileAttach**, **rtStreamSocketAttach**, **rtStreamMemoryCreate**, **rtStreamMemoryAttach**).

Returns:

Completion status of operation: 0 (ASN_OK) = success, negative return value is error.

int rtStreamGetIOBytes (ASN1CTXT * *pctxt*, size_t * *pPos*)

This function returns the number of processed octets. If the stream was opened as an input stream, then it returns the total number of read octets. If the stream was opened as an output stream, then it returns the total number of written octets. Otherwise, this function returns an error code.

Parameters:

pctxt Pointer to a context structure variable which has been initialized for stream operations via a call to `rtxStreamInit`.

pPos Pointer to argument to receive total number of processed octets.

Returns:

Completion status of operation: 0 (ASN_OK) = success, negative return value is error.

int rtStreamInit (ASN1CTXT * *pctxt*)

This function initializes a stream part of the context block. This function should be called first before any operation with a stream.

Parameters:

pctxt Pointer to context structure variable, for which stream to be initialized.

Returns:

Completion status of operation: 0 (ASN_OK) = success, negative return value is error.

ASN1BOOL rtStreamIsOpened (ASN1CTXT * *pctxt*)

Tests if this stream opened (for reading or writing).

Parameters:

pctxt Pointer to a context structure variable that has been initialized for stream operations.

Returns:

TRUE if this stream is opened for reading or writing; FALSE otherwise.

ASN1BOOL rtStreamIsReadable (ASN1CTXT * *pctxt*)

Tests if this stream opened for reading.

Parameters:

pctxt Pointer to a context structure variable that has been initialized for stream operations.

Returns:

TRUE if this stream is opened for reading; FALSE otherwise.

ASN1BOOL rtStreamIsWritable (ASN1CTXT * *pctxt*)

Tests if this stream opened for riting.

Parameters:

pctxt Pointer to a context structure variable that has been initialized for stream operations.

Returns:

TRUE if this stream is opened for writing; FALSE otherwise.

int rtStreamMark (ASN1CTXT * *pctxt*, size_t *readAheadLimit*)

Marks the current position in this input stream. A subsequent call to the `rtxStreamReset` function repositions this stream at the last marked position so that subsequent reads re-read the same bytes. The `readAheadLimit` argument tells this input stream to allow many bytes to be read before the mark position gets invalidated.

Parameters:

pctxt Pointer to a context structure variable that has been initialized for stream operations.
readAheadLimit The maximum limit of bytes that can be read before the mark position becomes invalid.

Returns:

Completion status of operation: 0 = success, negative return value is error.

ASN1BOOL rtStreamMarkSupported (ASN1CTXT * *pctxt*)

Tests if this input stream supports the mark and reset methods. Whether or not mark and reset are supported is an invariant property of a particular input stream instance. By default, it returns FALSE.

Parameters:

pctxt Pointer to a context structure variable that has been initialized for stream operations.

Returns:

TRUE if this stream instance supports the mark and reset methods; FALSE otherwise.

long rtStreamRead (ASN1CTXT * *pctxt*, ASN1OCTET * *pbuffer*, size_t *bufsize*)

This function reads up to '`bufsize`' bytes of data from the input stream into an array of octets. An attempt is made to read as many as `bufsize` octets, but a smaller number may be read, possibly zero. The number of octets actually read is returned as an integer. This functions blocks until input data is available, end of file is detected, or another error is occurred.

Parameters:

pctxt Pointer to context structure variable, which is initialized for streaming operations by call to `rtStreamInit`. Also, the stream must be opened as an input stream by call to appropriate function (see `rtStreamFileOpen`, `rtStreamFileAttach`, `rtStreamSocketAttach`, `rtStreamMemoryCreate`, `rtStreamMemoryAttach`).
pbuffer Pointer to a buffer to receive a data.
bufsize Size of the buffer.

Returns:

The total number of octets read into the buffer, or negative value with error code if any error is occurred.

int rtStreamReset (ASN1CTXT * *pctxt*)

Repositions this stream to the position recorded by the last call to the `rtxStreamMark` function.

Parameters:

pctxt Pointer to a context structure variable that has been initialized for stream operations.

Returns:

Completion status of operation: 0 = success, negative return value is error.

int rtStreamSkip (ASN1CTXT * *pctxt*, size_t *skipBytes*)

This function skips over and discards the specified amount of data octets from this input stream. The skip method may end up skipping over some smaller number of octets, possibly 0. The actual number of octets skipped is returned.

Parameters:

pctxt Pointer to context structure variable, which is initialized for streaming operations by call to **rtStreamInit**. Also, the stream must be opened as an input stream by call to appropriate function (see **rtStreamFileOpen**, **rtStreamFileAttach**, **rtStreamSocketAttach**, **rtStreamMemoryCreate**, **rtStreamMemoryAttach**).
skipBytes The number of octets to be skipped.

Returns:

Completion status of operation: 0 (ASN_OK) = success, negative return value is error.

long rtStreamWrite (ASN1CTXT * *pctxt*, const ASN1OCTET * *data*, size_t *numocts*)

This function writes the specified amount of octets from the specified array to the output stream.

Parameters:

pctxt Pointer to context structure variable, which is initialized for streaming operations by call to **rtStreamInit**. Also, the stream must be opened as an output stream by call to appropriate function (see **rtStreamFileOpen**, **rtStreamFileAttach**, **rtStreamSocketAttach**, **rtStreamMemoryCreate**, **rtStreamMemoryAttach**).
data The pointer to data to be written.
numocts The number of octets to write.

Returns:

The total number of octets written into the stream, or negative value with error code if any error is occurred.

Buffered stream functions.

Detailed Description

The **rtStreamDirectBuf*** functions are used to operate with buffered streams. They use the 'buffer' field of the ASN1CTXT context block structure as a buffer for input/output operations. These functions are used for performing stream-oriented decoding/encoding. Each buffered stream should be initialized first by a call to **rtStreamDirectBufCreate** function. After initialization, the stream may be opened for reading or writing by a call to one of the following functions: **rtStreamFileOpen**, **rtStreamFileAttach**, **rtStreamSocketAttach**, **rtStreamMemoryCreate**, or **rtStreamMemoryAttach**.

Functions

- int **rtStreamBufClose** (ASN1CTXT **pctxt*)
- int **rtStreamBufFlush** (ASN1CTXT **pctxt*)
- int **rtStreamBufInit** (ASN1CTXT **pctxt*)
- int **rtStreamBufMark** (ASN1CTXT **pctxt*, size_t *readAheadLimit*)
- int **rtStreamBufPreRead** (ASN1CTXT **pctxt*, size_t *size*)
- int **rtStreamBufRead** (ASN1CTXT **pctxt*, ASN1OCTET **pdata*, size_t *size*)

- int **rtStreamBufSkip** (ASN1CTXT *pctx, size_t skipBytes)
 - int **rtStreamBufReset** (ASN1CTXT *pctx)
 - int **rtStreamBufWrite** (ASN1CTXT *pctx, const ASN1OCTET *data, size_t numocts)
-

Function Documentation

int **rtStreamBufClose** (ASN1CTXT * *pctx*)

This function closes the buffered input or output stream and releases any system resources associated with the stream. For output streams this function also flushes all internal buffers to the stream.

Parameters:

pctx Pointer to context structure variable, which is initialized for streaming operations by call to **rtStreamBufInit**.

int **rtStreamBufFlush** (ASN1CTXT * *pctx*)

This function flushes the output stream and forces any buffered output octets to be written out.

Parameters:

pctx Pointer to context structure variable, which is initialized for streaming operations by call to **rtStreamBufInit**. Also, the stream must be opened as an input stream by call to appropriate function (see **rtStreamFileOpen**, **rtStreamFileAttach**, **rtStreamSocketAttach**, **rtStreamMemoryCreate**, **rtStreamMemoryAttach**).

Returns:

Completion status of operation: 0 (ASN_OK) = success, negative return value is error.

int **rtStreamBufInit** (ASN1CTXT * *pctx*)

This function initializes a buffered stream. This function should be called first before any operation with a buffered stream.

Parameters:

pctx Pointer to context structure variable, for which stream to be initialized.

Returns:

Completion status of operation: 0 (ASN_OK) = success, negative return value is error.

int **rtStreamBufMark** (ASN1CTXT * *pctx*, size_t *readAheadLimit*)

Marks the current position in this input stream. A subsequent call to the **rtStreamBufReset** function repositions this stream at the last marked position so that subsequent reads re-read the same bytes. The *readAheadLimit* argument tells this input stream to allow that many bytes to be read before the mark position gets invalidated.

Parameters:

pctx Pointer to context structure variable, which is initialized for streaming operations by call to **rtStreamBufInit**. Also, the stream must be opened as an input stream by call to appropriate function (see **rtStreamFileOpen**, **rtStreamFileAttach**, **rtStreamSocketAttach**, **rtStreamMemoryCreate**, **rtStreamMemoryAttach**).
readAheadLimit the maximum limit of bytes that can be read before the mark position becomes invalid.

Returns:

Completion status of operation: 0 (ASN_OK) = success, negative return value is error.

int rtStreamBufPreRead (ASN1CTXT * *pctxt*, size_t *size*)

This function pre-reads a 'size' number of octets of data from the input stream into an internal stream buffer. This function blocks until all necessary data is read, end of file is detected, or another error occurs. This function guarantees that the internal buffer will contain at least the 'size' number of octets. Thus, it is possible to obtain direct access to the memory buffer with pre-read data by using the 'buffer' field of the ASN1CTXT context block structure.

Parameters:

pctxt Pointer to context structure variable, which is initialized for streaming operations by call to **rtStreamBufInit**. Also, the stream must be opened as an input stream by call to appropriate function (see **rtStreamFileOpen**, **rtStreamFileAttach**, **rtStreamSocketAttach**, **rtStreamMemoryCreate**, **rtStreamMemoryAttach**).

size Number of octets to be pre-read.

Returns:

Completion status of operation: 0 (ASN_OK) = success, negative return value is error.

int rtStreamBufRead (ASN1CTXT * *pctxt*, ASN1OCTET * *pdata*, size_t *size*)

This function reads 'size' number of bytes of data from the input stream into an array of octets. If all or some octets already are in the internal buffer, then these octets will be copied from the internal buffer to the specified buffer. In contrast to the **rtStreamRead** function, this function blocks until all requested input data is read, end of file is detected, or another error occurs.

Parameters:

pctxt Pointer to context structure variable, which is initialized for streaming operations by call to **rtStreamBufInit**. Also, the stream must be opened as an input stream by call to appropriate function (see **rtStreamFileOpen**, **rtStreamFileAttach**, **rtStreamSocketAttach**, **rtStreamMemoryCreate**, **rtStreamMemoryAttach**).

pdata Pointer to a buffer to receive a data.

size Size of the buffer.

Returns:

Completion status of operation: 0 (ASN_OK) = success, negative return value is error.

int rtStreamBufReset (ASN1CTXT * *pctxt*)

This function repositions the stream to the position at the time the mark method was last called on the input stream.

Parameters:

pctxt Pointer to context structure variable, which is initialized for streaming operations by call to **rtStreamBufInit**. Also, the stream must be opened as an input stream by call to appropriate function (see **rtStreamFileOpen**, **rtStreamFileAttach**, **rtStreamSocketAttach**, **rtStreamMemoryCreate**, **rtStreamMemoryAttach**).

Returns:

Completion status of operation: 0 (ASN_OK) = success, negative return value is error.

int rtStreamBufSkip (ASN1CTXT * *pctxt*, size_t *skipBytes*)

This function skips over and discards the specified amount of data octets from the input stream. The skip method may end up skipping over some smaller number of octets, possibly 0. The actual number of octets skipped is returned.

Parameters:

pctxt Pointer to context structure variable, which is initialized for streaming operations by call to **rtStreamBufInit**. Also, the stream must be opened as an input stream by call to appropriate function (see **rtStreamFileOpen**, **rtStreamFileAttach**, **rtStreamSocketAttach**, **rtStreamMemoryCreate**, **rtStreamMemoryAttach**).

skipBytes The number of octets to be skipped.

Returns:

Completion status of operation: 0 (ASN_OK) = success, negative return value is error.

int rtStreamBufWrite (ASN1CTXT * pctxt, const ASN1OCTET * data, size_t numocts)

This function writes the specified amount of octets from the specified array to the buffered output stream. Ordinarily this function stores bytes from the given array into the stream buffer, flushing the buffer to the underlying output stream as needed. If the requested length is at least as large as this stream's buffer, however, then this method will flush the buffer and write the bytes directly to the underlying output stream.

Parameters:

pctxt Pointer to context structure variable, which is initialized for streaming operations by call to **rtStreamBufInit**. Also, the stream must be opened as an output stream by call to appropriate function (see **rtStreamFileOpen**, **rtStreamFileAttach**, **rtStreamSocketAttach**, **rtStreamMemoryCreate**, **rtStreamMemoryAttach**).

data The pointer to data to be written.

numocts The number of octets to write.

Returns:

Completion status of operation: 0 (ASN_OK) = success, negative return value is error.

File, socket, and memory streams.

Functions

- int **rtStreamFileAttach** (ASN1CTXT *pctxt, FILE *pFile, ASN1USINT flags)
 - int **rtStreamFileOpen** (ASN1CTXT *pctxt, const char *pFilename, ASN1USINT flags)
 - int **rtStreamFileCreateReader** (ASN1CTXT *pctxt, const char *pFilename)
 - int **rtStreamFileCreateWriter** (ASN1CTXT *pctxt, const char *pFilename)
 - int **rtStreamSocketAttach** (ASN1CTXT *pctxt, OSRTSOCKET socket, ASN1USINT flags)
 - int **rtStreamSocketCreateReader** (ASN1CTXT *pctxt, OSRTSOCKET socket)
 - int **rtStreamSocketCreateWriter** (ASN1CTXT *pctxt, const char *host, int port)
 - int **rtStreamSocketCreateWriter2** (ASN1CTXT *pctxt, OSRTSOCKET socket)
 - int **rtStreamSocketSetOwnership** (ASN1CTXT *pctxt, ASN1BOOL ownSocket)
 - int **rtStreamMemoryCreate** (ASN1CTXT *pctxt, ASN1USINT flags)
 - int **rtStreamMemoryAttach** (ASN1CTXT *pctxt, ASN1OCTET *pMemBuf, size_t bufSize, ASN1USINT flags)
 - ASN1OCTET * **rtStreamMemoryGetBuffer** (ASN1CTXT *pctxt, size_t *pSize)
 - int **rtStreamMemoryCreateReader** (ASN1CTXT *pctxt, ASN1OCTET *pMemBuf, size_t bufSize)
 - int **rtStreamMemoryCreateWriter** (ASN1CTXT *pctxt, ASN1OCTET *pMemBuf, size_t bufSize)
-

Function Documentation

int rtStreamFileAttach (ASN1CTXT * *pctxt*, FILE * *pFile*, ASN1USINT *flags*)

This function attaches the existing file structure pointer to the stream. The file should have already been opened either for reading or writing. The 'flags' parameter specifies the access mode for the stream - input or output.

Parameters:

pctxt Pointer to context structure variable, which is initialized for streaming operations by call to **rtStreamInit** or **rtStreamBufInit**.

pFile Pointer to FILE structure. File should be already opened either for the writing or reading.

flags Specifies the access mode for the stream:

- OSRTSTRMF_INPUT = input (reading) stream;
- OSRTSTRMF_OUTPUT = output (writing) stream.

Returns:

Completion status of operation: 0 (ASN_OK) = success, negative return value is error.

int rtStreamFileCreateReader (ASN1CTXT * *pctxt*, const char * *pFilename*)

This function creates an input file stream using the specified file name.

Parameters:

pctxt Pointer to a context structure variable that has been initialized for stream operations.

pFilename Pointer to null-terminated string that contains the name of file.

Returns:

Completion status of operation: 0 = success, negative return value is error.

int rtStreamFileCreateWriter (ASN1CTXT * *pctxt*, const char * *pFilename*)

This function creates an output file stream using the file name.

Parameters:

pctxt Pointer to a context structure variable that has been initialized for stream operations.

pFilename Pointer to null-terminated string that contains the name of file.

Returns:

Completion status of operation: 0 = success, negative return value is error.

int rtStreamFileOpen (ASN1CTXT * *pctxt*, const char * *pFilename*, ASN1USINT *flags*)

This function opens a file stream. The 'flags' parameter specifies the access mode for the stream - input or output.

Parameters:

pctxt Pointer to context structure variable, which is initialized for streaming operations by call to **rtStreamInit** or **rtStreamBufInit**.

pFilename Pointer to null-terminated string that contains the name of file.

flags Specifies the access mode for the stream:

- OSRTSTRMF_INPUT = input (reading) stream;
- OSRTSTRMF_OUTPUT = output (writing) stream.

Returns:

Completion status of operation: 0 (ASN_OK) = success, negative return value is error.

int rtStreamMemoryAttach (ASN1CTXT * *pctxt*, ASN1OCTET * *pMemBuf*, size_t *bufSize*, ASN1USINT *flags*)

This function opens a memory stream by using a specified fixed-size memory buffer. The 'flags' parameter specifies the access mode for the stream - input or output.

Parameters:

- pctxt* Pointer to context structure variable, which is initialized for streaming operations by call to **rtStreamInit** or **rtStreamBufInit**.
- pMemBuf* The pointer to the buffer.
- bufSize* The size of the buffer.
- flags* Specifies the access mode for the stream:
 - OSRTSTRMF_INPUT = input (reading) stream;
 - OSRTSTRMF_OUTPUT = output (writing) stream.

Returns:

Completion status of operation: 0 (ASN_OK) = success, negative return value is error.

int rtStreamMemoryCreate (ASN1CTXT * *pctxt*, ASN1USINT *flags*)

This function opens a memory stream (i.e. a stream to or from a dynamic byte array in memory). A dynamic memory buffer will be created by this function. The 'flags' parameter specifies the access mode for the stream - input or output.

Parameters:

- pctxt* Pointer to context structure variable, which is initialized for streaming operations by call to **rtStreamInit** or **rtStreamBufInit**.
- flags* Specifies the access mode for the stream:
 - OSRTSTRMF_INPUT = input (reading) stream;
 - OSRTSTRMF_OUTPUT = output (writing) stream.

Returns:

Completion status of operation: 0 (ASN_OK) = success, negative return value is error.

int rtStreamMemoryCreateReader (ASN1CTXT * *pctxt*, ASN1OCTET * *pMemBuf*, size_t *bufSize*)

This function creates an input memory stream using the specified buffer.

Parameters:

- pctxt* Pointer to a context structure variable that has been initialized for stream operations.
- pMemBuf* The pointer to the buffer
- bufSize* The size of the buffer

Returns:

Completion status of operation: 0 = success, negative return value is error.

int rtStreamMemoryCreateWriter (ASN1CTXT * *pctxt*, ASN1OCTET * *pMemBuf*, size_t *bufSize*)

This function creates an output memory stream using the specified buffer. If *pMemBuf* or *bufSize* is NULL then new buffer will be allocated.

Parameters:

- pctxt* Pointer to a context structure variable that has been initialized for stream operations.
- pMemBuf* The pointer to the buffer. Can be NULL - new buffer will be allocated in this case.
- bufSize* The size of the buffer. Can be 0 - new buffer will be allocated in this case.

Returns:

Completion status of operation: 0 = success, negative return value is error.

ASN1OCTET* rtStreamMemoryGetBuffer (ASN1CTXT * *pctxt*, size_t * *pSize*)

This function returns a memory stream's buffer pointer and size.

Parameters:

pctxt Pointer to context structure variable, which is initialized for memory streaming operations by call to **rtStreamMemoryCreate** or **rtStreamMemoryAttach**.

pSize The pointer to unsigned integer to receive the size of buffer.

Returns:

The pointer to memory buffer.

int rtStreamSocketAttach (ASN1CTXT * *pctxt*, OSRTSOCKET *socket*, ASN1USINT *flags*)

This function attaches an existing socket handle to a stream. The socket should be already opened and connected. The 'flags' parameter specifies the access mode for the stream - input or output.

Parameters:

pctxt Pointer to context structure variable, which is initialized for streaming operations by call to **rtStreamInit** or **rtStreamBufInit**.

socket The socket handle, created by **rtSocketCreate**. The connection for this socket should be already opened by using **rtSocketConnect** or **rtSocketAccept** functions.

flags Specifies the access mode for the stream:

- OSRTSTRMF_INPUT = input (reading) stream;
- OSRTSTRMF_OUTPUT = output (writing) stream.

Returns:

Completion status of operation: 0 (ASN_OK) = success, negative return value is error.

int rtStreamSocketCreateReader (ASN1CTXT * *pctxt*, OSRTSOCKET *socket*)

This function creates an input socket stream using the specified opened socket.

Parameters:

pctxt Pointer to a context structure variable that has been initialized for stream operations.

socket The socket handle, created by **rtSocketCreate**. The connection for this socket should be already opened by using **rtSocketConnect** or **rtSocketAccept** functions.

Returns:

Completion status of operation: 0 = success, negative return value is error.

int rtStreamSocketCreateWriter (ASN1CTXT * *pctxt*, const char * *host*, int *port*)

This function opens a socket stream for writing using host name or IP-address and port values.

Parameters:

pctxt Pointer to a context structure variable that has been initialized for stream operations.

host Name of host or IP address to which to connect.

port Port number to which to connect.

Returns:

Completion status of operation: 0 = success, negative return value is error.

int rtStreamSocketCreateWriter2 (ASN1CTXT * *pctxt*, OSRTSOCKET *socket*)

This function opens a socket stream for writing using an existing socket.

Parameters:

pctxt Pointer to a context structure variable that has been initialized for stream operations.
socket Existing socket on which to create stream.

Returns:

Completion status of operation: 0 = success, negative return value is error.

int rtStreamSocketSetOwnership (ASN1CTXT * *pctxt*, ASN1BOOL *ownSocket*)

This function transfers ownership of the socket to or from the stream instance. The socket will be closed and deleted when the stream is closed or goes out of scope. By default stream socket owns the socket.

Parameters:

pctxt Pointer to a context structure variable that has been initialized for stream operations.
ownSocket Boolean value.

ASN1C C Common Runtime Functions Class Documentation

OSRTPrintStream Struct Reference

```
#include <rtPrintStream.h>
```

Detailed Description

Structure to hold information about global PrintStream

Public Attributes

- rtPrintCallback **pfPrintFunc**
 - void * **pPrntStrmInfo**
-

The documentation for this struct was generated from the following file:

- **rtPrintStream.h**

OSRTStream Struct Reference

```
#include <rtStream.h>
```

Detailed Description

This structure is used to define a stream control block for keeping track of stream operations. A user may implement his own specific stream operations by defining read, skip, and close functions for input streams and write, flush, and close for output streams.

Public Attributes

- **OSRTStreamReadProc** read
 - **OSRTStreamBlockingReadProc** blockingRead
 - **OSRTStreamWriteProc** write
 - **OSRTStreamFlushProc** flush
 - **OSRTStreamCloseProc** close
 - **OSRTStreamSkipProc** skip
 - **OSRTStreamMarkProc** mark
 - **OSRTStreamResetProc** reset
 - void * extra
 - size_t bufsize
 - size_t readAheadLimit
 - size_t bytesProcessed
 - size_t markedBytesProcessed
 - size_t ioBytes
 - ASN1UINT id
 - ASN1USINT flags
-

Member Data Documentation

size_t OSRTStream::bufsize

physical size of ptxt->buffer.data buf

size_t OSRTStream::bytesProcessed

the number of bytes already processed

OSRTStreamCloseProc OSRTStream::close

pointer to close function

void* OSRTStream::extra

pointer to stream-specific data

ASN1USINT OSRTStream::flags

flags (see OSRTSTRMF_* macros)

OSRTStreamFlushProc OSRTStream::flush

pointer to flush function

ASN1UINT OSRTStream::id

id of stream (see OSRTSTRMID_* macros)

size_t OSRTStream::ioBytes

the actual number of bytes already read/written

size_t OSRTStream::markedBytesProcessed

the marked number of bytes already processed

OSRTStreamReadProc OSRTStream::read

pointer to read function

size_t OSRTStream::readAheadLimit

read ahead limit (used by **rtStreamMark/rtStreamReset**)

OSRTStreamWriteProc OSRTStream::write

pointer to write function

The documentation for this struct was generated from the following file:

- **rtStream.h**

ASN1C C Common Runtime Functions File Documentation

asn1type.h File Reference

Detailed Description

Common ASN.1 runtime constants, data structure definitions, and run-time functions to support the BER/DER/PER/XER as defined in the ITU-T standards.

```
#include <limits.h>
#include <stdio.h>
#include <string.h>
#include <stddef.h>
#include <setjmp.h>
#include <time.h>
#include <wchar.h>
#include "osSysTypes.h"
#include "asn1version.h"
#include <float.h>
#include "asn1compat.h"
#include "rtMemory.h"
```

Classes

- struct **ASN1OBJID**
- struct **ASN1OID64**
- struct **ASN1OctStr**
- struct **ASN1DynOctStr**
- struct **ASN1DynBitStr**
- struct **ASN1SeqOf**
- struct **ASN1SeqOfOctStr**
- struct **ASN1OpenType**
- struct **Asn1Object**
- struct **Asn116BitCharString**
- struct **Asn132BitCharString**
- struct **Asn1CharArray**
- struct **Asn1CharSet**
- struct **Asn116BitCharSet**
- struct **Asn132BitCharSet**
- struct **ASN1BigInt**
- struct **_Asn1RTSListNode**
- struct **_Asn1RTSList**
- struct **_Asn1RTDListNode**
- struct **_Asn1RTDList**
- struct **_Asn1RTStack**
- struct **_Asn1SizeCnst**
- struct **ASN1BUFFER**
- struct **ASN1BUFSAVE**
- struct **ASN1CCB**

- struct **ASN1ErrLocn**
- struct **ASN1ErrInfo**
- struct **ASN1CTXT**
- struct **ASN1MemBuf**
- struct **_OSDATETIME**

Defines

- #define **ASN1C_EXPANDED_TAGS**
- #define **TV_UNIV** 0
- #define **TV_APPL** 1
- #define **TV_CTXT** 2
- #define **TV_PRIV** 3
- #define **TV_PRIM** 0
- #define **TV_CONS** 1
- #define **TM_UNIV** 0x00000000
- #define **TM_APPL** 0x40000000
- #define **TM_CTXT** 0x80000000
- #define **TM_PRIV** 0xC0000000
- #define **TM_PRIM** 0x00000000
- #define **TM_CONS** 0x20000000
- #define **TM_IDCODE** 0x1FFFFFFF
- #define **ASN_K_BADTAG** 0xFFFFFFFF
- #define **ASN_K_NOTAG** 0xFFFFFFFF
- #define **TM_CLASS** 0xC0
- #define **TM_FORM** 0x20
- #define **TM_CLASS_FORM** 0xE0
- #define **TM_B_IDCODE** 0x1F
- #define **MINMSGLEN** 8
- #define **ASN_OK** 0
- #define **ASN_OK_FRAG** 2
- #define **ASN_E_BUFOVFLW** -1
- #define **ASN_E_ENDOFBUF** -2
- #define **ASN_E_IDNOTFOU** -3
- #define **ASN_E_INVOBJID** -4
- #define **ASN_E_INVLEN** -5
- #define **ASN_E_INVENUM** -6
- #define **ASN_E_SETDUPL** -7
- #define **ASN_E_SETMISRQ** -8
- #define **ASN_E_NOTINSET** -9
- #define **ASN_E_SEQOVFLW** -10
- #define **ASN_E_INVOPT** -11
- #define **ASN_E_NOMEM** -12
- #define **ASN_E_INVHEXS** -14
- #define **ASN_E_INVBINS** -15
- #define **ASN_E_INVREAL** -16
- #define **ASN_E_STROVFLW** -17
- #define **ASN_E_BADVALUE** -18
- #define **ASN_E_UNDEFVAL** -19
- #define **ASN_E_UNDEFTYP** -20
- #define **ASN_E_BADTAG** -21
- #define **ASN_E_TOODEEP** -22
- #define **ASN_E_CONSVIO** -23

- #define ASN_E_RANGERR -24
- #define ASN_E_ENDOFFILE -25
- #define ASN_E_INVUTF8 -26
- #define ASN_E_CONCMODF -27
- #define ASN_E_ILLSTATE -28
- #define ASN_E_OUTOFBND -29
- #define ASN_E_INVPARAM -30
- #define ASN_E_INVFORMAT -31
- #define ASN_E_NOTINIT -32
- #define ASN_E_TOOBIG -33
- #define ASN_E_INVCHAR -34
- #define ASN_E_XMLSTATE -35
- #define ASN_E_XMLPARSE -36
- #define ASN_E_SEQORDER -37
- #define ASN_E_INVINDEX -38
- #define ASN_E_INVTCVAL -39
- #define ASN_E_FILNOTFOU -40
- #define ASN_E_READERR -41
- #define ASN_E_WRITEERR -42
- #define ASN_E_INVBASE64 -43
- #define ASN_E_INVSOCKET -44
- #define ASN_E_HSTUNAVAIL -45
- #define ASN_E_CANTOPEN -46
- #define ASN_E_ARRAYSIZE -47
- #define ASN_E_EXPIRED -48
- #define ASN_E_NULLPTR -49
- #define ASN_E_TOOMANY -50
- #define ASN_E_LOOPDETECTED -51
- #define ASN_E_NOTSUPP -99
- #define ASN_K_INDEFLEN -9999
- #define ASN_E_FILEREAD ASN_E_READERR
- #define ASN_E_FILEWRITE ASN_E_WRITEERR
- #define ASN_E_XMLLIBNFOU ASN_E_NOTSUPP
- #define ASN_E_XMLLIBINV ASN_E_NOTSUPP
- #define ASN_ID_EOC 0
- #define ASN_ID_BOOL 1
- #define ASN_ID_INT 2
- #define ASN_ID_BITSTR 3
- #define ASN_ID_OCTSTR 4
- #define ASN_ID_NULL 5
- #define ASN_ID_OBJID 6
- #define ASN_ID_OBJDSC 7
- #define ASN_ID_EXTERN 8
- #define ASN_ID_REAL 9
- #define ASN_ID_ENUM 10
- #define ASN_ID_EPDV 11
- #define ASN_ID_RELOID 13
- #define ASN_ID_SEQ 16
- #define ASN_ID_SET 17
- #define ASN_SEQ_TAG 0x30
- #define ASN_SET_TAG 0x31
- #define ASN_ID_NumericString 18
- #define ASN_ID_PrintableString 19

- #define ASN_ID_TeletexString 20
- #define ASN_ID_T61String ASN_ID_TeletexString
- #define ASN_ID_VideotexString 21
- #define ASN_ID_IA5String 22
- #define ASN_ID_UTCTime 23
- #define ASN_ID_GeneralTime 24
- #define ASN_ID_GraphicString 25
- #define ASN_ID_VisibleString 26
- #define ASN_ID_GeneralString 27
- #define ASN_ID_UniversalString 28
- #define ASN_ID_BMPString 30
- #define XM_SEEK 0x01
- #define XM_ADVANCE 0x02
- #define XM_DYNAMIC 0x04
- #define XM_SKIP 0x08
- #define XM_OPTIONAL 0x10
- #define ASN_K_MAXDEPTH 32
- #define ASN_K_MAXSUBIDS 128
- #define ASN_K_MAXENUM 100
- #define ASN_K_MAXERRP 5
- #define ASN_K_MAXERRSTK 8
- #define ASN_K_ENCBUFSIZ 16*1024
- #define ASN_K_MEMBUFSEG 1024
- #define OSRTINDENTSPACES 3
- #define ASN1_K_PLUS_INFINITY 0x40
- #define ASN1_K_MINUS_INFINITY 0x41
- #define REAL_BINARY 0x80
- #define REAL_SIGN 0x40
- #define REAL_EXPLEN_MASK 0x03
- #define REAL_EXPLEN_1 0x00
- #define REAL_EXPLEN_2 0x01
- #define REAL_EXPLEN_3 0x02
- #define REAL_EXPLEN_LONG 0x03
- #define REAL_FACTOR_MASK 0x0c
- #define REAL_BASE_MASK 0x30
- #define REAL_BASE_2 0x00
- #define REAL_BASE_8 0x10
- #define REAL_BASE_16 0x20
- #define REAL_ISO6093_MASK 0x3F
- #define ASN1REALMAX (ASN1REAL)DBL_MAX
- #define ASN1REALMIN (ASN1REAL)-DBL_MAX
- #define ASN1TAG_LSHIFT 24
- #define ASN1_K_CCBMaskSize 32
- #define ASN1_K_NumBitsPerMask 16
- #define ASN1_K_MaxSetElements (ASN1_K_CCBMaskSize*ASN1_K_NumBitsPerMask)
- #define XM_K_MEMBLKSIZ (4*1024)
- #define ASN1DYNCTXT 0x8000
- #define ASN1INDEFLEN 0x4000
- #define ASN1TRACE 0x2000
- #define ASN1LASTEOC 0x1000
- #define ASN1FASTCOPY 0x0800
- #define ASN1CONSTAG 0x0400
- #define ASN1CANXER 0x0200

- #define ASN1SAVEBUF 0x0100
- #define ASN1OPENTYPE 0x0080
- #define OSRTISSTREAM(pctx) ((pctx)->pStream != 0)
- #define OSDT_MAXSECFRAC 20
- #define ASN1MEMBUFPTR(pmb) ((pmb)->buffer + (pmb)->startidx)
- #define ASN1MEMBUFENDPTR(pmb) ((pmb)->buffer + (pmb)->startidx + (pmb)->usedcnt)
- #define ASN1MEMBUFUSEDSize(pmb) ((size_t)(pmb)->usedcnt)
- #define ASN1MBAPPENDSTR(pmb, str) rtMemBufAppend(pmb,(OSOCTET*)str,strlen(str))
- #define ASN1MBAPPENDUTF8(pmb, str) rtMemBufAppend(pmb,(OSOCTET*)str, rtxUTF8LenBytes(str))
- #define ASN1MAX(a, b) (((a)>(b))?(a):(b))
- #define ASN1MIN(a, b) (((a)<(b))?(a):(b))
- #define ALLOC_ASN1ARRAY(pctx, pseqof, type)
- #define ALLOC_ASN1ARRAY1(pctx, pseqof, type)
- #define ALLOC_ASN1ARRAY2(pctx, n, type)
- #define ALLOC_ASN1ELEM(pctx, type) (type*) rtMemHeapAllocZ (&(pctx)->pTypeMemHeap, sizeof(type))
- #define ASN1ARRAYSIZE(x) (sizeof(x)/sizeof(x[0]))
- #define ASN1MALLOC(pctx, nbytes) rtMemHeapAlloc(&(pctx)->pTypeMemHeap, nbytes)
- #define ASN1REALLOC(pctx, pmem, nbytes) rtMemHeapRealloc(&(pctx)->pTypeMemHeap, pmem, nbytes)
- #define REALLOC_ASN1ARRAY(pctx, pseqof, type)
- #define ASN1MEMFREE(pctx) rtMemHeapFreeAll(&(pctx)->pTypeMemHeap)
- #define ASN1MEMFREEPTR(pctx, pmem) rtMemHeapFreePtr(&(pctx)->pTypeMemHeap, (void*)pmem)
- #define ASN1MEMRESET(pctx) rtMemHeapReset(&(pctx)->pTypeMemHeap)
- #define ASN1BUF_CUR(cp) (cp)->buffer.data[(cp)->buffer.byteIndex]
- #define ASN1BUF_PTR(cp) &(cp)->buffer.data[(cp)->buffer.byteIndex]
- #define ASN1BUF_INDEFLEN(cp) (((cp)->flags&ASN1INDEFLEN)!=0)
- #define ASN1BUF_PTR ASN1BUF_PTR(cp)
- #define ASN1NUMOCTS(nbits) ((nbits>0)?((nbits-1)/8)+1):0)
- #define RTDIAG1(msg)
- #define RTDIAG2(msg, a)
- #define RTDIAG3(msg, a, b)
- #define RTDIAG4(msg, a, b, c)
- #define RTDIAG5(msg, a, b, c, d)
- #define RTDIAG6(msg, a, b, c, d, e)
- #define RTDIAG7(msg, a, b, c, d, e, f)
- #define RTDIAG8(msg, a, b, c, d, e, f, g)
- #define RTDIAG9(msg, a, b, c, d, e, f, g, h)
- #define RTDIAG10(msg, a, b, c, d, e, f, g, h, i)
- #define RTHXDUMP(buffer, numocts)
- #define RTDIAGSTRM2(pctx, msg)
- #define RTDIAGSTRM3(pctx, msg, a)
- #define RTDIAGSTRM4(pctx, msg, a, b)
- #define RTDIAGSTRM5(pctx, msg, a, b, c)
- #define RTDIAGSTRM6(pctx, msg, a, b, c, d)
- #define RTDIAGSTRM7(pctx, msg, a, b, c, d, e)
- #define RTDIAGSTRM8(pctx, msg, a, b, c, d, e, f)
- #define RTDIAGSTRM9(pctx, msg, a, b, c, d, e, f, g)
- #define RTDIAGSTRM10(pctx, msg, a, b, c, d, e, f, g, h)
- #define RTDIAGSTRM11(pctx, msg, a, b, c, d, e, f, g, h, i)
- #define RTHXDUMPSTRM(pctx, buffer, numocts)

- #define **HEXCHARTONIBBLE**(ch, b)
- #define **NIBBLETOHEXCHAR**(b, ch)
- #define **EXTERN_C** extern
- #define **EXTERN**
- #define **ASN1CRTMALLOC0**(nbytes) malloc(nbytes)
- #define **ASN1CRTFREE0**(ptr) free(ptr)
- #define **ASN1CRTMALLOC** ASN1MALLOC
- #define **ASN1CRTFREE** ASN1MEMFREEPTR
- #define **rtDListFastInit**(pList)
- #define **OSSETBIT**(bitStr, bitIndex) rtSetBit (bitStr.data, bitStr.numbits, bitIndex)
- #define **OSSETBITP**(pBitStr, bitIndex) rtSetBit ((pBitStr)->data, (pBitStr)->numbits, bitIndex)
- #define **OSCLEARBIT**(bitStr, bitIndex) rtClearBit (bitStr.data, bitStr.numbits, bitIndex)
- #define **OSCLEARBITP**(pBitStr, bitIndex) rtClearBit ((pBitStr)->data, (pBitStr)->numbits, bitIndex)
- #define **OSTESTBIT**(bitStr, bitIndex) rtTestBit (bitStr.data, bitStr.numbits, bitIndex)
- #define **OSTESTBITP**(pBitStr, bitIndex) rtTestBit ((pBitStr)->data, (pBitStr)->numbits, bitIndex)
- #define **rtSetStrToBigInt** rtBigIntSetStr
- #define **rtSetInt64ToBigInt** rtBigIntSetInt64
- #define **rtSetBytesToBigInt** rtBigIntSetBytes
- #define **rtGetBigIntLen** rtBigIntGetDataLen
- #define **rtGetBigInt** rtBigIntGetData
- #define **rtPrintBigInt** rtBigIntPrint
- #define **rtCompareBigInt** rtBigIntCompare
- #define **rtCompareBigIntStr** rtBigIntStrCompare

Typedefs

- typedef OSUINT32 **ASN1TAG**
- typedef void * **ASN1ANY**
- typedef Asn1Object **ASN1Object**
- typedef const char * **ASN1GeneralizedTime**
- typedef const char * **ASN1GeneralString**
- typedef const char * **ASN1GraphicString**
- typedef const char * **ASN1IA5String**
- typedef const char * **ASN1ISO646String**
- typedef const char * **ASN1NumericString**
- typedef const char * **ASN1ObjectDescriptor**
- typedef const char * **ASN1PrintableString**
- typedef const char * **ASN1TeletexString**
- typedef const char * **ASN1T61String**
- typedef const char * **ASN1UTCTime**
- typedef const char * **ASN1UTF8String**
- typedef const char * **ASN1VideotexString**
- typedef const char * **ASN1VisibleString**
- typedef Asn116BitCharString **ASN1BMPString**
- typedef Asn132BitCharString **ASN1UniversalString**
- typedef ASN1BigInt **ASN1BigInt**
- typedef _Asn1RTSListNode **Asn1RTSListNode**
- typedef _Asn1RTSList **Asn1RTSList**
- typedef _Asn1RTDListNode **Asn1RTDListNode**
- typedef _Asn1RTDList **Asn1RTDList**
- typedef _Asn1RTStack **Asn1RTStack**
- typedef _Asn1SizeCnst **Asn1SizeCnst**
- typedef ASN1CTXT **ASN1CTXT**

- typedef ASN1MemBuf **ASN1MemBuf**
- typedef _OSDATETIME **OSDATETIME**
- typedef int(* **ASN1DumpCbFunc**)(const char *text_p, void *cbArg_p)

Enumerations

- enum **ASN1TagType** { **ASN1IMPL**, **ASN1EXPL** }
- enum **ASN1StrType** { **ASN1HEX**, **ASN1BIN**, **ASN1CHR** }
- enum **ASN1ActionType** { **ASN1ENCODE**, **ASN1DECODE** }

Functions

- long **rtBase64EncodeData** (ASN1CTXT *pctx, const OSOCTET *pSrcData, size_t srcDataSize, OSOCTET **ppDstData)
- long **rtBase64DecodeData** (ASN1CTXT *pctx, const OSOCTET *pSrcData, size_t srcDataSize, OSOCTET **ppDstData)
- int **rtCheckBuffer** (ASN1CTXT *pctx, size_t nbytes)
- void **rtCopyContext** (ASN1CTXT *pdest, ASN1CTXT *psrc)
- void **rtXchgContext** (ASN1CTXT *pdest, ASN1CTXT *psrc)
- void **rtCtxtSetFlag** (ASN1CTXT *pctx, OSUINT16 mask)
- void **rtCtxtClearFlag** (ASN1CTXT *pctx, OSUINT16 mask)
- int **rtInitContextBuffer** (ASN1CTXT *ctx_p, const OSOCTET *bufaddr, size_t bufsiz)
- int **rtInitContextUsingKey** (ASN1CTXT *ctx_p, const OSOCTET *key, size_t keylen)
- int **rtInitContext** (ASN1CTXT *pctx)
- int **rtInitSubContext** (ASN1CTXT *ctx_p, ASN1CTXT *psrc)
- int **rtPreInitContext** (ASN1CTXT *ctx_p)
- ASN1CTXT * **rtNewContext** (void)
- ASN1CTXT * **rtNewContextUsingKey** (const OSOCTET *key, size_t keylen)
- void **rtFreeContext** (ASN1CTXT *ctx_p)
- int **rtReadNextByte** (ASN1CTXT *pctx)
- void **rtSetCopyValues** (ASN1CTXT *ctx_p, OSBOOL value)
- void **rtSetFastCopy** (ASN1CTXT *ctx_p, OSBOOL value)
- Asn1RTDListNode * **rtDListAppend** (ASN1CTXT *ctx_p, Asn1RTDList *pList, void *pData)
- void **rtDListInit** (Asn1RTDList *pList)
- Asn1RTDListNode * **rtDListInsert** (ASN1CTXT *ctx_p, Asn1RTDList *pList, int index, void *pData)
- Asn1RTDListNode * **rtDListInsertBefore** (ASN1CTXT *ctx_p, Asn1RTDList *pList, Asn1RTDListNode *node, void *pData)
- Asn1RTDListNode * **rtDListInsertAfter** (ASN1CTXT *ctx_p, Asn1RTDList *pList, Asn1RTDListNode *node, void *pData)
- Asn1RTDListNode * **rtDListFindByIndex** (Asn1RTDList *pList, int index)
- Asn1RTDListNode * **rtDListFindByData** (Asn1RTDList *pList, void *data)
- int **rtDListFindIndexByData** (Asn1RTDList *pList, void *data)
- void **rtDListRemove** (Asn1RTDList *pList, Asn1RTDListNode *node)
- void **rtDListFreeNodes** (ASN1CTXT *ctx_p, Asn1RTDList *pList)
- void **rtDListFreeAll** (ASN1CTXT *ctx_p, Asn1RTDList *pList)
- int **rtDListToArray** (ASN1CTXT *ctx_p, Asn1RTDList *pList, void **ppArray, OSUINT32 *pElements, int elemSize)
- int **rtDListAppendArray** (ASN1CTXT *ctx_p, Asn1RTDList *pList, const void *pArray, OSUINT32 numElements, int elemSize)
- int **rtDListAppendArrayCopy** (ASN1CTXT *ctx_p, Asn1RTDList *pList, const void *pArray, OSUINT32 numElements, int elemSize)
- Asn1RTDListNode * **rtDListAppendNode** (ASN1CTXT *ctx_p, Asn1RTDList *pList, void *pData)
- void **rtdiag** (const char *fmtspec,...)
- void **rtDiagStream** (ASN1CTXT *pctx, const char *fmtspec,...)

- void **rtDiagStreamHexDump** (ASN1CTXT *pctxt, const OSOCTET *data, OSUINT32 numocts)
- void **rtDiagHexDump** (const OSOCTET *data, OSUINT32 numocts)
- void **rtSetDiag** (int value)
- void **rtHexDumpEx** (const OSOCTET *data, OSUINT32 numocts, int bytesPerUnit)
- void **rtHexDumpToFile** (FILE *fp, const OSOCTET *data, OSUINT32 numocts)
- void **rtHexDumpToFileEx** (FILE *fp, const OSOCTET *data, OSUINT32 numocts, int bytesPerUnit)
- int **rtHexDumpToString** (const OSOCTET *data, OSUINT32 numocts, char *buffer, int bufferIndex, int bufferSize)
- int **rtHexDumpToStringEx** (const OSOCTET *data, OSUINT32 numocts, char *buffer, int bufferIndex, int bufferSize, int bytesPerUnit)
- int **rtHexDumpToStream** (ASN1CTXT *pctxt, const OSOCTET *data, OSUINT32 numocts)
- int **rtHexDumpToStreamEx** (ASN1CTXT *pctxt, const OSOCTET *data, OSUINT32 numocts, int bytesPerUnit)
- int **rtErrAddIntParm** (ASN1ErrInfo *pErrInfo, int errParm)
- int **rtErrAddStrParm** (ASN1ErrInfo *pErrInfo, const char *errprm_p)
- int **rtErrAddTagParm** (ASN1ErrInfo *pErrInfo, ASN1TAG errParm)
- int **rtErrAddUIntParm** (ASN1ErrInfo *pErrInfo, unsigned int errParm)
- int **rtErrCopyData** (ASN1ErrInfo *pSrcErrInfo, ASN1ErrInfo *pDestErrInfo)
- void **rtErrFreeParms** (ASN1ErrInfo *pErrInfo)
- char * **rtErrFmtMsg** (ASN1ErrInfo *pErrInfo, char *bufp)
- char * **rtErrGetText** (ASN1CTXT *ctxt_p)
- void **rtErrLogUsingCB** (ASN1ErrInfo *pErrInfo, ASN1DumpCbFunc cb, void *cbArg)
- void **rtErrPrint** (ASN1ErrInfo *pErrInfo)
- int **rtErrReset** (ASN1ErrInfo *pErrInfo)
- int **rtErrSetData** (ASN1ErrInfo *pErrInfo, int status, const char *module, int lno)
- int **rtGetIdentByteCount** (OSUINT32 ident)
- int **rtGetIdent64ByteCount** (OSUINT64 ident)
- unsigned int **rtIntByteCount** (OSINT32 val)
- int **rtOctetBitLen** (OSOCTET w)
- void **rtMemBufInit** (ASN1CTXT *pCtxt, ASN1MemBuf *pMemBuf, OSUINT32 segsize)
- void **rtMemBufInitBuffer** (ASN1CTXT *pCtxt, ASN1MemBuf *pMemBuf, OSOCTET *buf, OSUINT32 bufsize, OSUINT32 segsize)
- void **rtMemBufFree** (ASN1MemBuf *pMemBuf)
- int **rtMemBufAppend** (ASN1MemBuf *pMemBuf, OSOCTET *pdata, OSUINT32 nbytes)
- int **rtMemBufCut** (ASN1MemBuf *pMemBuf, OSUINT32 fromOffset, OSUINT32 nbytes)
- void **rtMemBufReset** (ASN1MemBuf *pMemBuf)
- int **rtMemBufTrimW** (ASN1MemBuf *pMemBuf)
- int **rtMemBufSet** (ASN1MemBuf *pMemBuf, OSOCTET value, OSUINT32 nbytes)
- OSOCTET * **rtMemBufGetData** (ASN1MemBuf *pMemBuf, int *length)
- int **rtMemBufGetDataLen** (ASN1MemBuf *pMemBuf)
- int **rtMemBufPreAllocate** (ASN1MemBuf *pMemBuf, OSUINT32 nbytes)
- OSBOOL **rtMemBufSetExpandable** (ASN1MemBuf *pMemBuf, OSBOOL isExpandable)
- void **rtSetOID** (ASN1OBJID *ptarget, ASN1OBJID *psource)
- void **rtAddOID** (ASN1OBJID *ptarget, ASN1OBJID *psource)
- int **rtSetBit** (OSOCTET *pBits, int numbits, int bitIndex)
- int **rtClearBit** (OSOCTET *pBits, int numbits, int bitIndex)
- OSBOOL **rtTestBit** (const OSOCTET *pBits, int numbits, int bitIndex)
- void **rtSListInit** (Asn1RTSList *pList)
- void **rtSListInitEx** (ASN1CTXT *pctxt, Asn1RTSList *pList)
- void **rtSListFree** (Asn1RTSList *pList)
- Asn1RTSList * **rtSListCreate** (void)
- Asn1RTSList * **rtSListCreateEx** (ASN1CTXT *pctxt)
- Asn1RTSListNode * **rtSListAppend** (Asn1RTSList *pList, void *pData)

- OSBOOL **rtSListFind** (Asn1RTSList *pList, void *pData)
- void **rtSListRemove** (Asn1RTSList *pList, void *pData)
- Asn1RTStack * **rtStackCreate** (void)
- Asn1RTStack * **rtStackCreateEx** (ASN1CTXT *pctx)
- void **rtStackInit** (Asn1RTStack *pStack)
- void * **rtStackPop** (Asn1RTStack *pStack)
- int **rtStackPush** (Asn1RTStack *pStack, void *pData)
- int **rtWriteBytes** (ASN1CTXT *pctx, const OSOCTET *pdata, size_t nocts)
- OSREAL **rtGetPlusInfinity** (void)
- OSREAL **rtGetMinusInfinity** (void)
- OSREAL **rtGetMinusZero** (void)
- OSREAL **rtGetNaN** (void)
- OSBOOL **rtIsMinusInfinity** (OSREAL value)
- OSBOOL **rtIsMinusZero** (OSREAL value)
- OSBOOL **rtIsNaN** (OSREAL value)
- OSBOOL **rtIsPlusInfinity** (OSREAL value)
- const char * **rtBitStrToString** (OSUINT32 numbits, const OSOCTET *data, char *buffer, size_t bufsiz)
- const char * **rtBoolToString** (OSBOOL value)
- const char * **rtIntToString** (OSINT32 value, char *buffer, size_t bufsiz)
- const char * **rtInt64ToString** (OSINT64 value, char *buffer, size_t bufsiz)
- const char * **rtUIntToString** (OSUINT32 value, char *buffer, size_t bufsiz)
- const char * **rtUInt64ToString** (OSUINT64 value, char *buffer, size_t bufsiz)
- const char * **rtOIDToString** (OSUINT32 numids, OSUINT32 *data, char *buffer, size_t bufsiz)
- const char * **rtOID64ToString** (OSUINT32 numids, OSUINT64 *data, char *buffer, size_t bufsiz)
- const char * **rtOctStrToString** (OSUINT32 numocts, const OSOCTET *data, char *buffer, size_t bufsiz)
- const char * **rtTagToString** (ASN1TAG tag, char *buffer, size_t bufsiz)
- const char * **rtBCDToString** (OSUINT32 numocts, const OSOCTET *data, char *buffer, size_t bufsiz)
- int **rtStringToBCD** (const char *str, OSOCTET *bcdStr, size_t bufsiz)
- int **rtStringToDynBCD** (ASN1CTXT *pctx, const char *str, ASN1DynOctStr *pctxstr)
- const char * **rtBMPToCString** (ASN1BMPString *pBMPString, char *cstring, OSUINT32 cstrsize)
- const char * **rtBMPToNewCString** (ASN1BMPString *pBMPString)
- const char * **rtBMPToNewCStringEx** (ASN1CTXT *pctx, ASN1BMPString *pBMPString)
- ASN1BMPString * **rtCToBMPString** (ASN1CTXT *ctx_p, const char *cstring, ASN1BMPString *pBMPString, Asn116BitCharSet *pCharSet)
- OSBOOL **rtIsIn16BitCharSet** (OSUNICHAR ch, Asn116BitCharSet *pCharSet)
- const char * **rtUCSToCString** (ASN1UniversalString *pUCSString, char *cstring, OSUINT32 cstrsize)
- const char * **rtUCSToNewCString** (ASN1UniversalString *pUCSString)
- const char * **rtUCSToNewCStringEx** (ASN1CTXT *pctx, ASN1UniversalString *pUCSString)
- ASN1UniversalString * **rtCToUCSString** (ASN1CTXT *ctx_p, const char *cstring, ASN1UniversalString *pUCSString, Asn132BitCharSet *pCharSet)
- OSBOOL **rtIsIn32BitCharSet** (OS32BITCHAR ch, Asn132BitCharSet *pCharSet)
- wchar_t * **rtUCSToWCSSString** (ASN1UniversalString *pUCSString, wchar_t *wcstring, OSUINT32 wcstrsize)
- ASN1UniversalString * **rtWCSToUCSString** (ASN1CTXT *ctx_p, wchar_t *wcstring, ASN1UniversalString *pUCSString, Asn132BitCharSet *pCharSet)
- int **rtUTF8ToWCS** (ASN1CTXT *pctx, ASN1UTF8String inbuf, wchar_t *outbuf, size_t outbufsiz)
- long **rtWCSToUTF8** (ASN1CTXT *pctx, wchar_t *inbuf, size_t inlen, OSOCTET *outbuf, size_t outbufsiz)
- int **rtValidateUTF8** (ASN1CTXT *pctx, ASN1UTF8String inbuf)

- int **rtUTF8Len** (ASN1UTF8String inbuf)
- int **rtUTF8LenBytes** (ASN1UTF8String inbuf)
- int **rtUTF8CharSize** (OS32BITCHAR wc)
- int **rtUTF8EncodeChar** (OS32BITCHAR wc, OSOCTET *buf, int bufsiz)
- int **rtUTF8DecodeChar** (ASN1CTXT *pCtxt, ASN1UTF8String pinbuf, int *pInsize)
- char * **rtUTF8Strdup** (ASN1CTXT *pctxt, ASN1UTF8String utf8str)
- char * **rtUTF8Strndup** (ASN1CTXT *pctxt, ASN1UTF8String utf8str, int nbytes)
- void **rtBigIntInit** (ASN1BigInt *pInt)
- int **rtBigIntSetStr** (ASN1CTXT *pCtxt, ASN1BigInt *pInt, const char *value, int radix)
- int **rtBigIntSetInt64** (ASN1CTXT *pCtxt, ASN1BigInt *pInt, OSINT64 value)
- int **rtBigIntSetBytes** (ASN1CTXT *pCtxt, ASN1BigInt *pInt, OSOCTET *value, size_t vallen)
- size_t **rtBigIntGetDataLen** (ASN1BigInt *pInt)
- long **rtBigIntGetData** (ASN1CTXT *pCtxt, ASN1BigInt *pInt, OSOCTET *buffer, size_t bufSize)
- int **rtBigIntDigitsNum** (ASN1BigInt *pInt, int radix)
- int **rtBigIntCopy** (ASN1CTXT *pCtxt, const ASN1BigInt *pSrc, ASN1BigInt *pDst)
- int **rtBigIntFastCopy** (ASN1CTXT *pCtxt, ASN1BigInt *pSrc, ASN1BigInt *pDst)
- int **rtBigIntToString** (ASN1CTXT *pCtxt, ASN1BigInt *pInt, int radix, char *str, size_t strSize)
- int **rtBigIntPrint** (const char *name, ASN1BigInt *bigint, int radix)
- int **rtBigIntCompare** (const ASN1BigInt *arg1, const ASN1BigInt *arg2)
- int **rtBigIntStrCompare** (ASN1CTXT *pCtxt, const char *arg1, const char *arg2)
- void **rtBigIntFree** (ASN1CTXT *pCtxt, ASN1BigInt *pInt)
- int **rtSetLocalTime** (OSDATETIME *dateTime, time_t time, OSBOOL diffTime)
- int **rtMakeGeneralizedTime** (ASN1CTXT *pctxt, const OSDATETIME *dateTime, char **outdata, int outdataSize)
- int **rtMakeUTCTime** (ASN1CTXT *pctxt, const OSDATETIME *dateTime, char **outdata, int outdataSize)
- int **rtMakeXmlDateTime** (ASN1CTXT *pctxt, const OSDATETIME *dateTime, char **outdata, int outdataSize)
- int **rtParseGeneralizedTime** (ASN1CTXT *pctxt, const char *value, OSDATETIME *dateTime)
- int **rtParseUTCTime** (ASN1CTXT *pctxt, const char *value, OSDATETIME *dateTime)
- int **rtParseXmlDateTime** (ASN1CTXT *pctxt, const char *value, OSDATETIME *dateTime)
- int **rtFileReadBinary** (ASN1CTXT *pctxt, const char *filePath, OSOCTET **ppMsgBuf, size_t *pLength)
- int **rtGetLibVersion** ()
- const char * **rtGetLibInfo** ()

rtCompare.h File Reference

Detailed Description

Functions for comparing the values of primitive ASN.1 types.

```
#include "asn1type.h"
```

Functions

- ASN1BOOL **rtCmpBoolean** (ASN1ConstCharPtr name, ASN1BOOL value, ASN1BOOL compValue, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpInt8** (ASN1ConstCharPtr name, ASN1INT8 value, ASN1INT8 compValue, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpSInt** (ASN1ConstCharPtr name, ASN1SINT value, ASN1SINT compValue, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpUInt8** (ASN1ConstCharPtr name, ASN1UINT8 value, ASN1UINT8 compValue, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpUSInt** (ASN1ConstCharPtr name, ASN1USINT value, ASN1USINT compValue, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpInteger** (ASN1ConstCharPtr name, ASN1INT value, ASN1INT compValue, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpUnsigned** (ASN1ConstCharPtr name, ASN1UINT value, ASN1UINT compValue, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpInt64** (ASN1ConstCharPtr name, ASN1INT64 value, ASN1INT64 compValue, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpUInt64** (ASN1ConstCharPtr name, ASN1UINT64 value, ASN1UINT64 compValue, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpBitStr** (ASN1ConstCharPtr name, ASN1UINT numbits, ASN1ConstOctetPtr data, ASN1UINT compNumbits, ASN1ConstOctetPtr compData, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpOctStr** (ASN1ConstCharPtr name, ASN1UINT numocts, ASN1ConstOctetPtr data, ASN1UINT compNumocts, ASN1ConstOctetPtr compData, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpCharStr** (ASN1ConstCharPtr name, ASN1ConstCharPtr cstring, ASN1ConstCharPtr compCstring, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmp16BitCharStr** (ASN1ConstCharPtr name, Asn116BitCharString *bstring, Asn116BitCharString *compBstring, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmp32BitCharStr** (ASN1ConstCharPtr name, Asn132BitCharString *bstring, Asn132BitCharString *compBstring, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpReal** (ASN1ConstCharPtr name, ASN1REAL value, ASN1REAL compValue, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpOID** (ASN1ConstCharPtr name, ASN1OBJID *pOID, ASN1OBJID *pcompOID, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpOIDValue** (ASN1ConstCharPtr name, ASN1OBJID *pOID, ASN1OBJID *pcompOID, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpOID64** (ASN1ConstCharPtr name, ASN1OID64 *pOID, ASN1OID64 *pcompOID, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpOID64Value** (ASN1ConstCharPtr name, ASN1OID64 *pOID, ASN1OID64 *pcompOID, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpOpenType** (ASN1ConstCharPtr name, ASN1UINT numocts, ASN1ConstOctetPtr data, ASN1UINT compNumocts, ASN1ConstOctetPtr compData, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpOpenTypeExt** (ASN1ConstCharPtr name, Asn1RTDList *pElemList, Asn1RTDList *pCompElemList, char *errBuff, int errBuffSize)

- ASN1BOOL **rtCmpTag** (ASN1ConstCharPtr name, int tag, int compTag, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpSeqOfElements** (ASN1ConstCharPtr name, int noOfElems, int compNoOfElems, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpOptional** (ASN1ConstCharPtr name, unsigned presentBit, unsigned compPresentBit, char *errBuff, int errBuffSize)
- ASN1BOOL **rtCmpToStdoutBoolean** (ASN1ConstCharPtr name, ASN1BOOL value, ASN1BOOL compValue)
- ASN1BOOL **rtCmpToStdoutInteger** (ASN1ConstCharPtr name, ASN1INT value, ASN1INT compValue)
- ASN1BOOL **rtCmpToStdoutInt64** (ASN1ConstCharPtr name, ASN1INT64 value, ASN1INT64 compValue)
- ASN1BOOL **rtCmpToStdoutUnsigned** (ASN1ConstCharPtr name, ASN1UINT value, ASN1UINT compValue)
- ASN1BOOL **rtCmpToStdoutUInt64** (ASN1ConstCharPtr name, ASN1UINT64 value, ASN1UINT64 compValue)
- ASN1BOOL **rtCmpToStdoutBitStr** (ASN1ConstCharPtr name, ASN1UINT numbits, ASN1ConstOctetPtr data, ASN1UINT compNumbits, ASN1ConstOctetPtr compData)
- ASN1BOOL **rtCmpToStdoutOctStr** (ASN1ConstCharPtr name, ASN1UINT numocts, ASN1ConstOctetPtr data, ASN1UINT compNumocts, ASN1ConstOctetPtr compData)
- ASN1BOOL **rtCmpToStdoutCharStr** (ASN1ConstCharPtr name, ASN1ConstCharPtr cstring, ASN1ConstCharPtr compCString)
- ASN1BOOL **rtCmpToStdout16BitCharStr** (ASN1ConstCharPtr name, Asn116BitCharString *bstring, Asn116BitCharString *compBstring)
- ASN1BOOL **rtCmpToStdout32BitCharStr** (ASN1ConstCharPtr name, Asn132BitCharString *bstring, Asn132BitCharString *compBstring)
- ASN1BOOL **rtCmpToStdoutReal** (ASN1ConstCharPtr name, ASN1REAL value, ASN1REAL compValue)
- ASN1BOOL **rtCmpToStdoutOID** (ASN1ConstCharPtr name, ASN1OBJID *pOID, ASN1OBJID *pcompOID)
- ASN1BOOL **rtCmpToStdoutOIDValue** (ASN1ConstCharPtr name, ASN1OBJID *pOID, ASN1OBJID *pcompOID)
- ASN1BOOL **rtCmpToStdoutOID64** (ASN1ConstCharPtr name, ASN1OID64 *pOID, ASN1OID64 *pcompOID)
- ASN1BOOL **rtCmpToStdoutOID64Value** (ASN1ConstCharPtr name, ASN1OID64 *pOID, ASN1OID64 *pcompOID)
- ASN1BOOL **rtCmpToStdoutOpenType** (ASN1ConstCharPtr name, ASN1UINT numocts, ASN1ConstOctetPtr data, ASN1UINT compNumocts, ASN1ConstOctetPtr compData)
- ASN1BOOL **rtCmpToStdoutOpenTypeExt** (ASN1ConstCharPtr name, Asn1RTDList *pElemList, Asn1RTDList *pCompElemList)
- ASN1BOOL **rtCmpToStdoutTag** (ASN1ConstCharPtr name, int tag, int compTag)
- ASN1BOOL **rtCmpToStdoutSeqOfElements** (ASN1ConstCharPtr name, int noOfElems, int compNoOfElems)
- ASN1BOOL **rtCmpToStdoutOptional** (ASN1ConstCharPtr name, unsigned presentBit, unsigned compPresentBit)

rtCopy.h File Reference

Detailed Description

Functions for copying values of primitive ASN.1 types.

```
#include "asn1type.h"
```

Defines

- `#define RTCOPYCHARSTR(pctxt, src, dst) do { char* ptr; rtCopyCharStr (pctxt, src, &ptr); dst = ptr; } while(0)`

Functions

- `ASN1BOOL rtCopyBitStr (ASN1CTXT *pctxt, ASN1UINT srcNumbits, ASN1ConstOctetPtr pSrcData, ASN1UINT *pDstNumbits, ASN1OCTET *pDstData)`
- `ASN1BOOL rtCopyDynBitStr (ASN1CTXT *pctxt, ASN1DynBitStr *pSrcData, ASN1DynBitStr *pDstData)`
- `ASN1BOOL rtCopyOctStr (ASN1CTXT *pctxt, ASN1UINT srcNumocts, ASN1ConstOctetPtr pSrcData, ASN1UINT *pDstNumocts, ASN1OCTET *pDstData)`
- `ASN1BOOL rtCopyDynOctStr (ASN1CTXT *pctxt, ASN1DynOctStr *pSrcData, ASN1DynOctStr *pDstData)`
- `ASN1BOOL rtCopy16BitCharStr (ASN1CTXT *pctxt, Asn116BitCharString *srcStr, Asn116BitCharString *dstStr)`
- `ASN1BOOL rtCopy32BitCharStr (ASN1CTXT *pctxt, Asn132BitCharString *srcStr, Asn132BitCharString *dstStr)`
- `ASN1BOOL rtCopyOID (ASN1CTXT *pctxt, ASN1OBJID *srcOID, ASN1OBJID *dstOID)`
- `ASN1BOOL rtCopyOID64 (ASN1CTXT *pctxt, ASN1OID64 *srcOID, ASN1OID64 *dstOID)`
- `ASN1BOOL rtCopyOpenType (ASN1CTXT *pctxt, ASN1OpenType *srcOT, ASN1OpenType *dstOT)`
- `ASN1BOOL rtCopyOpenTypeExt (ASN1CTXT *pctxt, Asn1RTDList *srcList, Asn1RTDList *dstList)`

rtMemory.h File Reference

Detailed Description

Memory management function and macro definitions

```
#include "asn1type.h"
```

Defines

- `#define RT_MH_DONTKEEPFREE 0x1`
- `#define OSRTMH_PROPID_DEFBLKSIZE 1`
- `#define OSRTMH_PROPID_SETFLAGS 2`
- `#define OSRTMH_PROPID_CLEARFLAGS 3`
- `#define OSRTMH_PROPID_USER 10`
- `#define rtMemAlloc(pctx, nbytes) rtMemHeapAlloc(&(pctx)->pTypeMemHeap, nbytes)`
- `#define rtMemAllocType(pctx, ctype) (ctype*)rtMemHeapAlloc(&(pctx)->pTypeMemHeap, sizeof(ctype))`
- `#define rtMemAllocZ(pctx, nbytes) rtMemHeapAllocZ(&(pctx)->pTypeMemHeap, nbytes)`
- `#define rtMemAllocTypeZ(pctx, ctype) (ctype*)rtMemHeapAllocZ(&(pctx)->pTypeMemHeap, sizeof(ctype))`
- `#define rtMemRealloc(pctx, mem_p, nbytes) rtMemHeapRealloc(&(pctx)->pTypeMemHeap, (void*)mem_p, nbytes)`
- `#define rtMemFreePtr(pctx, mem_p)`
- `#define rtMemFree(pctx) rtMemHeapFreeAll(&(pctx)->pTypeMemHeap)`
- `#define rtMemReset(pctx) rtMemHeapReset(&(pctx)->pTypeMemHeap)`
- `#define OSCDECL`

Typedefs

- `typedef void *OSCDECL * OSMallocFunc (size_t size)`
- `typedef void *OSCDECL * OSReallocFunc (void *ptr, size_t size)`

Functions

- `typedef void (OSCDECL *OSFreeFunc)(void *ptr)`
- `void rtMemHeapAddRef (void **ppvMemHeap)`
- `void * rtMemHeapAlloc (void **ppvMemHeap, size_t nbytes)`
- `void * rtMemHeapAllocZ (void **ppvMemHeap, size_t nbytes)`
- `int rtMemHeapCheckPtr (void **ppvMemHeap, void *mem_p)`
- `int rtMemHeapCreate (void **ppvMemHeap)`
- `void rtMemHeapFreeAll (void **ppvMemHeap)`
- `void rtMemHeapFreePtr (void **ppvMemHeap, void *mem_p)`
- `void * rtMemHeapMarkSaved (void **ppvMemHeap, ASN1ConstVoidPtr mem_p, ASN1BOOL saved)`
- `void * rtMemHeapRealloc (void **ppvMemHeap, void *mem_p, size_t nbytes)`
- `void rtMemHeapRelease (void **ppvMemHeap)`
- `void rtMemHeapReset (void **ppvMemHeap)`
- `void rtMemHeapSetProperty (void **ppvMemHeap, ASN1UINT propId, void *pProp)`
- `void rtMemHeapSetAllocFuncs (OSMallocFunc malloc_func, OSReallocFunc realloc_func, OSFreeFunc free_func)`
- `void rtMemFreeOpenSeqExt (ASN1CTXT *pctx, Asn1RTDList *pElemList)`
- `void rtMemHeapSetFlags (ASN1CTXT *pctx, ASN1UINT flags)`
- `void rtMemHeapClearFlags (ASN1CTXT *pctx, ASN1UINT flags)`

- void **rtMemHeapSetDefBlkSize** (ASN1CTX *pctx, ASN1UINT blkSize)
- ASN1UINT **rtMemHeapGetDefBlkSize** (ASN1CTX *pctx)

rtPrint.h File Reference

Detailed Description

Functions to print the output in a "name=value" format.

```
#include <stdio.h>
#include "asn1type.h"
```

Functions

- void **rtPrintBoolean** (const char *name, ASN1BOOL value)
- void **rtPrintInteger** (const char *name, ASN1INT value)
- void **rtPrintInt64** (const char *name, ASN1INT64 value)
- void **rtPrintUnsigned** (const char *name, ASN1UINT value)
- void **rtPrintUInt64** (const char *name, ASN1UINT64 value)
- void **rtPrintBitStr** (const char *name, ASN1UINT numbits, ASN1ConstOctetPtr data, const char *conn)
- void **rtPrintBitStrBraceText** (const char *name, OSUINT32 numbits, const OSOCTET *data)
- void **rtPrintOctStr** (const char *name, ASN1UINT numocts, ASN1ConstOctetPtr data, const char *conn)
- void **rtPrintHexStr** (const char *name, OSUINT32 numocts, const OSOCTET *data)
- void **rtPrintCharStr** (const char *name, const char *cstring)
- void **rtPrint16BitCharStr** (const char *name, Asn116BitCharString *bstring, const char *conn)
- void **rtPrintUnicodeCharStr** (const char *name, Asn116BitCharString *bstring)
- void **rtPrint32BitCharStr** (const char *name, Asn132BitCharString *bstring, const char *conn)
- void **rtPrintUnivCharStr** (const char *name, Asn132BitCharString *bstring)
- void **rtPrintReal** (const char *name, ASN1REAL value)
- void **rtPrintOID** (const char *name, ASN1OBJID *pOID)
- void **rtPrintOIDValue** (ASN1OBJID *pOID)
- void **rtPrintOID64** (const char *name, ASN1OID64 *pOID)
- void **rtPrintOID64Value** (ASN1OID64 *pOID)
- void **rtPrintOpenType** (const char *name, ASN1UINT numocts, ASN1ConstOctetPtr data, const char *conn)
- void **rtPrintOpenTypeExt** (const char *name, Asn1RTDList *pElemList)
- void **rtPrintOpenTypeExtBraceText** (const char *name, Asn1RTDList *pElemList)
- void **rtPrintIndent** ()
- void **rtPrintIncrIndent** ()
- void **rtPrintDecrIndent** ()
- void **rtPrintCloseBrace** ()
- void **rtPrintOpenBrace** (const char *)

rtPrintStream.h File Reference

Detailed Description

Common runtime constants, data structure definitions, and run-time functions to support the streaming operations.

```
#include <stdarg.h>
#include "asn1type.h"
#include "asn1intl.h"
```

Classes

- struct **OSRTPrintStream**

Typedefs

- typedef void(* **rtPrintCallback**)(void *pPrntStrmInfo, ASN1ConstCharPtr fmts-spec, va_list arglist)
- typedef **OSRTPrintStream OSRTPrintStream**

Functions

- int **rtSetPrintStream** (ASN1CTXT *pctx, rtPrintCallback myCallback, void *pStrmInfo)
- int **rtSetGlobalPrintStream** (rtPrintCallback myCallback, void *pStrmInfo)
- int **rtPrintToStream** (ASN1CTXT *pctx, ASN1ConstCharPtr fmts-spec,...)
- int **rtDiagToStream** (ASN1CTXT *pctx, ASN1ConstCharPtr fmts-spec, va_list arglist)
- int **rtPrintStreamRelease** (ASN1CTXT *pctx)

Variables

- **OSRTPrintStream g_PrintStream**

rtPrintToStream.h File Reference

Detailed Description

Functions to print the output in a "name=value" format.

```
#include <stdio.h>
#include "asn1type.h"
```

Functions

- int **rtPrintToStreamBoolean** (ASN1CTXT *pctxt, ASN1ConstCharPtr name, ASN1BOOL value)
- int **rtPrintToStreamInteger** (ASN1CTXT *pctxt, ASN1ConstCharPtr name, ASN1INT value)
- int **rtPrintToStreamInt64** (ASN1CTXT *pctxt, ASN1ConstCharPtr name, ASN1INT64 value)
- int **rtPrintToStreamUnsigned** (ASN1CTXT *pctxt, ASN1ConstCharPtr name, ASN1UINT value)
- int **rtPrintToStreamUInt64** (ASN1CTXT *pctxt, ASN1ConstCharPtr name, ASN1UINT64 value)
- int **rtPrintToStreamBitStr** (ASN1CTXT *pctxt, ASN1ConstCharPtr name, ASN1UINT numbits, ASN1ConstOctetPtr data, ASN1ConstCharPtr conn)
- int **rtPrintToStreamBitStrBraceText** (ASN1CTXT *pctxt, const char *name, OSUINT32 numbits, const OSOCTET *data)
- int **rtPrintToStreamOctStr** (ASN1CTXT *pctxt, ASN1ConstCharPtr name, ASN1UINT numocts, ASN1ConstOctetPtr data, ASN1ConstCharPtr conn)
- int **rtPrintToStreamCharStr** (ASN1CTXT *pctxt, ASN1ConstCharPtr name, ASN1ConstCharPtr cstring)
- int **rtPrintToStream16BitCharStr** (ASN1CTXT *pctxt, ASN1ConstCharPtr name, Asn116BitCharString *bstring, ASN1ConstCharPtr conn)
- int **rtPrintToStream32BitCharStr** (ASN1CTXT *pctxt, ASN1ConstCharPtr name, Asn132BitCharString *bstring, ASN1ConstCharPtr conn)
- int **rtPrintToStreamReal** (ASN1CTXT *pctxt, ASN1ConstCharPtr name, ASN1REAL value)
- int **rtPrintToStreamOID** (ASN1CTXT *pctxt, ASN1ConstCharPtr name, ASN1OBJID *pOID)
- int **rtPrintToStreamOIDValue** (ASN1CTXT *pctxt, ASN1OBJID *pOID)
- int **rtPrintToStreamOID64** (ASN1CTXT *pctxt, ASN1ConstCharPtr name, ASN1OID64 *pOID)
- int **rtPrintToStreamOID64Value** (ASN1CTXT *pctxt, ASN1OID64 *pOID)
- int **rtPrintToStreamOpenType** (ASN1CTXT *pctxt, ASN1ConstCharPtr name, ASN1UINT numocts, ASN1ConstOctetPtr data, ASN1ConstCharPtr conn)
- int **rtPrintToStreamOpenTypeExt** (ASN1CTXT *pctxt, ASN1ConstCharPtr name, Asn1RTDList *pElemList)
- int **rtPrintToStreamUnivCharStr** (ASN1CTXT *pctxt, const char *name, Asn132BitCharString *bstring)
- int **rtPrintToStreamUnicodeCharStr** (ASN1CTXT *pctxt, const char *name, Asn116BitCharString *bstring)
- int **rtPrintToStreamHexStr** (ASN1CTXT *pctxt, const char *name, OSUINT32 numocts, const OSOCTET *data)
- int **rtPrintToStreamOpenTypeExtBraceText** (ASN1CTXT *pctxt, const char *name, Asn1RTDList *pElemList)
- int **rtPrintToStreamIndent** (ASN1CTXT *pctxt)
- void **rtPrintToStreamIncrIndent** ()
- void **rtPrintToStreamDecrIndent** ()
- int **rtPrintToStreamCloseBrace** (ASN1CTXT *pctxt)
- int **rtPrintToStreamOpenBrace** (ASN1CTXT *pctxt, const char *name)

rtPrintToString.h File Reference

Detailed Description

```
#include <stdio.h>
#include "asn1type.h"
```

Functions

- `int rtPrintToStringBoolean` (ASN1ConstCharPtr name, ASN1BOOL value, char *buffer, int bufferSize)
- `int rtPrintToStringInteger` (ASN1ConstCharPtr name, ASN1INT value, char *buffer, int bufferSize)
- `int rtPrintToStringInt64` (ASN1ConstCharPtr name, ASN1INT64 value, char *buffer, int bufferSize)
- `int rtPrintToStringUnsigned` (ASN1ConstCharPtr name, ASN1UINT value, char *buffer, int bufferSize)
- `int rtPrintToStringUInt64` (ASN1ConstCharPtr name, ASN1UINT64 value, char *buffer, int bufferSize)
- `int rtPrintToStringBitStr` (ASN1ConstCharPtr name, ASN1UINT numbits, ASN1ConstOctetPtr data, ASN1ConstCharPtr conn, char *buffer, int bufferSize)
- `int rtPrintToStringBitStrBraceText` (const char *name, OSUINT32 numbits, const OSOCTET *data, char *buffer, int bufferSize)
- `int rtPrintToStringOctStr` (ASN1ConstCharPtr name, ASN1UINT numocts, ASN1ConstOctetPtr data, ASN1ConstCharPtr conn, char *buffer, int bufferSize)
- `int rtPrintToStringCharStr` (ASN1ConstCharPtr name, ASN1ConstCharPtr cstring, char *buffer, int bufferSize)
- `int rtPrintToString16BitCharStr` (ASN1ConstCharPtr name, Asn116BitCharString *bstring, ASN1ConstCharPtr conn, char *buffer, int bufferSize)
- `int rtPrintToString32BitCharStr` (ASN1ConstCharPtr name, Asn132BitCharString *bstring, ASN1ConstCharPtr conn, char *buffer, int bufferSize)
- `int rtPrintToStringReal` (ASN1ConstCharPtr name, ASN1REAL value, char *buffer, int bufferSize)
- `int rtPrintToStringOID` (ASN1ConstCharPtr name, ASN1OBJID *pOID, char *buffer, int bufferSize)
- `int rtPrintToStringOID64` (ASN1ConstCharPtr name, ASN1OID64 *pOID, char *buffer, int bufferSize)
- `int rtPrintToStringOIDValue` (ASN1OBJID *pOID, char *buffer, int bufferIndex, int bufferSize)
- `int rtPrintToStringOID64Value` (ASN1OID64 *pOID, char *buffer, int bufferIndex, int bufferSize)
- `int rtPrintToStringOpenType` (ASN1ConstCharPtr name, ASN1UINT numocts, ASN1ConstOctetPtr data, ASN1ConstCharPtr conn, char *buffer, int bufferSize)
- `int rtPrintToStringOpenTypeExt` (ASN1ConstCharPtr name, Asn1RTDList *pElemList, char *buffer, int bufferSize)
- `int rtPrintToString` (ASN1ConstCharPtr namebuf, char *buffer, int bufSize)
- `int rtPrintToStringHexStr` (const char *name, OSUINT32 numocts, const OSOCTET *data, char *buffer, int bufSize)
- `int rtPrintToStringUnicodeCharStr` (const char *name, Asn116BitCharString *bstring, char *buffer, int bufSize)
- `int rtPrintToStringUnivCharStr` (const char *name, Asn132BitCharString *bstring, char *buffer, int bufSize)
- `int rtPrintToStringOpenTypeExtBraceText` (const char *name, Asn1RTDList *pElemList, char *buffer, int bufSize)
- `int rtPrintToStringIndent` (char *buffer, int bufSize)
- `void rtPrintToStringIncrIndent` ()

- void **rtPrintToStringDecrIndent** ()
- int **rtPrintToStringCloseBrace** (char *buffer, int bufSize)
- int **rtPrintToStringOpenBrace** (const char *, char *buffer, int bufSize)

rtSocket.h File Reference

Detailed Description

Common runtime constants, data structure definitions, and run-time functions to support TCP/IP or UDP socket operations.

```
#include "asn1type.h"
```

Defines

- `#define OSRTSOCKET_INVALID ((OSRTSOCKET)-1)`
- `#define OSIPADDR_ANY ((OSIPADDR)0)`
- `#define OSIPADDR_LOCAL ((OSIPADDR)0x7f000001UL)`
- `#define OSIPADDR_INVALID ((OSIPADDR)0xffffffffUL)`

Typedefs

- `typedef int OSRTSOCKET`
- `typedef unsigned long OSIPADDR`

Functions

- `int rtSocketAccept (OSRTSOCKET socket, OSRTSOCKET *pNewSocket, OSIPADDR *destAddr, int *destPort)`
- `int rtSocketAddrToStr (OSIPADDR ipAddr, char *pbuf, int bufsize)`
- `int rtSocketBind (OSRTSOCKET socket, OSIPADDR addr, int port)`
- `int rtSocketBlockingRead (OSRTSOCKET socket, ASN1OCTET *pbuf, size_t readBytes)`
- `int rtSocketClose (OSRTSOCKET socket)`
- `int rtSocketConnect (OSRTSOCKET socket, const char *host, int port)`
- `int rtSocketCreate (OSRTSOCKET *psocket)`
- `int rtSocketCreateUDP (OSRTSOCKET *psocket)`
- `int rtSocketsInit (void)`
- `int rtSocketsCleanup (void)`
- `int rtSocketListen (OSRTSOCKET socket, int maxConnection)`
- `int rtSocketRecv (OSRTSOCKET socket, ASN1OCTET *pbuf, ASN1UINT bufsize)`
- `int rtSocketRecvFrom (OSRTSOCKET socket, ASN1OCTET *pbuf, ASN1UINT bufsize, char *remotehost, int *remoteport)`
- `int rtSocketSend (OSRTSOCKET socket, const ASN1OCTET *pdata, ASN1UINT size)`
- `int rtSocketSendTo (OSRTSOCKET socket, const ASN1OCTET *pdata, ASN1UINT size, const char *remotehost, int remoteport)`
- `int rtSocketStrToAddr (const char *pIPAddrStr, OSIPADDR *pIPAddr)`
- `int rtSocketGetLocalIPAddress (char *pIPAddr, int bufSize)`

Typedef Documentation

`typedef unsigned long OSIPADDR`

The IP address represented as unsigned long value. The most significant 8 bits in this unsigned long value represent the first number of the IP address. The least significant 8 bits represent the last number of the IP address.

```
typedef int OSRTSOCKET
```

```
Socket's handle
```

rtStream.h File Reference

Detailed Description

Common runtime constants, data structure definitions, and run-time functions to support the streaming operations.

```
#include "asn1type.h"
#include "asn1int1.h"
#include "rtSocket.h"
```

Classes

- struct **OSRTStream**

Defines

- #define **OSRTSTRMF_INPUT** 0x0001
- #define **OSRTSTRMF_OUTPUT** 0x0002
- #define **OSRTSTRMF_REQBUFFER** 0x4000
- #define **OSRTSTRMF_BUFFERED** 0x8000
- #define **OSRTSTRMF_BUF_INPUT** (OSRTSTRMF_INPUT|OSRTSTRMF_BUFFERED)
- #define **OSRTSTRMF_BUF_OUTPUT** (OSRTSTRMF_OUTPUT|OSRTSTRMF_BUFFERED)
- #define **OSRTSTRMID_FILE** 1
- #define **OSRTSTRMID_SOCKET** 2
- #define **OSRTSTRMID_MEMORY** 3
- #define **OSRTSTRMID_BUFFERED** 4
- #define **OSRTSTRMID_DIRECTBUF** 5
- #define **OSRTSTRMID_USER** 1000
- #define **OSRTSTRM_K_BUFSIZE** 1024
- #define **OSRTSTRM_K_INVALIDMARK** ((ASN1UINT)-1)
- #define **OSRTSTREAM_BYTEINDEX**(pctx) ((pctx)->pStream->bytesProcessed + (pctx)->buffer.byteIndex)
- #define **OSRTSTREAM_ID**(pctx) ((pctx)->pStream->id)
- #define **OSRTSTREAM_FLAGS**(pctx) ((pctx)->pStream->flags)

Typedefs

- typedef long(* **OSRTStreamReadProc**)(struct **OSRTStream** *pStream, ASN1OCTET *pbuffer, size_t bufsize)
- typedef long(* **OSRTStreamBlockingReadProc**)(struct **OSRTStream** *pStream, ASN1OCTET *pbuffer, size_t toReadBytes)
- typedef int(* **OSRTStreamFlushProc**)(struct **OSRTStream** *pStream)
- typedef int(* **OSRTStreamCloseProc**)(struct **OSRTStream** *pStream)
- typedef long(* **OSRTStreamWriteProc**)(struct **OSRTStream** *pStream, const ASN1OCTET *data, size_t numocts)
- typedef int(* **OSRTStreamSkipProc**)(struct **OSRTStream** *pStream, size_t skipBytes)
- typedef int(* **OSRTStreamMarkProc**)(struct **OSRTStream** *pStream, size_t readAheadLimit)
- typedef int(* **OSRTStreamResetProc**)(struct **OSRTStream** *pStream)
- typedef **OSRTStream** **OSRTStream**

Functions

- int **rtStreamRelease** (ASN1CTXT *pctx)
- int **rtStreamClose** (ASN1CTXT *pctx)

- int **rtStreamFlush** (ASN1CTXT *pctx)
- int **rtStreamInit** (ASN1CTXT *pctx)
- long **rtStreamRead** (ASN1CTXT *pctx, ASN1OCTET *pbuffer, size_t bufsize)
- long **rtStreamBlockingRead** (ASN1CTXT *pctx, ASN1OCTET *pbuffer, size_t readBytes)
- int **rtStreamSkip** (ASN1CTXT *pctx, size_t skipBytes)
- long **rtStreamWrite** (ASN1CTXT *pctx, const ASN1OCTET *data, size_t numocts)
- int **rtStreamGetIOBytes** (ASN1CTXT *pctx, size_t *pPos)
- int **rtStreamGetIOBytes56** (ASN1CTXT *pctx)
- int **rtStreamMark** (ASN1CTXT *pctx, size_t readAheadLimit)
- int **rtStreamReset** (ASN1CTXT *pctx)
- ASN1BOOL **rtStreamMarkSupported** (ASN1CTXT *pctx)
- ASN1BOOL **rtStreamIsOpened** (ASN1CTXT *pctx)
- ASN1BOOL **rtStreamIsReadable** (ASN1CTXT *pctx)
- ASN1BOOL **rtStreamIsWritable** (ASN1CTXT *pctx)
- int **rtStreamBufClose** (ASN1CTXT *pctx)
- int **rtStreamBufFlush** (ASN1CTXT *pctx)
- int **rtStreamBufInit** (ASN1CTXT *pctx)
- int **rtStreamBufMark** (ASN1CTXT *pctx, size_t readAheadLimit)
- int **rtStreamBufPreRead** (ASN1CTXT *pctx, size_t size)
- int **rtStreamBufRead** (ASN1CTXT *pctx, ASN1OCTET *pdata, size_t size)
- int **rtStreamBufSkip** (ASN1CTXT *pctx, size_t skipBytes)
- int **rtStreamBufReset** (ASN1CTXT *pctx)
- int **rtStreamBufWrite** (ASN1CTXT *pctx, const ASN1OCTET *data, size_t numocts)
- int **rtStreamFileAttach** (ASN1CTXT *pctx, FILE *pFile, ASN1USINT flags)
- int **rtStreamFileOpen** (ASN1CTXT *pctx, const char *pFilename, ASN1USINT flags)
- int **rtStreamFileCreateReader** (ASN1CTXT *pctx, const char *pFilename)
- int **rtStreamFileCreateWriter** (ASN1CTXT *pctx, const char *pFilename)
- int **rtStreamSocketAttach** (ASN1CTXT *pctx, OSRTSOCKET socket, ASN1USINT flags)
- int **rtStreamSocketCreateReader** (ASN1CTXT *pctx, OSRTSOCKET socket)
- int **rtStreamSocketCreateWriter** (ASN1CTXT *pctx, const char *host, int port)
- int **rtStreamSocketCreateWriter2** (ASN1CTXT *pctx, OSRTSOCKET socket)
- int **rtStreamSocketSetOwnership** (ASN1CTXT *pctx, ASN1BOOL ownSocket)
- int **rtStreamMemoryCreate** (ASN1CTXT *pctx, ASN1USINT flags)
- int **rtStreamMemoryAttach** (ASN1CTXT *pctx, ASN1OCTET *pMemBuf, size_t bufSize, ASN1USINT flags)
- ASN1OCTET * **rtStreamMemoryGetBuffer** (ASN1CTXT *pctx, size_t *pSize)
- int **rtStreamMemoryCreateReader** (ASN1CTXT *pctx, ASN1OCTET *pMemBuf, size_t bufSize)
- int **rtStreamMemoryCreateWriter** (ASN1CTXT *pctx, ASN1OCTET *pMemBuf, size_t bufSize)

Index

- ALLOC_ASNIARRAY
 - rtmem, 8
- ALLOC_ASNIARRAY1
 - rtmem, 8
- ALLOC_ASNIARRAY2
 - rtmem, 8
- ALLOC_ASNI1ELEM
 - rtmem, 8
- ASN1ARRAYSIZE
 - rtmem, 9
- ASN1MALLOC
 - rtmem, 9
- ASN1MEMFREE
 - rtmem, 9
- ASN1MEMFREEPTR
 - rtmem, 9
- ASN1MEMRESET
 - rtmem, 9
- ASN1REALLOC
 - rtmem, 10
- asn1type.h, 105
- bcdHelper
 - rtBCDToString, 37
 - rtStringToBCD, 37
 - rtStringToDynBCD, 37
- Big Integer Helper Functions, 43
- bigintf
 - rtBigIntCompare, 43
 - rtBigIntCopy, 44
 - rtBigIntDigitsNum, 44
 - rtBigIntFastCopy, 44
 - rtBigIntFree, 44
 - rtBigIntGetData, 45
 - rtBigIntGetDataLen, 45
 - rtBigIntInit, 45
 - rtBigIntPrint, 45
 - rtBigIntSetBytes, 46
 - rtBigIntSetInt64, 46
 - rtBigIntSetStr, 46
 - rtBigIntStrCompare, 46
 - rtBigIntToString, 47
- Binary Coded Decimal (BCD) Helper Functions, 36
- Buffered stream functions., 94
- buffermanfun
 - rtMemBufAppend, 26
 - rtMemBufCut, 27
 - rtMemBufFree, 27
 - rtMemBufGetData, 27
 - rtMemBufGetDataLen, 27
 - rtMemBufInit, 27
 - rtMemBufInitBuffer, 28
 - rtMemBufPreAllocate, 28
 - rtMemBufReset, 28
 - rtMemBufSet, 28
 - rtMemBufSetExpandable, 29
 - rtMemBufTrimW, 29
- bufsize
 - OSRTStream, 103
- bytesProcessed
 - OSRTStream, 103
- Character String Conversion Functions, 38
- charstrcon
 - rtBMPToCString, 38
 - rtBMPToNewCString, 39
 - rtBMPToNewCStringEx, 39
 - rtCToBMPString, 39
 - rtCToUCSString, 39
 - rtUCSToCString, 40
 - rtUCSToNewCString, 40
 - rtUCSToNewCStringEx, 40
 - rtUCSToWCSSString, 40
 - rtUTF8CharSize, 41
 - rtUTF8EncodeChar, 41
 - rtUTF8Len, 41
 - rtUTF8LenBytes, 41
 - rtUTF8Strdup, 41
 - rtUTF8Strndup, 42
 - rtUTF8ToWCS, 42
 - rtValidateUTF8, 42
 - rtWCSToUCSString, 42
 - rtWCSToUTF8, 43
- close
 - OSRTStream, 103
- cmfun
 - rtFreeContext, 13
 - rtInitContext, 13
 - rtInitContextBuffer, 13
 - rtInitContextUsingKey, 14
 - rtNewContext, 14
 - rtNewContextUsingKey, 14
 - rtSetCopyValues, 15
 - rtSetFastCopy, 15
- cmp
 - rtCmp16BitCharStr, 51
 - rtCmp32BitCharStr, 52
 - rtCmpBitStr, 52
 - rtCmpBoolean, 52
 - rtCmpCharStr, 53
 - rtCmpInt64, 53
 - rtCmpInteger, 53
 - rtCmpOctStr, 53
 - rtCmpOID, 54
 - rtCmpOID64, 54

- rtCmpOpenType, 54
- rtCmpOpenTypeExt, 55
- rtCmpReal, 55
- rtCmpUInt64, 55
- rtCmpUnsigned, 56
- cmpStdout
 - rtCmpToStdout16BitCharStr, 57
 - rtCmpToStdout32BitCharStr, 57
 - rtCmpToStdoutBitStr, 57
 - rtCmpToStdoutBoolean, 58
 - rtCmpToStdoutCharStr, 58
 - rtCmpToStdoutInt64, 58
 - rtCmpToStdoutInteger, 58
 - rtCmpToStdoutOctStr, 58
 - rtCmpToStdoutOID, 58
 - rtCmpToStdoutOID64, 59
 - rtCmpToStdoutOID64Value, 59
 - rtCmpToStdoutOIDValue, 59
 - rtCmpToStdoutOpenType, 59
 - rtCmpToStdoutOpenTypeExt, 59
 - rtCmpToStdoutOptional, 60
 - rtCmpToStdoutReal, 60
 - rtCmpToStdoutSeqOfElements, 60
 - rtCmpToStdoutTag, 60
 - rtCmpToStdoutUInt64, 60
 - rtCmpToStdoutUnsigned, 60
- Comparison Functions, 50
- Comparison to Standard Output Functions, 56
- Context Management Functions, 13
- copy
 - rtCopy16BitCharStr, 61
 - rtCopy32BitCharStr, 62
 - rtCopyBitStr, 62
 - rtCopyCharStr, 62
 - rtCopyDynBitStr, 63
 - rtCopyDynOctStr, 63
 - rtCopyOctStr, 63
 - rtCopyOID, 63
 - rtCopyOID64, 64
 - rtCopyOpenType, 64
 - rtCopyOpenTypeExt, 64
- Copy Functions, 61
- cruntime
 - HEXCHARTONIBBLE, 5
 - NIBBLETOHEXCHAR, 5
 - rtClearBit, 5
 - rtFileReadBinary, 5
 - rtGetLibInfo, 6
 - rtGetLibVersion, 6
 - rtSetBit, 6
 - rtTestBit, 6
- diag
 - rtdiag, 20
 - rtDiagStream, 21
 - rtHexDump, 21
 - rtHexDumpEx, 21
 - rtHexDumpToFile, 21
 - rtHexDumpToFileEx, 21
 - rtHexDumpToStream, 22
 - rtHexDumpToStreamEx, 22
 - rtHexDumpToString, 22
 - rtHexDumpToStringEx, 22
 - rtSetDiag, 23
- Diagnostic Trace Functions, 20
- errfp
 - rtErrAddIntParm, 23
 - rtErrAddStrParm, 24
 - rtErrAddTagParm, 24
 - rtErrAddUIntParm, 24
 - rtErrFreeParms, 24
 - rtErrGetText, 25
 - rtErrLogUsingCB, 25
 - rtErrPrint, 25
 - rtErrReset, 25
 - rtErrSetData, 25
- Error Formatting and Print Functions, 23
- extra
 - OSRTStream, 103
- File, socket, and memory streams., 97
- flags
 - OSRTStream, 103
- flush
 - OSRTStream, 104
- Formatted Printing Functions, 33
- g_PrintStream
 - prtstrm, 71
- HEXCHARTONIBBLE
 - cruntime, 5
- id
 - OSRTStream, 104
- ioBytes
 - OSRTStream, 104
- Linked List and Stack Utility Functions, 30
- Linked List Utility Functions, 15
- llfuncs
 - rtDListAppend, 16
 - rtDListAppendArray, 16
 - rtDListAppendArrayCopy, 17
 - rtDListFastInit, 16
 - rtDListFindByData, 17
 - rtDListFindByIndex, 17
 - rtDListFindIndexByData, 17
 - rtDListFreeAll, 17
 - rtDListFreeNodes, 18
 - rtDListInit, 18
 - rtDListInsert, 18
 - rtDListInsertAfter, 19
 - rtDListInsertBefore, 19
 - rtDListRemove, 19
 - rtDListToArray, 19
- llsutil
 - rtSListAppend, 30

- rtSListCreate, 31
- rtSListCreateEx, 31
- rtSListFind, 31
- rtSListFree, 31
- rtSListInit, 31
- rtSListInitEx, 32
- rtSListRemove, 32
- rtStackCreate, 32
- rtStackCreateEx, 32
- rtStackInit, 32
- rtStackPop, 32
- rtStackPush, 32
- Low-level unbuffered stream functions., 90
- markedBytesProcessed
 - OSRTStream, 104
- Memory Allocation Macros and Functions, 6
- Memory Buffer Management Functions, 26
- NIBBLETOHEXCHAR
 - cruntime, 5
- Object Identifier Helper Functions, 29
- objidhelpers
 - rtAddOID, 29
 - rtSetOID, 30
- OSIPADDR
 - rtSocket.h, 125
- OSRTPrintStream, 102
 - prtstrm, 70
- OSRTSOCKET
 - rtSocket.h, 126
- OSRTStream, 103
 - bufsize, 103
 - bytesProcessed, 103
 - close, 103
 - extra, 103
 - flags, 103
 - flush, 104
 - id, 104
 - ioBytes, 104
 - markedBytesProcessed, 104
 - read, 104
 - readAheadLimit, 104
 - stream, 89
 - write, 104
- OSRTStreamBlockingReadProc
 - stream, 89
- OSRTStreamCloseProc
 - stream, 89
- OSRTStreamFlushProc
 - stream, 90
- OSRTStreamMarkProc
 - stream, 90
- OSRTStreamReadProc
 - stream, 90
- OSRTStreamResetProc
 - stream, 90
- OSRTStreamSkipProc
 - stream, 90
- OSRTStreamWriteProc
 - stream, 90
- pfun
 - rtBitStrToString, 34
 - rtBoolToString, 34
 - rtInt64ToString, 34
 - rtIntToString, 35
 - rtOctStrToString, 35
 - rtOID64ToString, 35
 - rtOIDToString, 35
 - rtTagToString, 36
 - rtUInt64ToString, 36
 - rtUIntToString, 36
- Print stream structure and function definitions., 69
- Print Values to Standard Output, 65
- Print values to text buffer functions., 78
- Print Values to user specified Stream, 71
- prtstrm
 - g_PrintStream, 71
 - OSRTPrintStream, 70
 - rtDiagToStream, 70
 - rtPrintStreamRelease, 70
 - rtPrintToStream, 70
 - rtSetGlobalPrintStream, 71
 - rtSetPrintStream, 71
- read
 - OSRTStream, 104
- readAheadLimit
 - OSRTStream, 104
- real
 - rtGetMinusInfinity, 33
 - rtGetPlusInfinity, 33
- REAL Helper Functions, 33
- REALLOC_ASN1ARRAY
 - rtmem, 10
- rtAddOID
 - objidhelpers, 29
- rtBCDToString
 - bcdHelper, 37
- rtBigIntCompare
 - bigintf, 43
- rtBigIntCopy
 - bigintf, 44
- rtBigIntDigitsNum
 - bigintf, 44
- rtBigIntFastCopy
 - bigintf, 44
- rtBigIntFree
 - bigintf, 44
- rtBigIntGetData
 - bigintf, 45
- rtBigIntGetDataLen
 - bigintf, 45
- rtBigIntInit

- bigintf, 45
- rtBigIntPrint
 - bigintf, 45
- rtBigIntSetBytes
 - bigintf, 46
- rtBigIntSetInt64
 - bigintf, 46
- rtBigIntSetStr
 - bigintf, 46
- rtBigIntStrCompare
 - bigintf, 46
- rtBigIntToString
 - bigintf, 47
- rtBitStrToString
 - pfun, 34
- rtBMPToCString
 - charstrcon, 38
- rtBMPToNewCString
 - charstrcon, 39
- rtBMPToNewCStringEx
 - charstrcon, 39
- rtBoolToString
 - pfun, 34
- rtClearBit
 - cruntime, 5
- rtCmp16BitCharStr
 - cmp, 51
- rtCmp32BitCharStr
 - cmp, 52
- rtCmpBitStr
 - cmp, 52
- rtCmpBoolean
 - cmp, 52
- rtCmpCharStr
 - cmp, 53
- rtCmpInt64
 - cmp, 53
- rtCmpInteger
 - cmp, 53
- rtCmpOctStr
 - cmp, 53
- rtCmpOID
 - cmp, 54
- rtCmpOID64
 - cmp, 54
- rtCmpOpenType
 - cmp, 54
- rtCmpOpenTypeExt
 - cmp, 55
- rtCmpReal
 - cmp, 55
- rtCmpToStdout16BitCharStr
 - cmpStdout, 57
- rtCmpToStdout32BitCharStr
 - cmpStdout, 57
- rtCmpToStdoutBitStr
 - cmpStdout, 57
- rtCmpToStdoutBoolean
 - cmpStdout, 58
- rtCmpToStdoutCharStr
 - cmpStdout, 58
- rtCmpToStdoutInt64
 - cmpStdout, 58
- rtCmpToStdoutInteger
 - cmpStdout, 58
- rtCmpToStdoutOctStr
 - cmpStdout, 58
- rtCmpToStdoutOID
 - cmpStdout, 58
- rtCmpToStdoutOID64
 - cmpStdout, 59
- rtCmpToStdoutOID64Value
 - cmpStdout, 59
- rtCmpToStdoutOIDValue
 - cmpStdout, 59
- rtCmpToStdoutOpenType
 - cmpStdout, 59
- rtCmpToStdoutOpenTypeExt
 - cmpStdout, 59
- rtCmpToStdoutOptional
 - cmpStdout, 60
- rtCmpToStdoutReal
 - cmpStdout, 60
- rtCmpToStdoutSeqOfElements
 - cmpStdout, 60
- rtCmpToStdoutTag
 - cmpStdout, 60
- rtCmpToStdoutUInt64
 - cmpStdout, 60
- rtCmpToStdoutUnsigned
 - cmpStdout, 60
- rtCmpUInt64
 - cmp, 55
- rtCmpUnsigned
 - cmp, 56
- rtCompare.h, 115
- rtCopy.h, 117
- rtCopy16BitCharStr
 - copy, 61
- rtCopy32BitCharStr
 - copy, 62
- rtCopyBitStr
 - copy, 62
- rtCopyCharStr
 - copy, 62
- rtCopyDynBitStr
 - copy, 63
- rtCopyDynOctStr
 - copy, 63
- rtCopyOctStr
 - copy, 63
- rtCopyOID

- copy, 63
- rtCopyOID64
 - copy, 64
- rtCopyOpenType
 - copy, 64
- rtCopyOpenTypeExt
 - copy, 64
- rtCToBMPString
 - charstrcon, 39
- rtCToUCSSString
 - charstrcon, 39
- rtdiag
 - diag, 20
- rtDiagStream
 - diag, 21
- rtDiagToStream
 - prtstrm, 70
- rtDListAppend
 - llfuns, 16
- rtDListAppendArray
 - llfuns, 16
- rtDListAppendArrayCopy
 - llfuns, 17
- rtDListFastInit
 - llfuns, 16
- rtDListFindByData
 - llfuns, 17
- rtDListFindByIndex
 - llfuns, 17
- rtDListFindIndexByData
 - llfuns, 17
- rtDListFreeAll
 - llfuns, 17
- rtDListFreeNodes
 - llfuns, 18
- rtDListInit
 - llfuns, 18
- rtDListInsert
 - llfuns, 18
- rtDListInsertAfter
 - llfuns, 19
- rtDListInsertBefore
 - llfuns, 19
- rtDListRemove
 - llfuns, 19
- rtDListToArray
 - llfuns, 19
- rtErrAddIntParm
 - errfp, 23
- rtErrAddStrParm
 - errfp, 24
- rtErrAddTagParm
 - errfp, 24
- rtErrAddUIntParm
 - errfp, 24
- rtErrFreeParms
 - errfp, 24
- rtErrGetText
 - errfp, 25
- rtErrLogUsingCB
 - errfp, 25
- rtErrPrint
 - errfp, 25
- rtErrReset
 - errfp, 25
- rtErrSetData
 - errfp, 25
- rtFileReadBinary
 - cruntime, 5
- rtFreeContext
 - cmfun, 13
- rtGetLibInfo
 - cruntime, 6
- rtGetLibVersion
 - cruntime, 6
- rtGetMinusInfinity
 - real, 33
- rtGetPlusInfinity
 - real, 33
- rtHexDump
 - diag, 21
- rtHexDumpEx
 - diag, 21
- rtHexDumpToFile
 - diag, 21
- rtHexDumpToFileEx
 - diag, 21
- rtHexDumpToStream
 - diag, 22
- rtHexDumpToStreamEx
 - diag, 22
- rtHexDumpToString
 - diag, 22
- rtHexDumpToStringEx
 - diag, 22
- rtInitContext
 - cmfun, 13
- rtInitContextBuffer
 - cmfun, 13
- rtInitContextUsingKey
 - cmfun, 14
- rtInt64ToString
 - pfun, 34
- rtIntToString
 - pfun, 35
- rtMakeGeneralizedTime
 - timeutilf, 48
- rtMakeUTCTime
 - timeutilf, 48
- rtMakeXmlDateTime
 - timeutilf, 48
- rtmem

- ALLOC_ASN1ARRAY, 8
- ALLOC_ASN1ARRAY1, 8
- ALLOC_ASN1ARRAY2, 8
- ALLOC_ASN1ELEM, 8
- ASN1ARRAYSIZE, 9
- ASN1MALLOC, 9
- ASN1MEMFREE, 9
- ASN1MEMFREEPTR, 9
- ASN1MEMRESET, 9
- ASN1REALLOC, 10
- REALLOC_ASN1ARRAY, 10
- rtMemAlloc, 10
- rtMemAllocType, 10
- rtMemAllocTypeZ, 11
- rtMemAllocZ, 11
- rtMemFree, 11
- rtMemFreePtr, 11
- rtMemHeapGetDefBlkSize, 12
- rtMemHeapSetDefBlkSize, 12
- rtMemRealloc, 11
- rtMemReset, 12
- rtMemSetAllocFuncs, 12
- rtMemAlloc
 - rtmem, 10
- rtMemAllocType
 - rtmem, 10
- rtMemAllocTypeZ
 - rtmem, 11
- rtMemAllocZ
 - rtmem, 11
- rtMemBufAppend
 - buffermanfun, 26
- rtMemBufCut
 - buffermanfun, 27
- rtMemBufFree
 - buffermanfun, 27
- rtMemBufGetData
 - buffermanfun, 27
- rtMemBufGetDataLen
 - buffermanfun, 27
- rtMemBufInit
 - buffermanfun, 27
- rtMemBufInitBuffer
 - buffermanfun, 28
- rtMemBufPreAllocate
 - buffermanfun, 28
- rtMemBufReset
 - buffermanfun, 28
- rtMemBufSet
 - buffermanfun, 28
- rtMemBufSetExpandable
 - buffermanfun, 29
- rtMemBufTrimW
 - buffermanfun, 29
- rtMemFree
 - rtmem, 11
- rtMemFreePtr
 - rtmem, 11
- rtMemHeapGetDefBlkSize
 - rtmem, 12
- rtMemHeapSetDefBlkSize
 - rtmem, 12
- rtMemory.h, 118
- rtMemRealloc
 - rtmem, 11
- rtMemReset
 - rtmem, 12
- rtMemSetAllocFuncs
 - rtmem, 12
- rtNewContext
 - cmfun, 14
- rtNewContextUsingKey
 - cmfun, 14
- rtOctStrToString
 - pfun, 35
- rtOID64ToString
 - pfun, 35
- rtOIDToString
 - pfun, 35
- rtParseGeneralizedTime
 - timeutilf, 49
- rtParseUTCTime
 - timeutilf, 49
- rtParseXmlDateTime
 - timeutilf, 49
- rtPrint.h, 120
- rtPrint16BitCharStr
 - valsToStdout, 66
- rtPrint32BitCharStr
 - valsToStdout, 66
- rtPrintBitStr
 - valsToStdout, 66
- rtPrintBitStrBraceText
 - valsToStdout, 66
- rtPrintBoolean
 - valsToStdout, 66
- rtPrintCharStr
 - valsToStdout, 66
- rtPrintCloseBrace
 - valsToStdout, 67
- rtPrintDecrIndent
 - valsToStdout, 67
- rtPrintHexStr
 - valsToStdout, 67
- rtPrintIncrIndent
 - valsToStdout, 67
- rtPrintIndent
 - valsToStdout, 67
- rtPrintInt64
 - valsToStdout, 67
- rtPrintInteger
 - valsToStdout, 67

rtPrintOctStr
 valsToStdout, 67
 rtPrintOID
 valsToStdout, 67
 rtPrintOID64
 valsToStdout, 68
 rtPrintOID64Value
 valsToStdout, 68
 rtPrintOIDValue
 valsToStdout, 68
 rtPrintOpenBrace
 valsToStdout, 68
 rtPrintOpenType
 valsToStdout, 68
 rtPrintOpenTypeExt
 valsToStdout, 68
 rtPrintOpenTypeExtBraceText
 valsToStdout, 68
 rtPrintReal
 valsToStdout, 69
 rtPrintStream.h, 121
 rtPrintStreamRelease
 prtstrm, 70
 rtPrintToStream
 prtstrm, 70
 rtPrintToStream.h, 122
 rtPrintToStream16BitCharStr
 valsToStream, 72
 rtPrintToStream32BitCharStr
 valsToStream, 72
 rtPrintToStreamBitStr
 valsToStream, 73
 rtPrintToStreamBitStrBraceText
 valsToStream, 73
 rtPrintToStreamBoolean
 valsToStream, 73
 rtPrintToStreamCharStr
 valsToStream, 73
 rtPrintToStreamCloseBrace
 valsToStream, 74
 rtPrintToStreamDecrIndent
 valsToStream, 74
 rtPrintToStreamHexStr
 valsToStream, 74
 rtPrintToStreamIncrIndent
 valsToStream, 74
 rtPrintToStreamIndent
 valsToStream, 74
 rtPrintToStreamInt64
 valsToStream, 74
 rtPrintToStreamInteger
 valsToStream, 74
 rtPrintToStreamOctStr
 valsToStream, 75
 rtPrintToStreamOID
 valsToStream, 75
 rtPrintToStreamOID64
 valsToStream, 75
 rtPrintToStreamOID64Value
 valsToStream, 75
 rtPrintToStreamOIDValue
 valsToStream, 75
 rtPrintToStreamOpenBrace
 valsToStream, 76
 rtPrintToStreamOpenType
 valsToStream, 76
 rtPrintToStreamOpenTypeExt
 valsToStream, 76
 rtPrintToStreamOpenTypeExtBraceText
 valsToStream, 76
 rtPrintToStreamReal
 valsToStream, 76
 rtPrintToStreamUInt64
 valsToStream, 77
 rtPrintToStreamUnicodeCharStr
 valsToStream, 77
 rtPrintToStreamUnivCharStr
 valsToStream, 77
 rtPrintToStreamUnsigned
 valsToStream, 77
 rtPrintToString
 rtPrintToString, 79
 rtPrintToString16BitCharStr, 79
 rtPrintToString32BitCharStr, 79
 rtPrintToStringBitStr, 79
 rtPrintToStringBitStrBraceText, 79
 rtPrintToStringBoolean, 80
 rtPrintToStringCharStr, 80
 rtPrintToStringCloseBrace, 80
 rtPrintToStringDecrIndent, 80
 rtPrintToStringHexStr, 80
 rtPrintToStringIncrIndent, 81
 rtPrintToStringIndent, 81
 rtPrintToStringInt64, 81
 rtPrintToStringInteger, 81
 rtPrintToStringOctStr, 81
 rtPrintToStringOID, 81
 rtPrintToStringOID64, 82
 rtPrintToStringOID64Value, 82
 rtPrintToStringOIDValue, 82
 rtPrintToStringOpenBrace, 82
 rtPrintToStringOpenType, 82
 rtPrintToStringOpenTypeExt, 83
 rtPrintToStringOpenTypeExtBraceText, 83
 rtPrintToStringReal, 83
 rtPrintToStringUInt64, 83
 rtPrintToStringUnicodeCharStr, 84
 rtPrintToStringUnivCharStr, 84
 rtPrintToStringUnsigned, 84
 rtPrintToString.h, 123
 rtPrintToString16BitCharStr
 rtPrintToString, 79

rtPrintToString32BitCharStr
 rtPrintToString, 79
 rtPrintToStringBitStr
 rtPrintToString, 79
 rtPrintToStringBitStrBraceText
 rtPrintToString, 79
 rtPrintToStringBoolean
 rtPrintToString, 80
 rtPrintToStringCharStr
 rtPrintToString, 80
 rtPrintToStringCloseBrace
 rtPrintToString, 80
 rtPrintToStringDecrIndent
 rtPrintToString, 80
 rtPrintToStringHexStr
 rtPrintToString, 80
 rtPrintToStringIncrIndent
 rtPrintToString, 81
 rtPrintToStringIndent
 rtPrintToString, 81
 rtPrintToStringInt64
 rtPrintToString, 81
 rtPrintToStringInteger
 rtPrintToString, 81
 rtPrintToStringOctStr
 rtPrintToString, 81
 rtPrintToStringOID
 rtPrintToString, 81
 rtPrintToStringOID64
 rtPrintToString, 82
 rtPrintToStringOID64Value
 rtPrintToString, 82
 rtPrintToStringOIDValue
 rtPrintToString, 82
 rtPrintToStringOpenBrace
 rtPrintToString, 82
 rtPrintToStringOpenType
 rtPrintToString, 82
 rtPrintToStringOpenTypeExt
 rtPrintToString, 83
 rtPrintToStringOpenTypeExtBraceText
 rtPrintToString, 83
 rtPrintToStringReal
 rtPrintToString, 83
 rtPrintToStringUInt64
 rtPrintToString, 83
 rtPrintToStringUnicodeCharStr
 rtPrintToString, 83
 rtPrintToStringUnivCharStr
 rtPrintToString, 84
 rtPrintToStringUnsigned
 rtPrintToString, 84
 rtPrintUInt64
 valsToStdout, 69
 rtPrintUnicodeCharStr
 valsToStdout, 69
 rtPrintUnivCharStr
 valsToStdout, 69
 rtPrintUnsigned
 valsToStdout, 69
 rtSetBit
 cruntime, 6
 rtSetCopyValues
 cmfun, 15
 rtSetDiag
 diag, 23
 rtSetFastCopy
 cmfun, 15
 rtSetGlobalPrintStream
 prtstrm, 71
 rtSetLocalTime
 timeutilf, 50
 rtSetOID
 objidhelpers, 30
 rtSetPrintStream
 prtstrm, 71
 rtSListAppend
 llsutil, 30
 rtSListCreate
 llsutil, 31
 rtSListCreateEx
 llsutil, 31
 rtSListFind
 llsutil, 31
 rtSListFree
 llsutil, 31
 rtSListInit
 llsutil, 31
 rtSListInitEx
 llsutil, 32
 rtSListRemove
 llsutil, 32
 rtSocket.h, 125
 OSIPADDR, 125
 OSRTSOCKET, 126
 rtSocketAccept
 sockets, 85
 rtSocketAddrToStr
 sockets, 85
 rtSocketBind
 sockets, 85
 rtSocketBlockingRead
 sockets, 85
 rtSocketClose
 sockets, 86
 rtSocketConnect
 sockets, 86
 rtSocketCreate
 sockets, 86
 rtSocketCreateUDP
 sockets, 86
 rtSocketGetLocalIPAddress

- sockets, 86
- rtSocketListen
 - sockets, 87
- rtSocketRecv
 - sockets, 87
- rtSocketRecvFrom
 - sockets, 87
- rtSocketsCleanup
 - sockets, 87
- rtSocketSend
 - sockets, 87
- rtSocketSendTo
 - sockets, 88
- rtSocketsInit
 - sockets, 88
- rtSocketStrToAddr
 - sockets, 88
- rtStackCreate
 - llsutil, 32
- rtStackCreateEx
 - llsutil, 32
- rtStackInit
 - llsutil, 32
- rtStackPop
 - llsutil, 32
- rtStackPush
 - llsutil, 32
- rtStream
 - rtStreamBlockingRead, 91
 - rtStreamClose, 91
 - rtStreamFlush, 91
 - rtStreamGetIOBytes, 92
 - rtStreamInit, 92
 - rtStreamIsOpened, 92
 - rtStreamIsReadable, 92
 - rtStreamIsWritable, 92
 - rtStreamMark, 93
 - rtStreamMarkSupported, 93
 - rtStreamRead, 93
 - rtStreamReset, 93
 - rtStreamSkip, 94
 - rtStreamWrite, 94
- rtStream.h, 127
- rtStreamBlockingRead
 - rtStream, 91
- rtStreamBuf
 - rtStreamBufClose, 95
 - rtStreamBufFlush, 95
 - rtStreamBufInit, 95
 - rtStreamBufMark, 95
 - rtStreamBufPreRead, 96
 - rtStreamBufRead, 96
 - rtStreamBufReset, 96
 - rtStreamBufSkip, 96
 - rtStreamBufWrite, 97
- rtStreamBufClose
 - rtStreamBuf, 95
- rtStreamBufFlush
 - rtStreamBuf, 95
- rtStreamBufInit
 - rtStreamBuf, 95
- rtStreamBufMark
 - rtStreamBuf, 95
- rtStreamBufPreRead
 - rtStreamBuf, 96
- rtStreamBufRead
 - rtStreamBuf, 96
- rtStreamBufReset
 - rtStreamBuf, 96
- rtStreamBufSkip
 - rtStreamBuf, 96
- rtStreamBufWrite
 - rtStreamBuf, 97
- rtStreamClose
 - rtStream, 91
- rtStreamFileAttach
 - rtStreamImp, 98
- rtStreamFileCreateReader
 - rtStreamImp, 98
- rtStreamFileCreateWriter
 - rtStreamImp, 98
- rtStreamFileOpen
 - rtStreamImp, 98
- rtStreamFlush
 - rtStream, 91
- rtStreamGetIOBytes
 - rtStream, 92
- rtStreamImp
 - rtStreamFileAttach, 98
 - rtStreamFileCreateReader, 98
 - rtStreamFileCreateWriter, 98
 - rtStreamFileOpen, 98
 - rtStreamMemoryAttach, 99
 - rtStreamMemoryCreate, 99
 - rtStreamMemoryCreateReader, 99
 - rtStreamMemoryCreateWriter, 99
 - rtStreamMemoryGetBuffer, 100
 - rtStreamSocketAttach, 100
 - rtStreamSocketCreateReader, 100
 - rtStreamSocketCreateWriter, 100
 - rtStreamSocketCreateWriter2, 101
 - rtStreamSocketSetOwnership, 101
- rtStreamInit
 - rtStream, 92
- rtStreamIsOpened
 - rtStream, 92
- rtStreamIsReadable
 - rtStream, 92
- rtStreamIsWritable
 - rtStream, 92
- rtStreamMark
 - rtStream, 93

- rtStreamMarkSupported
 - rtStream, 93
- rtStreamMemoryAttach
 - rtStreamImp, 99
- rtStreamMemoryCreate
 - rtStreamImp, 99
- rtStreamMemoryCreateReader
 - rtStreamImp, 99
- rtStreamMemoryCreateWriter
 - rtStreamImp, 99
- rtStreamMemoryGetBuffer
 - rtStreamImp, 100
- rtStreamRead
 - rtStream, 93
- rtStreamReset
 - rtStream, 93
- rtStreamSkip
 - rtStream, 94
- rtStreamSocketAttach
 - rtStreamImp, 100
- rtStreamSocketCreateReader
 - rtStreamImp, 100
- rtStreamSocketCreateWriter
 - rtStreamImp, 100
- rtStreamSocketCreateWriter2
 - rtStreamImp, 101
- rtStreamSocketSetOwnership
 - rtStreamImp, 101
- rtStreamWrite
 - rtStream, 94
- rtStringToBCD
 - bcdHelper, 37
- rtStringToDynBCD
 - bcdHelper, 37
- rtTagToString
 - pfun, 36
- rtTestBit
 - cruntime, 6
- rtUCSToCString
 - charstrcon, 40
- rtUCSToNewCString
 - charstrcon, 40
- rtUCSToNewCStringEx
 - charstrcon, 40
- rtUCSToWCSSString
 - charstrcon, 40
- rtUInt64ToString
 - pfun, 36
- rtUIntToString
 - pfun, 36
- rtUTF8CharSize
 - charstrcon, 41
- rtUTF8EncodeChar
 - charstrcon, 41
- rtUTF8Len
 - charstrcon, 41
- rtUTF8LenBytes
 - charstrcon, 41
- rtUTF8Strdup
 - charstrcon, 41
- rtUTF8Strndup
 - charstrcon, 42
- rtUTF8ToWCS
 - charstrcon, 42
- rtValidateUTF8
 - charstrcon, 42
- rtWCSToUCSSString
 - charstrcon, 42
- rtWCSToUTF8
 - charstrcon, 43
- sockets
 - rtSocketAccept, 85
 - rtSocketAddrToStr, 85
 - rtSocketBind, 85
 - rtSocketBlockingRead, 85
 - rtSocketClose, 86
 - rtSocketConnect, 86
 - rtSocketCreate, 86
 - rtSocketCreateUDP, 86
 - rtSocketGetLocalIPAddress, 86
 - rtSocketListen, 87
 - rtSocketRecv, 87
 - rtSocketRecvFrom, 87
 - rtSocketsCleanup, 87
 - rtSocketSend, 87
 - rtSocketSendTo, 88
 - rtSocketsInit, 88
 - rtSocketStrToAddr, 88
- stream
 - OSRTStream, 89
 - OSRTStreamBlockingReadProc, 89
 - OSRTStreamCloseProc, 89
 - OSRTStreamFlushProc, 90
 - OSRTStreamMarkProc, 90
 - OSRTStreamReadProc, 90
 - OSRTStreamResetProc, 90
 - OSRTStreamSkipProc, 90
 - OSRTStreamWriteProc, 90
- Stream structure definitions., 88
- TCP/IP and UDP socket functions., 84
- Time Helper Functions, 47
- timeutilf
 - rtMakeGeneralizedTime, 48
 - rtMakeUTCTime, 48
 - rtMakeXmlDateTime, 48
 - rtParseGeneralizedTime, 49
 - rtParseUTCTime, 49
 - rtParseXmlDateTime, 49
 - rtSetLocalTime, 50
- valsToStdout
 - rtPrint16BitCharStr, 66
 - rtPrint32BitCharStr, 66

- rtPrintBitStr, 66
- rtPrintBitStrBraceText, 66
- rtPrintBoolean, 66
- rtPrintCharStr, 66
- rtPrintCloseBrace, 67
- rtPrintDecrIndent, 67
- rtPrintHexStr, 67
- rtPrintIncrIndent, 67
- rtPrintIndent, 67
- rtPrintInt64, 67
- rtPrintInteger, 67
- rtPrintOctStr, 67
- rtPrintOID, 67
- rtPrintOID64, 68
- rtPrintOID64Value, 68
- rtPrintOIDValue, 68
- rtPrintOpenBrace, 68
- rtPrintOpenType, 68
- rtPrintOpenTypeExt, 68
- rtPrintOpenTypeExtBraceText, 68
- rtPrintReal, 69
- rtPrintUInt64, 69
- rtPrintUnicodeCharStr, 69
- rtPrintUnivCharStr, 69
- rtPrintUnsigned, 69
- valsToStream
 - rtPrintToStream16BitCharStr, 72
 - rtPrintToStream32BitCharStr, 72
 - rtPrintToStreamBitStr, 73
 - rtPrintToStreamBitStrBraceText, 73
 - rtPrintToStreamBoolean, 73
 - rtPrintToStreamCharStr, 73
 - rtPrintToStreamCloseBrace, 74
 - rtPrintToStreamDecrIndent, 74
 - rtPrintToStreamHexStr, 74
 - rtPrintToStreamIncrIndent, 74
 - rtPrintToStreamIndent, 74
 - rtPrintToStreamInt64, 74
 - rtPrintToStreamInteger, 74
 - rtPrintToStreamOctStr, 75
 - rtPrintToStreamOID, 75
 - rtPrintToStreamOID64, 75
 - rtPrintToStreamOID64Value, 75
 - rtPrintToStreamOIDValue, 75
 - rtPrintToStreamOpenBrace, 76
 - rtPrintToStreamOpenType, 76
 - rtPrintToStreamOpenTypeExt, 76
 - rtPrintToStreamOpenTypeExtBraceText, 76
 - rtPrintToStreamReal, 76
 - rtPrintToStreamUInt64, 77
 - rtPrintToStreamUnicodeCharStr, 77
 - rtPrintToStreamUnivCharStr, 77
 - rtPrintToStreamUnsigned, 77
- write
 - OSRTStream, 104