Summary of Comments on MMSREV11.PDF

Page: 1

Sequence number: 1
Author: Denise M Patrishkoff
Date: 2/19/03 2:27:37 PM
Type: Note Annotations in this document are changes to the manual - last changes are 2/19/03.

Page: 4

Sequence number: 1
Date: 2/7/02 11:07:07 AM
Type: Strikeout 810-

Sequence number: 2
Date: 2/7/02 11:07:10 AM
Type: Strikeout 810-

Sequence number: 3
Date: 2/7/02 11:07:24 AM
Type: Note new area code - 586

Page: 97

ms_count_ind_pend
This function has changed - a new parameter has been added - please refer to the release notes for more information

Page: 98

ms_count_req_pend
This function prototype was incorrect - ST_INT ms_count_req_pend (ST_INT chan);

Page: 114

Sequence number: 1
Date: 2/7/02 11:08:23 AM
Type: Note: by default s_debug_sel is now set to ACSE_ERR_PRINT | ACSE_NERR_PRINT

Sequence number: 2
Date: 2/7/02 11:08:28 AM
Type: Strikeout - By default, s_debug_sel is set to ACSE_ERR_PRINT.

Page: 128

Sequence number: 1
Date: 10/11/01 12:31:47 PM -04'00'
Type: Note: this variable is now set to SD_FALSE by default

Sequence number: 2
Date: 10/11/01 12:34:23 PM -04'00'
Type: Note: S_THISFILE was eliminated - any module using SISCO logging must have the following statement instead: static char *thisFileName = _FILE_;

Sequence number: 3
Date: 10/11/01 2:19:32 PM -04'00'
Type: Note: mem_chk.h has changed - please examine the file included with your release - do not use any sections included in #SMEM_ENABLE (for use with MMS-EASE Lite ONLY)

Sequence number: 4
Date: 10/11/01 12:42:23 PM -04'00'
Type: Note: chk_free_wipe has been deleted.

Sequence number: 6
Date: 2/7/02 11:09:39 AM
Type: Strikeout - #ifdef S_THISFILE

Sequence number: 7
Date: 2/7/02 11:10:03 AM
Type: Strikeout - x_chk_free_wipe

Sequence number: 8
Date: 2/7/02 11:10:51 AM
Type: Strikeout - The default is SD_TRUE.

Page: 139

Sequence number: 1
Date: 2/7/02 11:11:45 AM
Type: Note: gs_install function is no longer used. Multi-threading is now initialized on first call to S_LOCK_RESOURCES.

Sequence number: 2
Date: 2/15/02 8:08:03 AM
Type: Note: This macro is really called S_LOCK_COMMON_RESOURCES.

Sequence number: 3
Date: 2/14/02 2:35:44 PM
Type: Note: For WIN32, Tru64 Unix and AIX, the define is now present in glbsem.h - no longer necessary to use the S_MT_SUPPORT switch when compiling.

Sequence number: 4
Date: 2/7/02 11:11:40 AM
Type: Strikeout - To enable multi-threaded support, you must call the gs_install function before any other APIs or macros can be used.

Page: 141

Sequence number: 1
Date: 2/14/02 2:36:06 PM
Type: Note: The implementation of this function has been changed. By default, the function creates an auto-reset or manual-reset, non-signaled event semaphore.

Sequence number: 2
Date: 2/14/02 2:38:58 PM
Type: Note: New function prototype: ST_EVENT_SEM gs_get_event_sem (ST_BOOLEAN manualReset) where manualReset is either SD_TRUE or SD_FALSE

Page: 142

Sequence number: 1
Date: 10/11/00 1:37:36 PM -04'00'
Type: Note: gs_install function is no longer used. Multi-threading is now initialized on first call to S_LOCK_RESOURCES.

Sequence number: 2
Date: 2/7/02 12:47:23 PM
Type: Strikeout - gs_install (entire function)
Usage: This function is used to install the multi-thread API and must be called before any other API function can be used.

Function Prototype: ST_RET gs_install (ST_VOID);
Parameters: None
Return Value: ST_RET SD_SUCCESS. The semaphore was installed successfully.
SD_FAILURE. Attempt to install the semaphore failed.

Page: 145

Sequence number: 1
Date: 10/11/00 12:39:43 PM -04'00'
Type: Note: If the timeout is greater than 0, then the function waits for the event semaphore for the duration of the timeout period.

Sequence number: 2
Date: 10/11/00 12:41:05 PM -04'00'
Type: Note: There are three return values to this function:
SD_SUCCESS - the semaphore is signaled.
SD_TIMEOUT - the timeout period elapsed and the semaphore is non-signaled.
SD_FAILURE - any other error condition.

Page: 153

Sequence number: 1
Date: 4/3/01 10:38:01 AM -04'00'
Type: Note: function now documented
ms_reset_init_param
Usage: This function is used to reset the MMS Initiate parameters by moving them into the appropriate parameters in the mms_chan_info[chan] structure. This should be normally be done after connection termination.
Function Prototype: ST_VOID ms_reset_init_param (ST_INT chan);
Parameters:
chan This is the channel number associated with this connection. This is used to map into mms_chan_info structure.
Return Value: ST_VOID (ignored)

Page: 155

Sequence number: 1
Date: 12/4/01 12:35:19 PM
Type: Note: 2nd sentence of #8 is revised to say...The program within u_init_ind must be written to decide whether or not to accept or decline the connect indication at the MMS level.

Sequence number: 2
Date: 2/7/02 12:49:26 PM
Type: Strikeout – The program, within u_init_ind on Node B, then modifies the preferred initiate parameters.

Page: 188

Sequence number: 1
Date: 2/7/02 12:50:02 PM
Type: Note: mp_error_resp should be mp_err_resp

Page: 282
Sequence number: 1
Date: 2/19/03 2:33:54 PM
Type: Note: This variable is obsolete along with the ms_asn1_to_runtime function - use ms_runtime_create and ms_runtime_destroy instead - however this variable remains for backward compatibility.

Page: 307
Sequence number: 1
Date: 12/1/97 2:47:49 PM
Type: Note: Correction to BCD - A ST_INT8 should be used when x is [1..2]. The ST_INT16 integer is used when x is [3..4]. The ST_INT32 integer is used when x is [5..8].

Sequence number: 2
Date: 12/1/97 2:48:46 PM
Type: Note: Correction to Int64 - The SISCO macro for the C language representation of Int64 is ST_INT64.

Sequence number: 3
Date: 12/1/97 2:49:43 PM
Type: Note: Correction to Uint64 - This type is encoded as a MMS unsigned integer eight bytes in length where the value must be between 0 and +2 64th power -1.

Page: 308
Sequence number: 1
Date: 2/6/02 10:39:34 AM
Type: Note: New time type
Utctime - This type is encoded as UtcTime with seconds relative to GMT midnight January 1, 1970. The SISCO macro for the C language representation of Utctime is a structure (MMS_UTC_TIME) containing 3 consecutive ST_UINT32. The value contained in the first ST_UINT32 represents the number of seconds since January 1, 1970. The seconds ST_UINT32 represents number of microseconds of a second. And the last ST_UINT32 contains quality flags, only least significant byte is used.

Page: 313
Sequence number: 1
Date: 2/19/03 2:35:17 PM
Type: Note: This sentence contains a typo. It should say ms_add_named_type.

Page: 315
Sequence number: 1
Date: 2/7/02 12:52:14 PM
Type: Note: el_tag has new values
RT-ARR_START 1
RT_STR_START 2
RT_BOOL 3
RT_BIT_STRING 4
RT_INTEGER 5
RT_UNSIGNED 6
RT_FLOATING_POINT 7
RT_OCTET_STRING 9
RT_VISIBLE_STRING 10
RT_GENERAL_TIME 11
RT_BINARY_TIME 12
RT_BCD 13
RT_BOOLEANARRAY 14

RT_UTC_TIME 17
RT_STR_END 18
RT_ARR_END 19

Sequence number: 2
Date: 2/6/02 3:01:55 PM
Type: Note: new elements in runtime_type structure
Also for MMS-EASE ST_INT is now defined as ST_RTINT.

Sequence number: 3
Date: 2/6/02 3:08:46 PM
Type: Note: new structure

```
struct runtime_type
{
ST_UCHAR el_tag;
ST_RTINT el_size;
ST_RTINT offset_to_last;
union
{
struct
{
ST_RTINT el_len; /
ST_RTINT pad;
/* included to allow aggregate initialization*/
} p;
struct /* structure (top or bottom)*/
{
ST_RTINT num_rt_blks;
ST_RTINT pad;
/* included to allow aggregate initialization*/
ST_BOOLEAN packd;
} str;
struct
{
ST_RTINT num_elmnts;
ST_RTINT num_rt_blks;
ST_BOOLEAN packd;
} arr;
ST_CHAR name[MAX_IDENT_LEN+1];
} u;
};
typedef struct runtime_type RUNTIME_TYPE;
```

Sequence number: 4
Date: 2/7/02 12:51:36 PM
Type: Strikeout – entire structure

```
struct runtime_type
{
ST_UCHAR el_tag;
ST_INT el_size;
ST_INT offset_to_last;
union
{
struct
{
ST_INT el_len;
} p;
struct
{
ST_BOOLEAN packd;
ST_INT num_rt_blks;
} str;
struct
{
ST_INT num_elmnts;
ST_INT num_rt_blks;
ST_BOOLEAN packd;
ST_INT loops;
} arr;
ST_CHAR name[MAX_IDENT_LEN+1];
```

```
} u;
};
typedef struct runtime_type RUNTIME_TYPE;
```

Page: 326

Sequence number: 1
Date: 2/6/02 3:54:45 PM
Type: Note: new element in m_arb_data_ctrl

Sequence number: 2
Date: 2/7/02 12:53:51 PM
Type: Note :

```
typedef struct m_arb_data_ctrl
{
ST_RET (*arrStart) (RT_AA_CTRL *rtaa);
ST_RET (*arrEnd) (RT_AA_CTRL *rtaa);
ST_RET (*strStart) (RT_AA_CTRL *rtaa);
ST_RET (*strEnd) (RT_AA_CTRL *rtaa);
ST_RET (*int8) (ST_INT8 *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*int16) (ST_INT16 *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*int32) (ST_INT32 *data_dest, RT_AA_CTRL *rtaa);
#ifdef INT64_SUPPORT
ST_RET (*int64) (ST_INT64 *data_dest, RT_AA_CTRL *rtaa);
#endif
ST_RET (*uint8) (ST_UINT8 *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*uint16) (ST_UINT16 *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*uint32) (ST_UINT32 *data_dest, RT_AA_CTRL *rtaa);
#ifdef INT64_SUPPORT
ST_RET (*uint64) (ST_UINT64 *data_dest, RT_AA_CTRL *rtaa);
#endif
ST_RET (*flt) (ST_FLOAT *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*dbl) (ST_DOUBLE *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*oct) (ST_UCHAR *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*booln) (ST_BOOLEAN *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*bcd1) (ST_INT8 *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*bcd2) (ST_INT16 *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*bcd4) (ST_INT32 *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*bs) (ST_UCHAR *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*vis) (ST_CHAR *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*bt4) (ST_INT32 *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*bt6) (ST_INT32 *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*gt) (time_t *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*utc) (MMS_UTC_TIME *data_dest, RT_AA_CTRL *rtaa);
} M_ARB_DATA_CTRL;
```

Sequence number: 3
Date: 2/7/02 12:54:42 PM
Type: Strikeout – entire structure

```
typedef struct m_arb_data_ctrl
{
ST_RET (*arrStart) (RT_AA_CTRL *rtaa);
ST_RET (*arrEnd) (RT_AA_CTRL *rtaa);
ST_RET (*strStart) (RT_AA_CTRL *rtaa);
ST_RET (*strEnd) (RT_AA_CTRL *rtaa);
ST_RET (*int8) (ST_INT8 *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*int16) (ST_INT16 *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*int32) (ST_INT32 *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*int64) (ST_INT64 *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*uint8) (ST_UINT8 *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*uint16) (ST_UINT16 *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*uint32) (ST_UINT32 *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*uint64) (ST_UINT64 *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*flt) (ST_FLOAT *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*dbl) (ST_DOUBLE *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*oct) (ST_UCHAR *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*bool) (ST_BOOLEAN *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*bcd1) (ST_INT8 *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*bcd2) (ST_INT16 *data_dest, RT_AA_CTRL *rtaa);
```

```
ST_RET (*bcd4) (ST_INT32 *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*bs) (ST_UCHAR *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*vis) (ST_CHAR *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*bt4) (ST_INT32 *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*bt6) (ST_INT32 *data_dest, RT_AA_CTRL *rtaa);
ST_RET (*gt) (time_t *data_dest, RT_AA_CTRL *rtaa);
} M_ARB_DATA_CTRL;
```

Page: 328

Sequence number: 1
Date: 2/6/02 3:14:19 PM
Type: Note: utc - This function is called when a UCT element is encountered in the derived MMS type. A pointer to the beginning of a structure of type MMS_UTC_TIME and a pointer to the RT_AA_CTRL structure representing the utc in the MMS Data are supplied as arguments.

Page: 330

Sequence number: 1
Date: 2/6/02 3:28:24 PM
Type: Note: strt_asn1_bld is now called asn1r_strt_asn1_bld.
This function must be called first to initialize the buffer before calling this function.

Sequence number: 2
Date: 2/6/02 3:29:25 PM
Type: Note: ST_RET ms_aa_to_asn1 (ASN1_ENC_CTXT *aCtx, ALT_ACCESS *alt_acc);

Sequence number: 3
Date: 2/6/02 3:47:39 PM
Type: Note: New example

```
ASN1_ENC_CTXT aCtxt
/* ASN.1 encode content */
asn1r_strt_asn1_bld (&aCtx, asn1_buffer, asn1_buf_size);
if (ms_aa_to_asn1 (alt_acc) = SD_SUCCESS)
{
asn1len = aCtx.asn1r_buf_end - aCtx.asn1r_field_ptr;
asn1ptr = aCtx.asn1r_field_ptr +1;
}
```

Sequence number: 4
Author: denise
Date: 2/7/02 11:05:54 AM
Type: Strikeout

```
strt_
```

Sequence number: 5
Author: denise
Date: 2/7/02 11:06:03 AM
Type: Strikeout

```
asn1_bld
```

Sequence number: 6
Author: denise
Date: 2/7/02 11:06:10 AM
Type: Strikeout

```
strt_asn1_bld (aa_buf, aa_buf_size);
if (ms_aa_to_asn1 (&vmi->i.alt_acc) = SD_SUCCESS)
{
asn1_start = asn1_field_ptr+1;
asn1_len = (aa_buf + aa_buf_size) - asn1_start;
vl->alt_access.len = asn1_len;
vl->alt_access.data = asn1_start;
vl->alt_access_pres = SD_TRUE;
}
```

Page: 339

Sequence number: 1
Date: 2/19/03 2:30:59 PM
Type: Note: This function has been replaced with the ms_runtime_create function - it is documented in the release notes. There is also a new function used to free the runtime_type called ms_runtime_destroy.

Sequence number: 2
Date: 2/19/03 2:29:31 PM
Type: Note: to use this function it is necessary to calculate the number of RUNTIME_TYPE structures needed. This is determined by using m_calc_rt_size.this reference is incorrect - please see page 315 for a description of this structure.

Sequence number: 3
Date: 2/7/02 11:04:20 AM
Type: Strikeout See page 2-89

Sequence number: 4
Date: 2/19/03 2:31:51 PM
Type: Note: If you do not want to change your code to use these new functions - there is a backward compatible macro replacement for this function.

Page: 342

Sequence number: 1
Date: 2/7/02 11:03:50 AM
Type: Note: new parameter for this function
ST_RET ms_local_to_asn1_aa
(ASN1_ENC_CTXT *aCtx, SD_CONST RUNTIME_TYPE *rt_head,
ST_INT rt_num,
ALT_ACCESS *alt_acc,
ST_CHAR *dptr);

Sequence number: 2
Date: 2/7/02 12:55:22 PM
Type: Strikeout -
ST_RET ms_local_to_asn1_aa (RUNTIME_TYPE *rt_head, ST_INT rt_num, ALT_ACCESS *alt_acc, ST_CHAR *dptr);

Page: 343

Sequence number: 1
Date: 2/7/02 12:55:40 PM
Type: Note: ST_RET ms_locl_to_asn1_aa (ASN1_ENC_CTXT *aCtx, NAMED_TYPE *tptr, ALT_ACCESS *alt_acc, ST_CHAR *dptr);

Sequence number: 2
Date: 2/7/02 12:55:46 PM
Type: Note: New Example

```
#define BUFFER_LEN 1000
ST_CHAR asn1_buffer [BUFFER_LEN];
ASN1_ENC_CTXT aCtx; /* ASN.1 encode context */
struct object_name type_name;
type = ms_find_named_type_obj (type_name, 0);
asn1r_strt_asn1_bld (&aCtx, asn1_buffer, BUFFER_LEN);
ms_locl_to_asn1 (&aCtx, type, data);
asn1len = aCtx.asn1r_buf_end - aCtx.asn1r_field_ptr;
asn1prt = aCtx.asn1r_field_ptr + 1;
```

Sequence number: 3
Date: 2/7/02 11:00:32 AM
Type: Strikeout - Additionally, the location and length of the ASN.1 representation is passed to the application program using the ASN1DE global variable field_ptr.

Sequence number: 4
Date: 2/7/02 12:55:33 PM
Type: Strikeout - ST_RET ms_locl_to_asn1 (NAMED_TYPE *type, ST_CHAR *src);

Sequence number: 5
Date: 2/7/02 12:55:53 PM
Type: Strikeout –

```
#define BUFFER_LEN 100
ST_CHAR asn1_buffer[BUFFER_LEN]; /* buffer to put ASN.1 in */
struct object_name type_name; /* the name of the type */
type = ms_find_named_type_obj(type_name,0);
/* get the named type pointer */
strt_asn1_bld(asn1_buffer,BUFFER_LEN);
/* initialize the ASN.1 tools */
ms_locl_to_asn1(typeptr,src); /* convert the data */
asn1len = (asn1_buffer + BUFFER_LEN) - field_ptr - 1;
/* length of ASN.1 */
asn1ptr = field_ptr+1; /* pointer to ASN.1 */
```

Page: 344

Sequence number: 1
Date: 2/7/02 12:56:55 PM
Type: Note
New Example:

```
#define BUFFER_LEN 1000
ST_CHAR asn1_buffer [BUFFER_LEN];
ASN1_ENC_CTXT aCtx; /* ASN.1 encode context */
struct object_name type_name;
type = ms_find_named_type_obj (type_name, 0);
asn1r_strt_asn1_bld (&aCtx, asn1_buffer, BUFFER_LEN);
ms_locl_to_asn1_aa (&aCtx, typeptr, alt_acc, dptr);
asn1len = aCtx.asn1r_buf_end - aCtx.asn1r_field_ptr;
asn1prt = aCtx.asn1r_field_ptr + 1;
```

Sequence number: 2
Date: 2/7/02 12:56:12 PM
Type: Note: new parameter in this function
ST_RET ms_locl_to_asn1_aa (ASN1_ENC_CTXT *aCtx, NAMED_TYPE *tptr,
ALT_ACCESS *alt_acc, ST_CHAR *dptr);

Sequence number: 3
Date: 2/7/02 12:56:25 PM
Type: Strikeout - ST_RET ms_locl_to_asn1_aa (NAMED_TYPE *tptr,
ALT_ACCESS *alt_acc, ST_CHAR *dptr);

Sequence number: 4
Date: 2/7/02 12:56:49 PM
Type: Strikeout - Additionally, the location and length of the ASN.1 representation is passed to the application program using the ASN1DE global variable field_ptr.

Sequence number: 5
Date: 2/7/02 12:57:05 PM
Type: Strikeout

```
#define BUFFER_LEN 100
ST_CHAR asn1_buffer[BUFFER_LEN]; /* buffer to put ASN.1 in */
struct object_name type_name; /* the name of the type */
type = ms_find_named_type_obj(type_name,0);
```

```
/* get the named type pointer */
strt_asn1_bld(asn1_buffer,BUFFER_LEN);
/* initialize the ASN.1 tools */
ms_locl_to_asn1(typeptr,src); /* convert the data */
asn1len = (asn1_buffer + BUFFER_LEN) - field_ptr - 1;
/* length of ASN.1 */
asn1ptr = field_ptr+1; /* pointer to ASN.1 */
```

Page: 362

Sequence number: 1
Date: 2/7/02 12:57:55 PM
Type: Note - change to the data element

```
Data ::= CHOICE {
context tag 0 is reserved for access_result
array [1] IMPLICIT SEQUENCE OF Data,
structure [2] IMPLICIT SEQUENCE OF Data,
boolean [3] IMPLICIT BOOLEAN,
bit-string [4] IMPLICIT BIT STRING,
integer [5] IMPLICIT INTEGER,
unsigned [6] IMPLICIT INTEGER,
floating-point [7] IMPLICIT FloatingPoint,
real [8] IMPLICIT REAL,
octet-string [9] IMPLICIT OCTETSTRING,
visible-string [10] IMPLICIT VisibleString,
generalized-time [11] IMPLICIT GeneralizedTime,
binary-time [12] IMPLICIT TimeOfDay,
bcd [13] IMPLICIT INTEGER,
booleanArray [14] IMPLICIT BITSTRING,
objid [15] IMPLICIT OBJECT IDENTIFIER,
utc-time [17] IMPLICIT UtcTime
}
```

Sequence number: 2
Date: 2/7/02 12:58:03 PM
Type: Strikeout -

```
Data ::= CHOICE {
context tag 0 is reserved for access_result
array [1] IMPLICIT SEQUENCE OF Data,
structure [2] IMPLICIT SEQUENCE OF Data,
boolean [3] IMPLICIT BOOLEAN,
bit-string [4] IMPLICIT BIT STRING,
integer [5] IMPLICIT INTEGER,
unsigned [6] IMPLICIT INTEGER,
floating-point [7] IMPLICIT FloatingPoint,
real [8] IMPLICIT REAL,
octet-string [9] IMPLICIT OCTETSTRING,
visible-string [10] IMPLICIT VisibleString,
generalized-time [11] IMPLICIT GeneralizedTime
binary-time [12] IMPLICIT TimeOfDay,
bcd [13] IMPLICIT INTEGER,
booleanArray [14] IMPLICIT BITSTRING
objid [15] IMPLICIT OBJECT IDENTIFIER
}
```

Page: 400

Sequence number: 1
Date: 2/7/02 12:58:21 PM
Type: Note: This pointer to a structure of type VAR_ACC_SPEC contains - this is not an array

Sequence number: 2
Date: 2/7/02 12:58:38 PM
Type: Strikeout - an array

Page: 417

Sequence number: 1
Date: 2/7/02 12:59:02 PM
Type: Note - This pointer to a structure of type VAR_ACC_SPEC contains - this is not an array

Sequence number: 2
Date: 2/7/02 12:59:39 PM
Type: Strikeout - an array

Page: 439

Sequence number: 1
Date: 8/12/98 3:14:49 PM -04'00'
Type: Note: mp_error_resp should be mp_err_resp

Page: 659

Sequence number: 1
Date: 2/6/02 10:53:51 AM
Type: Note:
additional data structure
/* UTC Time */
typedef struct mms_utc_time_tag
{
ST_UINT32 secs;
ST_UINT32 usec;
ST_UINT32 qflags;
} MMS_UTC_TIME;
Fields
secs This is the number of seconds since GMT midnight.
usec This is the number of microseconds of a second.
qflags These are the qualify flags - 8 least significant bits only.

Sequence number: 2
Date: 2/6/02 1:05:14 PM
Type: Note:
Common Conversion Functions
convert_btod_to_utc
Usage: This function converts MMS_BTOD relative to 1/1/1984) to the MMS_UTC_TIME (time relative to 1/1/1970). The qflags field in the MMS_UTC_TIME need to be set by the calling function. Only the MMS_BTOD6 form of the MMS_BTOD struct can be converted to the MMS_UTC_TIME.
Function Prototype: ST_RET convert_btod_to_utc (MMS_BTOD *btod, MMS_UTC_TIME *utc);
Parameters:
btod pointer to MMS_BTOD struct that should be converted to the MMS_UTC_TIME
utc pointer to MMS_UTC_TIME struct where the result of the conversion will be placed
Return:
SD_SUCCESS if function successful
SD_FAILURE otherwise

Sequence number: 3
Date: 2/6/02 1:04:42 PM
Type: Note: Common Conversion Functions
convert_utc_to_btod
Usage: This function converts MMS_UTC_TIME (time relative to 1/1/1970) to the MMS_BTOD (time relative to 1/1/1984). The form field in the MMS_BTOD is set to MMS_BTOD6 by this function.
Function Prototype: ST_RET convert_utc_to_btod (MMS_UTC_TIME *utc, MMS_BTOD *btod);
Parameters:
utc pointer to MMS_UTC_TIME struct that should be converted to the MMS_BTOD

btod pointer to MMS_BTOD struct where the result of the conversion will be placed
Return:
SD_SUCCESS if function successful
SD_FAILURE otherwise

Page: 1008

Sequence number: 1
Date: 9/12/00 3:11:19 PM -04'00'
Type: Note: MVE_VAR_LIST - This error code is also returned if the named variable list is not found.

Page: 1061

Sequence number: 1
Date: 2/14/02 2:34:49 PM
Type: Note - data structure has changed to the following:

```
# struct time_str
# {
# BOOLEAN b1;
# LONG btime1; /* Binary Time of Day - 4 byte */
# BOOLEAN b2;
# LONG btime2; /* Binary Time of Day - 6 byte */
# BOOLEAN b3;
# LONG gtime; /* Generalized Time */
# BOOLEAN b4;
# MMS_UTC_TIME utc_time; /* UTC Time */
# BOOLEAN b5;
# }
```

Sequence number: 2
Date: 2/7/02 1:00:28 PM
Type: Strikeout

```
# {
# BOOLEAN b1;
# LONG btime1; /* Binary Time Of Day - 4 byte */
# BOOLEAN b2;
# LONG btime2; /* Binary Time Of Day - 6 byte */
# BOOLEAN b3;
# LONG gtime; /* Generalized Time */
# };
```

Page: 1062

Sequence number: 1
Date: 2/6/02 12:54:42 PM
Type: Note:
TypeName = time_str | TypeDef = {(b1)Bool, (btime1) Long, (b2)Bool, (btime2)Long, (b3)Bool, (gtime)Long, (b3)Bool, (utc_time) MMS_UTC_TIME, (b5)Bool};

Sequence number: 2
Date: 2/7/02 1:00:40 PM
Type: Strikeout - TypeName = time_str | TypeDef = {(b1)Bool,(btime1)Long,(b2)Bool,(btime2)Long,(b3)Bool,(gtime)Long}

Page: 1068

Sequence number: 1
Date: 10/11/01 12:30:32 PM -04'00'
Type: Note: the parameter m_track_prev_free was deleted - do not use

Sequence number: 2
Date: 2/7/02 1:00:59 PM
Type: Strikeout # m_track_prev_free