MergeCOM-3

Advanced Integrator's Tool Kit

Reference Manual

P/N 009-13114-03 (24)

Document Revision History

Version 18

February 19, 2002

Updated page numbers, and MC_Cleanup_Memory parameters/summary/return values.

Version 19

February 19, 2002

Updated MC_NewSyntaxList

Added MC_Register_Enhanced_MemoryLog_Function and MC_Register_MemoryLog_Function.

Version 20

October 21, 2002

Updated for version 3.2.

Added MC_Get_Meta_ServiceName, updated MC_Close_Association, MC_Open_File, MC_Get(_Next)_Encapsulated_Value_To_Function, MC_Set(_next)_Value, MC_Open_Empty_Message, MC_Register_Compression_Callbacks, MC_Set_Value_Representation. Also added OF, Deflated Explicit VR Little Endian, JPEG 2000, Merge eFilm copyright information, new configuration parameters. The service table was updated to match contents of mergecom.srv.

Version 21

June 23, 2003

Updated for version 3.3.

Added MC_Get_Tags_Dict_Info, MC_Set_xxxx_Config_Value updated, additional information for MC_Register_Compression_Callbacks, updated config parm section, updated Set/Get config parm functions.

Version 23

September 15, 2004

Updated for version 3.4.0.

Added documentation of MC_Continue_Read_Message_To_Stream, MC_Set_Negotiation_Info_For_Association, MC_NewProposedServiceListAsync, MC_NewServiceWithExtInfoFromName, MC_NewServiceWithExtInfoFromUID, MC_Wait_For_Association_On_Port, MC_Wait_For_Secure_Association_On_Port, MC_Get_Listen_Socket_For_Port, and MC_Close_Listen_Port. Added updates for the AssocInfo and ServiceInfo structures. Added description of MSG_FILE_ITEM_OBJ_TRACE and COMPRESSION_RGB_TRANSFORM_FORMAT configuration options. Added new callback type in description of MC_Register_Callback_Function. Added new return value to MC_Send_Request_Message and MC_Send_Request_Message_For_Service, MC_MAX_OPERATIONS_EXCEEDED.

Version 24

March 30, 2005

Updated for version 3.5.0.

Added documentation for the new configuration options in the 3.5.0 release. Added documentation to the sample conformance statement of the new SOP Classes supported in 3.5.0.

Table of Contents

Overview

This reference manual contains a detailed description of the functionality of the MergeCOM-3 Advanced Tool Kit Library. This includes library configuration, application programmer's interface (API) specification, and a DICOM conformance statement for the tool kit.

Structure of Documentation

The MergeCOM-3 Advanced documentation is structured as shown in Figure 4.

Read Me FIRST!

The User's Manual is the foundation for all other documentation because it explains the concepts of DICOM and the Advanced Tool Kit. Before plunging into the Reference Manual or Sample Application Guide you should be comfortable with the material in the User's Manual.

The Reference Manual is where you go for detailed information on the Advanced Tool Kit. This includes the Application Programming Interface (API), tool kit configuration, the runtime object database, and status logging. The Reference Manual also includes a DICOM conformance statement for the tool kit.

The DICOM Extended Tool Kit Manual is an optional extension that describes the organization of the MergeCOM-3 DICOM Database and how to use to extend standard services and define your own private services. Tools are supplied for converting the contents of the database into the binary runtime object database.

sample applications

The Sample Application Guide describes approaches to developing specific classes of DICOM applications. It presents the pertinent information from Parts 3 or 4 of the DICOM Standard in a more readable way and in the context of the Advanced Tool Kit. The Application Guide also details the DICOM messages that can be passed between applications on the network. Also, a sample application is described and the application supplied in source form for your platform.

Platform specific information required to use the Advanced tool kit on your target platform are specified in Platform Notes. This includes supported compilers, compiler options, link options, configuration, and run-time related issues.

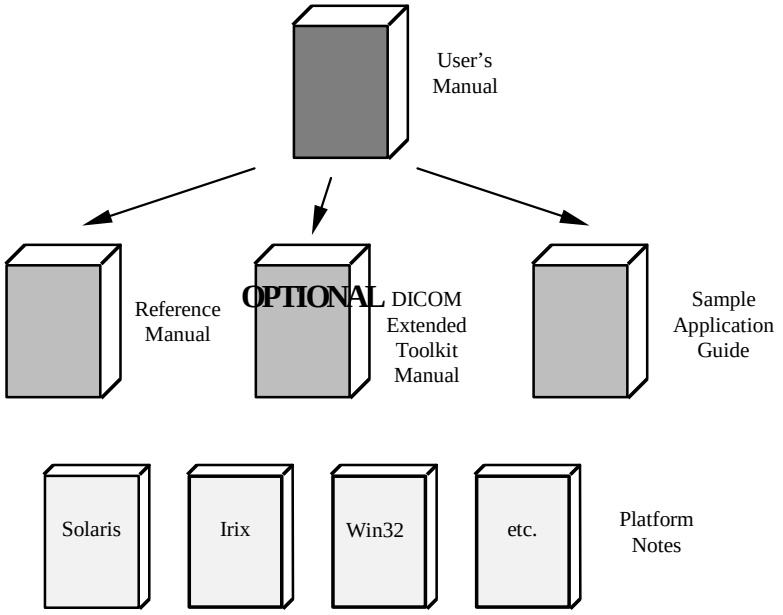

Figure 1: MergeCOM-3 Advanced Tool Kit Documentation Roadmap

Conventions

This manual follows a few formatting conventions.

Terms that are being defined are presented in **boldface**.

sample margin note

Margin notes (in the left margin) are used to highlight important points or sections of the document.

Sample commands appear in **`bold courier`** font, while sample output , source code, and function calls appear in `standard courier` font.

Hexadecimal numbers are written with a trailing H. For example 16 decimal is equivalent to 10H hexadecimal.

Performance Tuning

Portions of the document that can relate directly to the performance of your application are marked with the special margin note ***Performance Tuning***.

Configuration

Tool Kit configuration is accomplished through the use of initialization or configuration files. Initialization files, also called "ini" files, contain configuration information the tool kit will use to initialize its internal settings.

Each of the four tool kit initialization files follow the same format. The format of the initialization files is the same format that is used by others in the industry, namely Microsoft. Configuration files are broken down into sections for easier organization and grouping of parameters. Each section has a section heading enclosed in square brackets. Next, parameters are defined by putting the parameter name to the left of an equal sign and its initial value to the right. Figure 2 illustrates the format of an "ini" file.

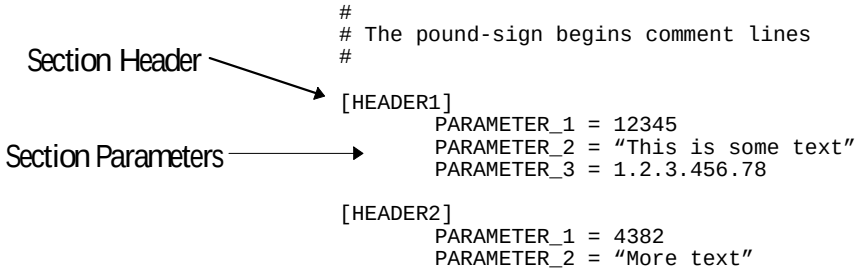

Figure 2: Format of a configuration file.

Notice that parameter names are relative to their header sections. For example, `PARAMETER_1` and `PARAMETER_2` are defined twice in the above example "ini" file. But, since each is defined in a different section, they are considered different entities.

Since the MergeCOM-3 Advanced Tool Kit is very versatile and configurable, it uses initialization files extensively. The tool kit makes use of four configuration files: the Merge Initialization File, the Library Profile, the Application Profile and the Service Profile. The *MergeCOM-3 Advanced Tool Kit Users Manual* discusses each of these files in detail.

Any application using the MergeCOM-3 Advanced library is required to initialize the library using the `MC_Library_Initialization( )` call. In most cases the first parameter passed to this function will be `NULL` and initialization of the library will occur from the configuration files described in this section.

The `genconf` utility (described later in this section) can be used to convert these configuration files into a function that can be compiled into an object file and linked directly into your application. This generated function can also be specified as the first parameter to `MC_Library_Initialization( ).` This approach should only be used in environments where the configuration is not likely to be updated in the field, where performance at initialization time is of great concern, or you running in an embedded environment (without a file system) with the embedded version of the library.

Initialization File Integration

The MergeCOM-3 Advanced Tool Kit allows initialization files to be integrated into an application at "link" time. The tool kit provides utilities to "compile" your initialization files into ANSI-C source code files that can then be compiled and linked into an object file. This object file is linked into the application thus allowing your application to access the initial values at run-time. This is a great advantage to systems that require limited disk access and the only option for embedded systems.

As with all "compiled programs", the disadvantage is that when initial values change, the files must be "recompiled" and re-linked into your application.

Performance Tuning

Generating a Static Configuration Function (`genconf`)

The `genconf` utility supplied with MergeCOM-3 Advanced allows you to convert the four configuration files (the Initialization file, the Application profile, the System Profile and the Service Profile) into an ANSI-C function that can be compiled into an object file and statically linked to your application along with the tool kit library.

In most cases this approach should not be used, since it forces you to re-run `genconf` and re-link your application whenever you wish to change the initial

configuration values. It can be used in environments where high performance at initialization time is critical, or in embedded environments. It may also be of use in end user applications to reduce installation confusion.

To use `genconf`, follow these steps:

1. Make your current directory (i.e., `cd` to) the directory that contains the Initialization File (i.e. `merge.ini`).
2. Make certain that the `MERGE_INI` environmental variable is pointing to the Initialization File.
3. Make certain that the parameters dealing with locations of other configuration files within the Initialization File are correct. For example, make sure `MERGECOM_3_PROFILE` contains the path list of the System Profile.
4. Make sure that the initial values in ALL configuration files are set to the initial values you desire. Remember, if the initial values need to be changed, you must rerun `genconf`, recompile and re-link your application.
5. Run `genconf`.

When `genconf` starts, it will display a banner similar to the following:

```
% genconf

MergeCOM-3 (tm)     Configuration Source Generator     Version 3.2.0
Copyright (c) Merge eFilm 1994-2002  All rights reserved
```

An ANSI-C source file named `mc3cfg.c` is generated that defines several structures and a configuration function called `MC_Config_Values( )`. Compile and link this source file into your application. You can then use this configuration function by specifying the `MC_Config_Values( )` function as the first parameter of the `MC_Library_Initialization( )` call.

Never modify the `mc3cfg.c` to change configuration parameters. Always modify the configuration files and rerun `genconf`. Note that the `genconf` application MUST ONLY be used with the version of the tool kit that was shipped to you. Using mixed version of the library and genconf will cause your applications to fail.

Performance Tuning

Generating a Static Dictionary Function (`gendict`)

The `gendict` utility supplied with MergeCOM-3 Advanced allows you to convert the data dictionary into an ANSI-C function that can be compiled into an object file and statically linked to your application along with the tool kit library.

In most cases this approach should not be used, since it forces you to re-run `gendict` and re-link your application whenever you wish to change the initial configuration values. It can be used in environments where high performance at initialization time is critical, or in embedded environments. It may also be of use in end user applications to reduce installation confusion.

To use `gendict`, follow these steps:

1. Make your current directory (i.e., `cd` to) the directory that contains the data dictionary (i.e. `mrgcom3.dct`).

2. Make certain that the `MERGE_INI` environmental variable is pointing to the Initialization File.
3. Make certain that the `DICTIONARY_FILE` parameter in the System Profile (i.e., `mergecom.pro`) is set correctly.
4. Run `gendict`.

When `gendict` starts, it will display a banner similar to the following:

```
% gendict

MergeCOM-3 (tm)     DICOM Dictionary Source Generator     Version 3.2.0
Copyright (c) Merge eFilm 1994-2002  All rights reserved
```

An ANSI-C source file named `mc3dict.c` is generated that defines several structures and a configuration function called `MC_Dictionary_Values`. Compile and link this source file into your application. You can then use this configuration function by specifying the `MC_Dictionary_Values` function as the second parameter of the `MC_Library_Initialization( )` call.

Never modify the `mc3dict.c` to change configuration parameters. Always modify the configuration files and rerun `genconf`.

Application Programming Interface

Overview of API

MergeCOM-3 provides functions which are used to construct and manipulate “message objects” and “file objects” as well as functions to establish a communication session with other DICOM systems to exchange DICOM messages.

“Message objects” and “file objects” are defined in the DICOM standard and in the conformance statements of DICOM applications.

The functions of the API may be divided into eleven major groups:

1. **Library initialization and reset functions**
   These must be called initially and when wishing to reset the library to its initially configured state.
2. **Run time configuration**
   These allow your application to change configurable parameters at runtime.
3. **Application Registration**
   These register your DICOM application entity with the library.
4. **Association functions**
   These deal with opening DICOM associations and querying the characteristics of a proposed or open association.
5. **Message object functions**
   These deal with the creation and manipulation of message objects.
6. **Item object functions**
   These deal with the creation and freeing of the item objects that are contained in attributes of message objects that are of value representation SQ.

7. **File object functions**
   These functions deal with the creation, manipulation, and freeing of file objects.

8. **Message, file and item object functions**
   These functions are used in the encoding, decoding, and manipulation of the attributes of messages, files and items. This includes the handling of private attributes.

9. **Message Transfer**
   These are used in sending and receiving DICOM messages over an open association.

10. **DICOMDIR Object Functions**
    These deal with the upkeep and manipulation of DICOMDIR objects.

11. **Structured Reporting Object Functions**
    These deal with the upkeep and manipulation of Structured Reporting (SR) objects.

12. **Miscellaneous Functions**.

Library Initialization and Reset

The MergeCOM-3 library initialization functions follow:

MC_Library_Initialization
MC_Library_Release
MC_Library_Reset
MC_Set_MergeINI
MC_Get_Version_String

Run Time Configuration Functions

The MergeCOM-3 run-time configuration functions follow:

MC_Set_Bool_Config_Value
MC_Set_Int_Config_Value
MC_Set_Log_Destination
MC_Set_Long_Config_Value
MC_Set_String_Config_Value
MC_Get_Bool_Config_Value
MC_Get_Int_Config_Value
MC_Get_Log_Destination
MC_Get_Long_Config_Value
MC_Get_String_Config_Value

Application Registration

The MergeCOM-3 application registration functions may be grouped as follows:

MC_Register_Application
MC_Release_Application
MC_Register_Callback_Function
MC_Register_pCallback_Function
MC_Release_Callback_Function
MC_Set_Message_Callbacks

Association Functions

The MergeCOM-3 association functions may be grouped as follows:

1. Association object creation and release.

MC_Open_Association
MC_Open_Secure_Association
MC_Close_Association

MC_Abort_Association
MC_Release_Parent_Association

MC_Wait_For_Association
MC_Wait_For_Association_On_Port
MC_Wait_For_Secure_Association
MC_Wait_For_Secure_Association_On_Port
MC_Close_Listen_Port
MC_Parse_Association_Request
MC_Accept_Association
MC_Reject_Association

MC_Set_Negotiation_Info (depracated)
MC_Set_Negotiation_Info_For_Association
MC_Get_Negotiation_Info
MC_Clear_Negotiation_Info (depracated)

MC_NewProposedServiceList
MC_NewProposedServiceListAsync
MC_NewServiceFrom… Functions
MC_NewSyntaxList
MC_FreeServiceList
MC_FreeService
MC_FreeSyntaxList

2. Query of association characteristics.

   MC_Get_Association_Info
   MC_Get_Listen_Socket
   MC_Get_Listen_Socket_For_Port
   MC_Get_First_Acceptable_Service
   MC_Get_Next_Acceptable_Service

Message Object Functions

The MergeCOM-3 message object functions may be grouped as follows:

1. Message object creation and release.

   MC_Open_Message
   MC_Free_Message
   MC_Open_Empty_Message
   MC_Set_Service_Command

2. Message duplication

   MC_Duplicate_Message

3. Clearing all message attribute values.

   MC_Empty_Message

4. Message validation functions: those dealing with insuring that a message does not violate DICOM or conformance rules.

   MC_Validate_Message
   MC_Validate_Attribute
   MC_Get_Next_Validate_Error

5. Message streaming functions: those dealing with retrieving message values from a DICOM stream or creating a DICOM stream from message values.

   MC_Message_To_Stream
   MC_Stream_To_Message

MC_Stream_To_Message_With_Offset
MC_Get_Stream_Length

6. Message attribute information functions: those functions which report which attributes are in a message and information about message attributes.

   MC_List_Message

Item Object Functions

The MergeCOM-3 message object functions may be grouped as follows:

1. Functions supporting attributes with a value representation of SQ (Sequence of Items):

   MC_Open_Item
   MC_Free_Item
   MC_Empty_Item
   MC_List_Item

File Object Functions

The MergeCOM-3 file object functions may be grouped as follows:

1. File object creation and release.

   MC_Create_File
   MC_Create_Empty_File
   MC_Free_File

2. Clearing all file attribute values.

   MC_Empty_File

3. Message validation functions: those dealing with insuring that a message does not violate DICOM or conformance rules.

   MC_Validate_File
   MC_Get_Next_Validate_Error

4. File opening, closing and information functions: those dealing with retrieving and writing DICOM file objects from or to media as well as getting information about the file object.

   MC_Open_File
   MC_Open_File_Bypass_OBOW
   MC_Open_File_Upto_Tag
   MC_Write_File
   MC_Write_File_By_Callback
   MC_Get_File_Length

5. File manipulation functions: those functions that manipulate the elements that are specific to a file.

   MC_Reset_Filename
   MC_Get_Filename
   MC_Set_File_Preamble
   MC_Get_File_Preamble

6. File transformation functions: those dealing with changing “file objects” to “message objects”.

   MC_File_To_Message
   MC_Message_To_File

7. Message attribute information functions: those functions which report which attributes are in a message and information about message attributes.

MC_List_File

Message, File and Item Object Functions

The MergeCOM-3 message and item object functions can be grouped as follows:

Message attribute information functions: those functions which report which attributes are in a message and information about message attributes.

MC_Get_First_Attribute
MC_Get_Next_Attribute
MC_Get_Attribute_Info
MC_Get_pAttribute_Info

Message attribute manipulation.

MC_Add_Standard_Attribute
MC_Add_Nonstandard_Attribute
MC_Add_Private_Attribute
MC_Add_Private_Block
MC_Byte_Swap_OBOW
MC_Delete_Attribute
MC_Delete_Current_Value
MC_Delete_Private_Attribute
MC_Delete_Private_Block
MC_Delete_Range

Functions to change the value representation of a message attribute from Unknown_VR to a valid value representation code:

MC_Set_Value_Representation
MC_Set_pValue_Representation

Setting values for standard attributes.

MC_Set_Value... Functions
MC_Set_Value_From_Function
MC_Set_Next_Value... Functions
MC_Set_Value_To_NULL
MC_Set_Next_Value_To_NULL
MC_Set_Value_To_Empty
MC_Set_Encapsulated_Value_From_Function
MC_Set_Next_Encapsulated_Value_From_Function
MC_Close_Encapsulated_Value

Setting values for private attributes.

MC_Set_pValue... Functions
MC_Set_pValue_From_Function
MC_Set_Next_pValue... Functions
MC_Set_pValue_To_NULL
MC_Set_Next_pValue_To_NULL
MC_Set_pValue_To_Empty

Getting values of standard attributes.

MC_Get_Value... Functions
MC_Get_Value_To_Function
MC_Get_Value_Count
MC_Get_Value_Length
MC_Get_Next_Value...Functions
MC_Get_Encapsulated_Value_From_Function
MC_Get_Next_Encapsulated_Value_From_Function

Getting values of private attributes.

MC_Get_pValue... Functions
MC_Get_pValue_To_Function
MC_Get_pValue_Count
MC_Get_pValue_Length
MC_Get_Next_pValue... Functions

Message Transfer

The MergeCOM-3 message transfer functions follow:

MC_Send_Request_Message
MC_Send_Request_Message_For_Service
MC_Send_Response_Message
MC_Read_Message
MC_Read_Message_To_Tag
MC_Continue_Read_Message
MC_Continue_Read_Message_To_Tag
MC_Set_Message_Transfer_Syntax
MC_Get_Message_Transfer_Syntax

DICOMDIR Object Functions

The DICOMDIR object functions may be grouped as follows:

1. DICOMDIR entity functions: those functions that deal with the manipulation of DICOMDIR entities.

   MC_Dir_Root_Entity
   MC_Dir_Next_Entity
   MC_Dir_Add_Entity
   MC_Dir_Delete_Referenced_Entity

2. DICOMDIR directory record functions: those functions that deal with the manipulation of DICOMDIR directory records.

   MC_Dir_First_Record
   MC_Dir_Next_Record
   MC_Dir_Add_Record
   MC_Dir_Delete_Record

3. DICOMDIR MRDR directory record functions: those functions which allow users to create MRDR (Multi-referenced file directory record) records and set up references to them.

   MC_Dir_Open_MRDR
   MC_Dir_Reference_MRDR
   MC_Dir_Remove_Ref_MRDR

4. DICOMDIR statistical functions: those functions which allow users to obtain counts of the over all items and entities contained within a DICOMDIR object.

   MC_Dir_Root_Count
   MC_Dir_Entity_Count
   MC_Dir_Item_Count

5. DICOMDIR sorting functions: this function which allow users to sort the record types in each DICOMDIR entity, and the record entries within each record type in order within a DICOMDIR object.

   MC_Dir_Sort

Structured Reporting Object Functions

The Structured Reporting object functions are grouped as follows:

Getting values of SR objects:

MC_ SR_Get_First_Child
MC_SR_Get_Next_Child
MC_ SR_Get_Root

Adding values to SR objects:

MC_SR_Add_Child
MC_SR_Add_Root
MC_SR_Get_Location

Converting messages to or from SR objects:

MC_Message_To_SR
MC_SR_To_Message

Deleting child SR objects:

MC_SR_Delete_Child

Miscellaneous Functions

The following miscellaneous functions are available:

MC_Error_Message
MC_Get_Message_Service
MC_Get_MergeCOM_Service
MC_Get_UID_From_MergeCOM_Service
MC_Get_Tag_Info
MC_Get_pTag_Info
MC_Get_Enum_From_Transfer_Syntax
MC_Get_Transfer_Syntax_From_Enum
MC_Set_Log_Prefix
MC_Register_Network_Capture_Callbacks

API Function Reference

The next section of this manual contains an alphabetically arranged function reference section. Each reference page(s) provide:

- A brief description of the API function,
- A synopsis of the API function which contains a list of include files required to use the function, the function prototype and a description of each function parameter,
- Remarks outlining the use of the function,
- A list of status codes returned by the function,
- and a “See also” cross reference to other functions.

MC_Abort_Association

Terminates an open association immediately.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Abort_Association (
      int* AssociationID
)
```

AssociationID Address of the association object's identification number

Remarks

MC_Abort_Association abruptly terminates the DICOM association. In effect, this function is used to tell the remote system that the association is no longer valid.

It is the responsibility of the requester of an association (client) to close a DICOM association. If a service provider (server) cannot proceed with the association it should use **MC_Abort_Association** to terminate the association. The service user (client) uses **MC_Close_Association** for normal association completion, or **MC_Abort_Association** if an error situation occurs.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	The *AssociationID* parameter was NULL.
MC_INVALID_ASSOC_ID	**AssociationID* is not a valid association object ID.
MC_LIBRARY_NOT_INITIALIZED	The library has not been properly initialized.

See Also

MC_Close_Association
MC_Open_Association
MC_Open_Secure_Association
MC_Wait_For_Association
MC_Wait_For_Secure_Association

MC_Accept_Association

Accept a remote application's request for a DICOM association.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Accept_Association (
      int AssociationID
)
```

AssociationID The association object's identification number

Remarks

If a **MC_Wait_For_Association** or **MC_Wait_For_Secure_Association** function completes normally, one of two functions must be called. **MC_Accept_Association** informs the remote application that the association can proceed. Use **MC_Reject_Association** to reject the association request.

If **MC_Accept_Association** returns **MC_ASSOCIATION_ABORTED or MC_SYSTEM_ERROR** no further calls may be made for the association.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_ASSOC_ID	*AssociationID* is not a valid association object ID.
MC_NO_REQUEST_PENDING	There are no pending association request for this *AssociationID*.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.
MC_ASSOCIATION_ABORTED	The association has been aborted.

See Also

MC_Wait_For_Association
MC_Wait_For_Secure_Association
MC_Reject_Association

MC_Add_Nonstandard_Attribute

Adds a new non-DICOM standard attribute to a message object.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Add_Nonstandard_Attribute (
      int MsgFileItemID,
      unsigned long Tag,
      MC_VR ValueRep
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message, MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
Tag	DICOM tag which identifies this attribute.
ValueRep	A code identifying the value representation of the attribute. The **MC_VR** enumerated type is defined in "mc3msg.h". The valid codes are: **AE, AS, CS, DA, DS, DT, IS, LO, LT, PN, SH, ST, TM, UI, SS, US, AT, SL, UL, FL, FD, OB, OW, OF, SQ**

Remarks

MC_Add_Nonstandard_Attribute adds a nonstandard (and non-private) attribute that does not exist in the DICOM Data Dictionary to an existing message. A message can not have nonstandard attributes and still conform to DICOM. This implies **MC_Validate_Message** will return an error when validating a message with nonstandard attributes. This function should only be used in extreme cases: when communicating with non-conformant implementations of DICOM 3.0. Private attributes should be used to extend the standard, not nonstandard attributes.

Because nonstandard attributes are not defined in the Data Dictionary, you must supply the Value Representation code: *ValueRep*. If, however, an attempt is made to add an attribute which is in the Data Dictionary, an error status will be returned.

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_TAG	The attribute with an ID of *Tag* is a standard attribute found in the Data Dictionary. Instead, use **MC_Add_Standard_Attribute**.
MC_TAG_ALREADY_EXISTS	The message already contains an attribute with an ID of *Tag*.
MC_INVALID_VR_CODE	*ValueRep* is invalid.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID, or item object ID.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment.

A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Add_Standard_Attribute
MC_Add_Private_Attribute
MC_Add_Private_Block
MC_Delete_Attribute
MC_Delete_Private_Attribute
MC_Delete_Private_Block
MC_Delete_Range

MC_Add_Private_Attribute

Adds a private DICOM attribute to a message object.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Add_Private_Attribute (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      MC_VR ValueRep
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message, MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
PrivateCode	The code string which identifies which block in the private Group is to "own" the new attribute.
Group	The number identifying the private group. It must be an odd number.
ElementByte	The number identifying this attribute within the private block.
ValueRep	A code identifying the value representation of the attribute. The **MC_VR** enumerated type is defined in "mc3msg.h". The valid codes are: **AE, AS, CS, DA, DS, DT, IS, LO, LT, PN, SH, ST, TM, UI, SS, US, AT, SL, UL, FL, FD, OB, OW, OF, SQ**

Remarks

MC_Add_Private_Attribute adds an attribute that does not exist in the Data Dictionary (a private attribute) to an existing message. A message can have private attributes and still conform to DICOM, as long as the attributes are not equivalent to and do not replace standard attributes. Private attributes are stored in an odd-number *Group* (the high-order portion of a DICOM tag). An identifying *PrivateCode* must be supplied to differentiate one owner's block of private attributes from another in any given *Group*. Since the attribute is not in the Data Dictionary, it is necessary to supply the value representation of the attribute's values: *ValueRep*.

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	*PrivateCode* was NULL.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID, or item object ID.
MC_NOT_FOUND	There is no private block identified by *PrivateCode*.
MC_INVALID_GROUP	*Group* is not an odd number.
MC_INVALID_VR_CODE	*ValueRep* is invalid.
MC_TAG_ALREADY_EXISTS	The private attribute already exists in the message object.

MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Add_Standard_Attribute
MC_Add_Private_Attribute
MC_Add_Nonstandard_Attribute
MC_Delete_Attribute
MC_Delete_Private_Attribute
MC_Delete_Private_Block
MC_Delete_Range

MC_Add_Private_Block

Creates a new DICOM private attribute block descriptor in a message object.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Add_Private_Block (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message, MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
PrivateCode	The code string which identifies the block which is being added to Group.
Group	The number identifying the private group. It must be an odd number.

Remarks

MC_Add_Private_Block adds an attribute to a given *Group* to identify a block of private attributes in the group. This must be done before **MC_Add_Private_Attribute** can be used to add attributes for the private block. The first unused tag in the range gggg0010 through gggg00FF will be assigned to this block and given *PrivateCode* as its value (gggg is *Group*).

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	*PrivateCode* was NULL.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.
MC_INVALID_GROUP	*Group* is not an odd number.
MC_INVALID_VR_CODE	*ValueRep* is invalid.
MC_TOO_MANY_BLOCKS	There already exists the maximum 240 private blocks in the group.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Add_Standard_Attribute
MC_Add_Private_Attribute
MC_Add_Nonstandard_Attribute
MC_Delete_Range
MC_Delete_Attribute
MC_Delete_Private_Attribute
MC_Delete_Private_Block

MC_Add_Standard_Attribute

Adds a new DICOM standard attribute to a message object.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Add_Standard_Attribute (
      int MsgFileItemID,
      unsigned long Tag
)
```

MsgFileItemID The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message, MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.

Tag DICOM tag which identifies the attribute.

Remarks

MC_Add_Standard_Attribute adds an attribute that exists in the DICOM Data Dictionary (a standard attribute) to an existing message. Adding this attribute may make the message non-conformant to DICOM since all necessary attributes should be determined by the service and command specified in the **MC_Open_Message** function. This implies **MC_Validate_Message** would report an error when used to validate such a message. This function might be used to construct a message to communicate with another non-conformant application.

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.
MC_TAG_ALREADY_EXISTS	The message already contains an attribute with an ID of *Tag*.
MC_INVALID_TAG	The Data Dictionary does not contain an attribute with an ID of *Tag*.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Add_Nonstandard_Attribute
MC_Add_Private_Attribute
MC_Add_Private_Block
MC_Delete_Range
MC_Delete_Attribute
MC_Delete_Private_Attribute
MC_Delete_Private_Block

MC_Byte_Swap_OBOW

Byte swaps an attribute of value representation OB, OW, or OF.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Byte_Swap_OBOW (
      int MsgFileItemID,
      unsigned long Tag
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
Tag	DICOM tag which identifies the attribute to byte swap.

Remarks

MC_Byte_Swap_OBOW byte swaps attributes of value representation OB, OW, or OF**.**

MC_Byte_Swap_OBOW can be of use when encoding or decoding 8-bit pixel data. A discussion of 8-bit pixel data is contained in the *MergeCOM-3 Advanced Tool Kit User's Manual*.

When data is OB, no operation is performed, and **MC_NORMAL_COMPLETION** is returned.

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID, or item object ID.
MC_INVALID_TAG	*Tag* is not a valid tag in *MsgFileItemID*.
MC_EMPTY_VALUE	The attribute has no value assigned to it.
MC_INCOMPATABLE_VR	The VR of *Tag* was not OB, OW, OL or OF.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Set_Value_From_Function **MC_Get_Value_To_Function**

MC_Cleanup_Memory

Releases memory allocated but not in use by the MergeCOM-3 library.

Synopsis

```
#include "mergecom.h"
MC_STATUS MC_Cleanup_Memory (int Timeout)
```

Timeout	Maximum time, in seconds, that this function should attempt to cleanup memory.

Remarks

MC_Cleanup_Memory allows an application to reduce the amount of memory consumed by the toolkit. The Merge toolkit uses an internal memory management system to increase performance. This increase in performance is at the expense of keeping allocated memory block available by not freeing them to the operating system. This call allows memory blocks not in use to be returned to the operating system.

This call can take a significant amount of time if a large amount of memory is to be freed. The timeout can be used to make incremental improvements in the memory usage. Calls to this function pick up where the previous call left off. If the timeout expires before the function is complete, MC_TIMEOUT is returned

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_LIBRARY_NOT_INITIALIZED	Call made prior to library initialization
MC_TIMEOUT	The timeout expired before the entire cleanup was finished.
MC_VALUE_OUT_OF_RANGE	The timeout was less then 0.

See Also

MC_Clear_Negotiation_Info

De-registers extended negotiation information.

NOTE: Use of this call is deprecated. **MC_Set_Negotiation_Info_For_Association** and setting of extended negotiation information through service lists should be used in place of this call and **MC_Set_Negotiation_Info**.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Clear_Negotiation_Info (
      int ApplicationID,
      char* ServiceName
)
```

ApplicationID	The identification number for the registered application.
ServiceName	The name given to a valid DICOM service.

Remarks

The DICOM standard allows application entities to exchange "extended negotiation information" when establishing an association. The contents of the negotiation information must be known to both the association requestor application and the association acceptor application. Such extended negotiation is not often used for DICOM services, but some services may require it.

MC_Set_Negotiation_Info allows the caller to supply MergeCOM-3 with extended negotiation information which it will use when establishing associations.

When a **MC_Open_Association** or **MC_Open_Secure_Association** call is made, MergeCOM-3 sends any registered extended negotiation information to the association acceptor. The acceptor normally returns an updated version of the negotiation information which can be accessed using the **MC_Get_Negotiation_Info** call.

When a **MC_Wait_For_Association** or **MC_Wait_For_Secure_Association** call is made, MergeCOM-3 stores any received extended negotiation information. This information may be accessed using the **MC_Get_Negotiation_Info** call. **MC_Set_Negotiation_Info** may be used to "update" the extended negotiation information before calling **MC_Accept_Association** to accept the association. MergeCOM-3 will return any registered negotiation information to the remote application.

MC_Clear_Negotiation_Info is used to remove extended information previously registered for a service using the **MC_Set_Negotiation_Info** call.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_APPLICATION_ID	*ApplicationID* is not a valid application identifier.
MC_NULL_POINTER_PARM	*ServiceName* was NULL.
MC_UNKNOWN_SERVICE	*ServiceName* was not registered in the MergeCOM-3 configuration files.

MC_NO_INFO_REGISTERED	There was no extended negotiation information registered for *ServiceName* for this application.

See Also

MC_Get_Negotiation_Info

MC_Set_Negotiation_Info_For_Association
MC_Set_Negotiation_Info

MC_Close_Association

Gracefully shuts down an open association with a remote DICOM application.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Close_Association (
      int* AssociationID
)
```

AssociationID — The association identification number returned by **MC_Open_Association, MC_Open_Secure_Association, MC_Wait_For_Association** or **MC_Wait_For_Secure_Association**.

Remarks

MC_Close_Association gracefully shuts down the association connection and releases system resources used by the association. This function is used to end an association which has proceeded with no errors.

It is the responsibility of the requester of an association (client) to close a DICOM association. If a service provider (server) cannot proceed with the association it should use **MC_Abort_Association** to terminate the association. The service user (client) uses **MC_Close_Association** for normal association completion, or **MC_Abort_Association** if an error situation occurs.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	The *AssociationID* parameter was NULL.
MC_INVALID_ASSOC_ID	**AssociationID* is not a valid association object ID.
MC_ASSOCIATION_ABORTED	There was an error while closing the association. Error message logged.

See Also

MC_Abort_Association
MC_Open_Association
MC_Open_Secure_Association
MC_Wait_For_Association
MC_Wait_For_Secure_Association

MC_Close_Encapsulated_Value

Terminates an encapsulated value by placing the end delimiter in the stream.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Close_Encapsulated_Value (
      int MsgFileItemID
      unsigned long Tag
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
Tag	The attribute which contains encapsulated data, but no end delimiter.

Remarks

MC_Close_Encapsulated_Value places an end delimiter for encapsulated data. This should only be done when at least **MC_Set_Encapsulated_Value_From_Function** has been used to store encapsulated data in the message for the attribute, and can also be used after **MC_Set_Next_Encapsulated_Value_From_Function**.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_TRANSFER_SYNTAX	The message's transfer syntax is non-encapsulated.
MC_INVALID_MESSAGE_ID	*MessageID* is not a valid message object ID.
MC_INVALID_TAG	The message does not contain an attribute with an ID of *Tag*.

See Also

MC_Set_Encapsulated_Value_From_Function
MC_Set_Next_Encapsulated_Value_From_Function

MC_Close_Listen_Port

Stops listening for incoming DICOM connections on a port

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Close_Listen_Port (
      int Port
)
```

Port — Listen port opened by a call to **MC_Wait_For_Association**, **MC_Wait_For_Association_On_Port**, **MC_Wait_For_Secure_Association**, or **MC_Wait_For_Secure_Association_On_Port** functions.

Remarks

MC_Close_Listen_Port causes MergeCOM-3 to stop listening for incoming DICOM associations on a TCP/IP listen port. The port may be the default listen port as configured by the TCPIP_LISTEN_PORT option from **MC_Wait_For_Association** or **MC_Wait_For_Secure_Association.** It may also be a listen port specified in a call to the **MC_Wait_For_Association_On_Port** or **MC_Wait_For_Secure_Association_On_Port** functions.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_LIBRARY_NOT_INITIALIZED	The library has not been properly initialized.
MC_INVALID_PORT_NUMBER	*Port* is not a valid listen port.

See Also

MC_Wait_For_Association **MC_Wait_For_Association_On_Port**
MC_Wait_For_Secure_Association
MC_Wait_For_Secure_Association_On_Port

MC_Continue_Read_Message

Continues the reading of the current message, that arrived from the remote application, up until the end of the message. Continuing requires that a previous **MC_Read_Message_To_Tag** or **MC_Continue_Read_Message_To_Tag** function was called.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Continue_Read_Message (
      int AssociationID,
      int* MessageID,
)
```

AssociationID	The association object's identification number.
MessageID	The ID of the message to be continued up until *StopTag*.

Remarks

MC_Continue_Read_Message reads from after previous stop tag, up to the end of the message.

The message passed in must have been previously read with **MC_Read_Message_To_Tag**, or **MC_Continue_Read_Message_To_Tag** with a stop tag less then (FFFF,FFFF).

MC_Read_Message_To_Tag, or **MC_Read_Message** may be used immediately following this function to receive a new message from the remote application.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_MESSAGE_ID	*MessageID* is not a valid message object ID.
MC_NULL_POINTER_PARM	The *MessageID* parameter was NULL.
MC_INVALID_ASSOC_ID	*AssociationID* is not a valid association object ID.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.
The association is dropped if any of the following are returned:	
MC_NETWORK_SHUT_DOWN	The network connect unexpectedly dropped.
MC_ASSOCIATION_ABORTED	The association has been aborted. A message describing the error has been written to the MergeCOM-3 log file.
MC_ASSOCIATION_CLOSED	The association has been closed.
MC_INACTIVITY_TIMEOUT	A timeout occurred in the middle of receiving a message.

MC_CONFIG_INFO_ERROR	The message information file describing the message's service/command pair could not be accessed.
MC_INVALID_MESSAGE_RECEIVED	An improperly formatted DICOM message was received. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Read_Message
MC_Read_Message_To_Tag
MC_Continue_Read_Message_To_Tag
MC_Continue_Read_Message_To_Stream

MC_Continue_Read_Message_To_Stream

Continues the reading of the current message, that arrived from the remote applicate, to a callback function. Continuing requires that a previous **MC_Read_Message_To_Tag** or **MC_Continue_Read_Message_To_Tag** function was called.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Continue_Read_Message_To_Stream (
      int AssociationID,
      int* MessageID,
      void* UserInfo,
      MC_STATUS (*YourReceiveStreamFunction)()
)
```

AssociationID	The association object's identification number.
MessageID	The ID of the message to be continued.
UserInfo	Address of data which will be passed on to *YourReceiveStreamFunction* each time it is called. This may be NULL.
YourReceiveStreamFunction	Name of a function which will be called repeatedly to provide blocks of streamed DICOM message data.

The function must be prototyped as follows:

```
MC_STATUS YourReceiveStreamFunction (
      int CBmessageID,
      void* CBuserInfo,
      int CBdataSize,
      void* CBdataBuffer,
      int CBisFirst,
      int CBisLast
)
```

CBmessageID	The identifier assigned to the message object by the **MC_Open_Message** function.
CBuserInfo	Address of data which is being passed from the **MC_Continue_Read_Message_To_Stream** function. This may be NULL.
CBdataSize	The number of bytes of stream data being provided to you in *CbdataBuffer*.
CBdataBuffer	The buffer containing stream data from the message object.
CBisFirst	Is TRUE (not zero) when MergeCOM-3 is providing the first block of stream data.
CBisLast	Is TRUE (not zero) when MergeCOM-3 is providing the last block of stream data.

Remarks

MC_Continue_Read_Message_To_Stream reads from after previous stop tag until the end of the message. The message passed in must have been read with **MC_Read_Message_To_Tag**, or **MC_Continue_Read_Message_To_Tag**.

MC_Continue_Read_Message_To_Stream can be used to increase performance by not having MergeCOM-3 process a message and pass it directly to disk as it is being read off of the network. The **MC_Read_Message_To_Tag** can be used to get basic information about the message such as the SOP Instance UID of the message from the Affected SOP Instance UID tag or the SOP Class UID of the message from the Affected SOP Class UID tag. The transfer syntax that the message is encoded in can be determined from the **MC_Get_Message_Transfer_Syntax** function after the message has been initially read with **MC_Read_Message_To_Tag**.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_MESSAGE_ID	*MessageID* is not a valid message object ID or MC_Read_Message_To_Tag was not called for *MessageID*.
MC_NULL_POINTER_PARM	The *MessageID* parameter was NULL.
MC_INVALID_ASSOC_ID	*AssociationID* is not a valid association object ID.
MC_STATE_VIOLATION	*AssociationID* is not a valid association that has been accepted.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.
The association is dropped if any of the following are returned:	
MC_NETWORK_SHUT_DOWN	The network connect unexpectedly dropped.
MC_ASSOCIATION_ABORTED	The association has been aborted. A message describing the error has been written to the MergeCOM-3 log file.
MC_ASSOCIATION_CLOSED	The association has been closed.
MC_INACTIVITY_TIMEOUT	A timeout occurred in the middle of receiving a message.
MC_CONFIG_INFO_ERROR	The message information file describing the message's service/command pair could not be accessed.
MC_INVALID_MESSAGE_RECEIVED	An improperly formatted DICOM message was received. A message describing the error has been written to the MergeCOM-3 log file.
MC_CALLBACK_CANNOT_COMPLY	*YourReceiveStreamFunction* returned an error.

See Also

MC_Read_Message
MC_Continue_Read_Message
MC_Read_Message_To_Tag
MC_Get_Message_Transfer_Syntax

MC_Continue_Read_Message_To_Tag

Continues the reading of the current message, that arrived from the remote applicate, up to and including the specified tag. Continuing requires that a previous **MC_Read_Message_To_Tag** or **MC_Continue_Read_Message_To_Tag** function was called.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Continue_Read_Message_To_Tag (
      int AssociationID,
      unsigned long StopTag,
      int* MessageID,
)
```

AssociationID	The association object's identification number.
StopTag	The tag to stop a read from the remote application.
MessageID	The ID of the message to be continued up until *StopTag*.

Remarks

MC_Continue_Read_Message_To_Tag reads from after previous stop tag, up to and including *StopTag*.

The message passed in must have been read with **MC_Read_Message_To_Tag**, or **MC_Continue_Read_Message_To_Tag** with a stop tag less then *StopTag*.

The message returned must be read until the last tag (FFFF,FFFF) either explicitly with **MC_Continue_Read_Message_To_Tag**, or by calling **MC_Continue_Read_Message** before using **MC_Read_Message_To_Tag**, or **MC_Read_Message** again.

MC_Continue_Read_Message_To_Tag may be called multiple times with increasing stop tags.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_MESSAGE_ID	*MessageID* is not a valid message object ID.
MC_INVALID_TAG	*StopTag* was within the command set (less then (0001,0000)), or the tag requested has already been read in.
MC_MUST_CONTINUE_BEFORE_READING	A previous message was not finished before this call
MC_NULL_POINTER_PARM	The *MessageID* parameter was NULL.
MC_INVALID_ASSOC_ID	*AssociationID* is not a valid association object ID.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

The association is dropped if any of the following are returned:

MC_NETWORK_SHUT_DOWN	The network connect unexpectedly dropped.
MC_ASSOCIATION_ABORTED	The association has been aborted. A message describing the error has been written to the MergeCOM-3 log file.
MC_ASSOCIATION_CLOSED	The association has been closed.
MC_INACTIVITY_TIMEOUT	A timeout occurred in the middle of receiving a message.
MC_CONFIG_INFO_ERROR	The message information file describing the message's service/command pair could not be accessed.
MC_INVALID_MESSAGE_RECEIVED	An improperly formatted DICOM message was received. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Read_Message
MC_Read_Message_To_Tag
MC_Continue_Read_Message
MC_Continue_Read_Message_To_Stream

MC_Create_Empty_File

Creates a new empty file object

Synopsis

```
#include "mc3media.h"

MC_STATUS MC_Create_Empty_File (
      int* FileIDPtr,
      char* Filename
)
```

FileIDPtr	Upon successful completion, the file object identifier is returned here.
Filename	String filename to be associated with this file object.

Remarks

The **MC_Create_Empty_File** function creates a "file object" which contains no attributes. The resulting file object is given an identification number which is returned in **FileIDPtr*. All functions dealing with this file must provide this file ID number.

The opened file object is not associated with any particular DICOM service or command. If the file object is going to be converted to a message and sent to a network partner, or if **MC_Validate_File** is to be called for the file object, **MC_Set_Service_Command** must be called first to associate the file object with a given DICOM service and command.

If a file is opened using **MC_Create_Empty_File**, it is not necessary to add attributes to the file object before setting attribute values. If one of the set value functions (e.g. **MC_Set_Value_From_String**) is used for an attribute, the attribute will automatically be added to the file object before the value is set.

NOTE: This is not the case if a file object is opened with **MC_Create_File** or if **MC_Set_Service_Command** is called for the file. In these cases the file is associated with a given service/command pair and attributes other than those associated with that service and command must be explicitly added to the file before setting values for the added attributes

---

If a file object generated by this function is to be used as a DICOMDIR, the **MC_Set_Service_Command** function must be called for the object in order for the **MC_Dir_...** functions to work.

NOTE: **MC_Set_Service_Command** will not parse attributes contained in a file object. If directory records are contained in a file created with **MC_Create_Empty_File** and **MC_Set_Service_Command** is called on the object, the **MC_Dir_...** functions will not recognize the directory records.

The DICOM file prefix for the new file object is set to "DICM". All bytes in the file preamble are set to 00H.

Reference the "DICOM V3.0 Standard, Final Text - October 29, 1993" for more information about commands associated with messages and files.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.

MC_NULL_POINTER_PARM	The *FileIDPtr* or *Filename* parameter was NULL.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment.

See Also

MC_Create_File
MC_Open_Item
MC_Open_Message
MC_Reset_Filename

MC_Create_File

Creates a new file object

Synopsis

```
#include "mc3media.h"

MC_STATUS MC_Create_File (
      int* FileIDPtr,
      char* Filename,
      char* ServiceName,
      MC_COMMAND Command
)
```

FileIDPtr	Upon successful completion, the file object identifier is returned here.
Filename	Filename associated with file object.
ServiceName	String name of a DICOM service to be associated with this message object. Name are found in the file mergecom.srv.
Command	The **MC_COMMAND** enumerated values are defined in "mc3msg.h".

Remarks

The **MC_Create_File** function creates a "file object" which contains all the attributes of a DICOM file which will be used for the given *ServiceName* and *Command*. The resulting file object is given an identification number which is returned in **FileIDPtr*. All functions dealing with this file must provide this file ID number. The *ServiceName* and *Command* are used to access configuration information which describes the attributes of the file. If such configuration information is not available, an empty file object is created and a warning message is logged.

This function can also be used to create a DICOMDIR object. When the service is set for a DICOMDIR, MergeCOM-3 creates the internal data structures needed to manipulate a DICOMDIR. An empty root directory entity is created in the new DICOMDIR object and the root directory record offset attributes are initialized.

MC_Create_File generates in each file object created the DICOM File Meta Information attributes used by DICOM media services. The user is responsible for filling in these attributes. The DICOM prefix for the file is set to "DICM". All bytes in the file preamble are set to 00H.

MC_Create_Empty_File should be used if the service and command are not yet known, or if there is no need to validate that values will be set only for attributes assigned to a given service/command pair.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	The *FileIDPtr*, *Filename*, or *ServiceName* parameter was NULL.
MC_INVALID_COMMAND	*Command* is not a supported command.
MC_MSGFILE_ERROR	An error occurred attempting to access the configuration data file.

MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Create_Empty_File
MC_Open_Item
MC_Open_Message

MC_Delete_Attribute

Removes an attribute from a message object.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Delete_Attribute (
      int MsgFileItemID,
      unsigned long Tag
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
Tag	DICOM tag which identifies the attribute.

Remarks

MC_Delete_Attribute removes an attribute (standard or nonstandard) from an existing message. Removing this attribute may make the message non-conformant to DICOM since all necessary attributes are determined by the service and command provided by **MC_Open_Message**. This implies **MC_Validate_Message** might report an error when used to validate such a message. This function could be used to clean up a message from a non-conformant application.

NOTE: If the attribute has a value representation of SQ, the item objects identified by the attribute's values are automatically freed (by calling **MC_Free_Item**). If the freed item object is in use as a value in any other sequence of items, the value in those sequences is cleared.

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_TAG	The message does not contain an attribute with an ID of *Tag*.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.

See Also

MC_Add_Standard_Attribute
MC_Add_Private_Attribute
MC_Add_Private_Block
MC_Add_Nonstandard_Attribute
MC_Delete_Range
MC_Delete_Private_Attribute
MC_Delete_Private_Block

MC_Delete_Current_Value

Deletes the current value within an attributes value list

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Delete_Current_Value (
      int MsgFileItemID,
      unsigned long Tag
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
Tag	DICOM tag which identifies the attribute.

Remarks

The **MC_Delete_Current_Value** function deletes the current value within an attribute's value list. The current value is defined as the last attribute retrieved with a call to the **MC_Get_Next_Value...** or **MC_Get_Value...** functions.

MC_Delete_Current_Value will not work with attributes whose value representation is OB, OW, or OF.

NOTE: If the attribute has a value representation of SQ, the item objects identified by the attribute's value are automatically freed (by calling **MC_Free_Item**). If the freed item object is in use as a value in other sequence of items, the value in those sequences is cleared.

Performance Tuning:

Calling this function for an attribute that has a large enough value to be stored in a temporary file is not recommended for performance reasons. For this condition, all of the attribute's values are read into memory, the specific value within the attribute is deleted, and the remaining values are re-written to the temporary file.

Reference the "DICOM V3.0 Standard, Final Text - October 29, 1993" for more information about these commands.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.
MC_INVALID_TAG	The message does not contain an attribute with an ID of *Tag*.
MC_INCOMPATABLE_VR	The function was called to delete an attribute value whose value representation(VR) was OB, OW, or OF.
MC_NULL_VALUE	The attribute's value was set to NULL.
MC_EMPTY_VALUE	The attribute is empty.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Get_Value... Functions **MC_Get_Next_Value... Functions**

MC_Delete_Private_Attribute

Removes a private attribute from a message object.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Delete_Private_Attribute (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned short ElementByte
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
PrivateCode	The code string which identifies which block in the private *Group* "owns" the attribute.
Group	The number identifying the private group. It must be an odd number.
ElementByte	The number identifying this attribute within the private block.

Remarks

MC_Delete_Private_Attribute deletes an attribute that does not exist in the Data Dictionary (a private attribute) to an existing message. A message can have private attributes and still conform to DICOM, as long as the attributes are not equivalent to and do not replace standard attributes. Private attributes are stored in an odd-number *Group* (the high-order portion of a DICOM tag). An identifying *PrivateCode* must be supplied to differentiate one owner's block of private attributes from another in any given *Group*. The block number assigned to *PrivateCode* along with the *ElementByte* form the effective tag of the private attribute

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	*PrivateCode* was NULL.
MC_INVALID_TAG	The private attribute does not exist in the message object.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.
MC_INVALID_GROUP	*Group* was not an odd number.
MC_NOT_FOUND	The private block identified by *PrivateCode* was not in *Group*.

See Also

MC_Add_Standard_Attribute
MC_Add_Private_Attribute
MC_Add_Private_Block
MC_Add_Nonstandard_Attribute
MC_Delete_Attribute
MC_Delete_Range
MC_Delete_Private_Block

MC_Delete_Private_Block

Removes a private block descriptor and its attributes from a message object.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Delete_Private_Block (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
PrivateCode	The code string which identifies which block in the private *Group* is to be deleted.
Group	The number identifying the private group. It must be an odd number.

Remarks

MC_Delete_Private_Block removes a block of private attributes from an existing message. A block of private attributes are all attributes in the private group having the same specified *PrivateCode*. The private block's identification attribute is also removed from the message.

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	*PrivateCode* was NULL.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.
MC_INVALID_GROUP	*Group* was not an odd number.
MC_NOT_FOUND	The private block identified by *PrivateCode* was not in *Group*

See Also

MC_Add_Standard_Attribute
MC_Add_Private_Attribute
MC_Add_Private_Block
MC_Add_Nonstandard_Attribute
MC_Delete_Attribute
MC_Delete_Private_Attribute
MC_Delete_Range

MC_Delete_Range

Removes a range of attributes from a message object.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Delete_Range (
      int MsgFileItemID,
      unsigned long FirstTag,
      unsigned long LastTag
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
FirstTag	Identifier of the first attribute which is to be removed from the message object.
LastTag	Identifier of the last attribute which is to be removed from the message object.

Remarks

MC_Delete_Range removes a range of attributes (standard or nonstandard) from an existing message. Removing these attributes may make the message non-conformant to DICOM since all necessary attributes are determined by the service and command specified by **MC_Open_Message**. All attributes with tags equal to or greater than *FirstTag* and equal to or less than *LastTag* will be removed. *FirstTag* and *LastTag* need not exist in the message.

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.

See Also

MC_Add_Standard_Attribute
MC_Add_Private_Attribute
MC_Add_Private_Block
MC_Add_Nonstandard_Attribute
MC_Delete_Attribute
MC_Delete_Private_Attribute
MC_Delete_Private_Block

MC_Dir_Add_Entity

Adds a lower level directory entity to a specified DICOMDIR directory record and creates the first directory record in the new directory entity

Synopsis

```
#include "mc3media.h"

MC_STATUS MC_Dir_Add_Entity (
      int DirID,
      int ItemID,
      char* NewItemName,
      int* NewEntityID,
      int* NewItemID
)
```

DirID	The identifier assigned to this DICOMDIR object by the **MC_Create_File** or **MC_Create_Empty_File** functions.
ItemID	The identifier assigned to the record to which the lower level entity is to be added.
NewItemName	String name of the item to be associated with this item object.
NewEntityID	Upon successful completion, the entity object identifier is returned here.
NewItemID	Upon successful completion, the item object identifier of the first directory record in the new entity object is returned here.

Remarks

MC_Dir_Add_Entity creates a new lower level directory entity referenced by the item object *ItemID*, and places it in the DICOMDIR object identified by *DirID*. **MC_Dir_Add_Entity** creates the first directory record, of type *NewItemName*, within the new directory entity.

The new directory record is placed in *DirID*'s directory record sequence and all internal links to the new entity are adjusted by MergeCOM-3.

NewItemName is used to access configuration information which describes the attributes of the new directory record. If such configuration information is not available, an empty item object is created, and a warning message is logged.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_DICOMDIR_ID	The *DirID* value is not a valid DICOMDIR object ID.
MC_INVALID_ITEM_ID	The *ItemID* value is not a valid directory record object ID.
MC_NULL_POINTER_PARM	The *NewEntityID*, *NewItemID*, or *NewItemName* parameter was NULL.
MC_MSGFILE_ERROR	An error occurred attempting to access the configuration data file for the directory record.

MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.
MC_INVALID_LOWER_DIR_RECORD	The directory record type specified by *NewItemName* is an invalid lower level type for the directory record *ItemID*.
MC_BAD_DIR_RECORD_TYPE	The directory record type specified in the directory record *ItemID* is invalid.

See Also

MC_Dir_Add_Record
MC_Dir_Delete_Referenced_Entity
MC_Dir_First_Record
MC_Dir_Next_Entity
MC_Dir_Next_Record
MC_Dir_Root_Entity

MC_Dir_Add_Record

Adds a DICOMDIR directory record to a specified directory entity

Synopsis

```
#include "mc3media.h"

MC_STATUS MC_Dir_Add_Record (
      int DirID,
      int EntityID,
      char* NewItemName,
      int* NewItemID
)
```

DirID	The identifier assigned to this DICOMDIR object by the **MC_Create_File** or **MC_Create_Empty_File** functions.
EntityID	The identifier of the directory entity object in which the directory record is being added.
NewItemName	String name of the item to be associated with this item object. See mergecom.srv item table for list of names.
NewItemID	Upon successful completion, the item object identifier of the new directory record is returned here.

Remarks

MC_Dir_Add_Record creates a new directory record of type *NewItemName* and appends it to the end of the entity object specified by *EntityID*. A pointer to the new directory record is returned in *NewItemID*.

The *NewItemName* parameter is used to access configuration information which describes the attributes of the new directory record. If such configuration information is not available, an empty item object is created, and a warning message is logged.

The new directory record is placed in *DirID*'s directory record sequence attribute and all internal links are adjusted by MergeCOM-3 to reflect the new record.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_DICOMDIR_ID	The *DirID* value is not a valid DICOMDIR object ID.
MC_INVALID_ENTITY_ID	The *EntityID* value is not a valid entity object ID.
MC_NULL_POINTER_PARM	The *NewItemID* or *NewItemName* parameter was NULL.
MC_UNABLE_TO_GET_ITEM_ID	MergeCOM-3 found a corrupted record ID in a directory record offset attribute while trying to traverse the DICOMDIR object pointed to by *DirID*.
MC_MSGFILE_ERROR	An error occurred attempting to access the configuration data file for the directory record.

MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.
MC_INVALID_LOWER_DIR_RECORD	The directory record type specified by *NewItemName* is an invalid lower level type for the parent directory record of *EntityID*.
MC_BAD_DIR_RECORD_TYPE	The directory record type specified in the parent directory record of *EntityID* is invalid.

See Also

MC_Dir_Add_Entity
MC_Dir_Delete_Record
MC_Dir_Delete_Referenced_Entity
MC_Dir_First_Record
MC_Dir_Next_Entity
MC_Dir_Next_Record
MC_Dir_Root_Entity

MC_Dir_Delete_Record

Deletes a DICOMDIR directory record and any directory entities it references

Synopsis

```
#include "mc3media.h"

MC_STATUS MC_Dir_Delete_Record (
      int DirID,
      int ItemID
)
```

DirID	The identifier assigned to this DICOMDIR object by the **MC_Create_File** or **MC_Create_Empty_File** functions.
ItemID	The identifier assigned to the record to be deleted.

Remarks

MC_Dir_Delete_Record deletes the directory record pointed to by *ItemID*. The resources allocated to the item are automatically freed. All lower level directory entities referenced by that item are also freed.

If the record is the only entry in a directory entity, the directory entity object will also be deleted. If the deleted record is the last one to point to an MRDR the MRDR will be deleted. The resources allocated to the directory entities and records are automatically freed.

All internal links will be adjusted within the affected directory records by MergeCOM-3.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_DICOMDIR_ID	The *DirID* value is not a valid DICOMDIR object ID.
MC_INVALID_ITEM_ID	The *ItemID* value is not a valid record object ID.
MC_NULL_POINTER_PARM	The *DirID* parameter was NULL.
MC_UNABLE_TO_GET_ITEM_ID	MergeCOM-3 found a corrupted record ID in a directory record offset attribute while trying to traverse the DICOMDIR object pointed to by *DirID*.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Dir_Add_Entity
MC_Dir_Add_Record
MC_Dir_Delete_Referenced_Entity
MC_Dir_First_Record
MC_Dir_Next_Entity
MC_Dir_Next_Record

MC_Dir_Delete_Referenced_Entity

Deletes the lower level directory entity of a specified DICOMDIR directory record

Synopsis

```
#include "mc3media.h"

MC_STATUS MC_Dir_Delete_Referenced_Entity (
      int DirID,
      int ItemID
)
```

DirID	The identifier assigned to this DICOMDIR object by the **MC_Create_File** or **MC_Create_Empty_File** functions.
ItemID	The identifier assigned to the directory record whose next lower level directory entity is being deleted.

Remarks

MC_Dir_Delete_Referenced_Entity deletes the directory record which is referenced by ItemID. If the deleted record is the last one to point to a multiple reference directory record(MRDR) the MRDR will be deleted.

If there was no referenced entity for *ItemID*, **MC_Dir_Delete_Referenced_Entity** will return **MC_NORMAL_COMPLETION**.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_DICOMDIR_ID	The *DirID* value is not a valid DICOMDIR object ID.
MC_INVALID_ITEM_ID	The *ItemID* value is not a valid record object ID.
MC_INVALID_ENTITY_ID	The *EntityID* value is not a valid entity object ID.
MC_NULL_POINTER_PARM	The *DirID* parameter was NULL.
MC_UNABLE_TO_GET_ITEM_ID	MergeCOM-3 found a corrupted record ID in a directory record offset attribute while trying to traverse the DICOMDIR object pointed to by *DirID*.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Dir_Add_Entity
MC_Dir_Add_Record
MC_Dir_Delete_Record
MC_Dir_First_Record
MC_Dir_Next_Entity
MC_Dir_Next_Record
MC_Dir_Root_Entity

MC_Dir_Entity_Count

Returns a count of the number of items contained within a particular DICOMDIR entity.

Synopsis

```
#include "mc3media.h"

MC_STATUS MC_Dir_Entity_Count (
      int DirID,
      int EntityID,
      int* Count
)
```

DirID	The identifier assigned to this DICOMDIR object by the **MC_Create_File** or **MC_Create_Empty_File** functions.
EntityID	The identifier assigned to the DICOMDIR entity which is going to be counted.
Count	Upon successful completion, the number of items contained within a particular entity is returned here.

Remarks

The count returned by **MC_Dir_Entity_Count** is maintained internally whenever items and entities are added or deleted from a DICOMDIR. Because this count is maintained internally, there is no performance penalty associated with this function call. The entity referenced by *EntityID* can be obtained by a call to **MC_Dir_Add_Entity, MC_Dir_Next_Entity, MC_Dir_First_Record, MC_Dir_Next_Record,** or **MC_Dir_Root_Entity.**

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_DICOMDIR_ID	The *DirID* value is not a valid DICOMDIR object ID.
MC_INVALID_ITEM_ID	The *ItemID* value is not a valid directory record object ID.
MC_INVALID_ENTITY_ID	The *EntityID* value is not a valid DICOMDIR entity object ID.
MC_UNABLE_TO_GET_ITEM_ID	The *DirID* value does not contain a reference to a valid root item.
MC_NULL_POINTER_PARM	The *Count* parameter was NULL.
MC_MSGFILE_ERROR	An error occurred attempting to access the configuration data file for the directory record.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Dir_Root_Count **MC_Dir_Item_Count**

MC_Dir_First_Record

Retrieve a pointer to the first directory record in a DICOMDIR directory entity

Synopsis

```
#include "mc3media.h"

MC_STATUS MC_Dir_First_Record (
      int DirID,
      int EntityID,
      int* ItemID,
      char** ItemName,
      int* IsLast
)
```

DirID	The identifier assigned to this DICOMDIR object by the **MC_Create_File** or **MC_Create_Empty_File** functions.
EntityID	The identifier assigned to this directory entity object.
ItemID	Upon successful completion, the item object identifier of the first directory record in the directory entity *EntityID* is returned here.
ItemName	Upon successful completion, a pointer to the string name of the item associated with this item object is returned here.
IsLast	Upon successful completion, this parameter is set to true (non-zero) if the first record is also the last record in the directory entity (i.e., it is the only element)

Remarks

MC_Dir_First_Record retrieves the identifier of the first directory record in the directory entity *EntityID*.

The item type for *ItemID* is also returned in the parameter *ItemName*. This item type can be used to create another record with **MC_Dir_Add_Record** or a new record in a new entity with **MC_Dir_Add_Entity**.

If *EntityID* is an empty root directory entity, **IsLast* is set to non-zero, *ItemID* is set to 0 and *ItemName* is set to NULL.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_DICOMDIR_ID	The *DirID* value is not a valid DICOMDIR object ID.
MC_INVALID_ENTITY_ID	The *EntityID* value is not a valid entity object ID.
MC_NULL_POINTER_PARM	The *ItemID*, *ItemName*, or *IsLast* parameter was NULL.
MC_UNABLE_TO_GET_ITEM_ID	MergeCOM-3 found a corrupted record ID in a directory record offset attribute while trying to traverse the DICOMDIR object pointed to by *DirID*.
MC_EMPTY_ROOT_ENTITY	The entity requested is empty.

See Also

MC_Dir_Next_Entity
MC_Dir_Next_Record
MC_Dir_Root_Entity

MC_Dir_Item_Count

Returns a count of the number of items contained "below" a particular DICOMDIR item, including the starting item.

Synopsis

```
#include "mc3media.h"

MC_STATUS MC_Dir_Item_Count (
      int DirID,
      int ItemID,
      int* Count
)
```

DirID	The identifier assigned to this DICOMDIR object by the **MC_Create_File** or **MC_Create_Empty_File** functions.
ItemID	The identifier assigned to the DICOMDIR item the counting process is starting from.
Count	Upon successful completion, the number of items contained "below" a particular item is returned here.

Remarks

The count returned by **MC_Dir_Item_Count** is computed during this function call. Because this count is generated on the fly, there is a performance penalty associated with this function call. During this function call, each entity referenced by the given item is traversed, and the total number of items is generated.

The item referenced by *ItemID* can be obtained by a call to **MC_Dir_Add_Record, MC_Dir_First_Record** or **MC_Dir_Next_Record.**

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_DICOMDIR_ID	The *DirID* value is not a valid DICOMDIR object ID.
MC_INVALID_ITEM_ID	The *ItemID* value is not a valid directory record object ID.
MC_INVALID_ENTITY_ID	The *EntityID* value is not a valid DICOMDIR entity object ID.
MC_UNABLE_TO_GET_ITEM_ID	The *DirID* value does not contain a reference to a valid root item.
MC_NULL_POINTER_PARM	The *Count* parameter was NULL.
MC_MSGFILE_ERROR	An error occurred attempting to access the configuration data file for the directory record.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Dir_Root_Count **MC_Dir_Entity_Count**

MC_Dir_Next_Entity

Retrieves a pointer to the next lower directory entity linked to a specified DICOMDIR directory record

Synopsis

```
#include "mc3media.h"

MC_STATUS MC_Dir_Next_Entity (
      int DirID,
      int ItemID,
      int* NextEntityID,
      int* NextItemID,
      char** NextItemName,
      int* IsLast
)
```

DirID	The identifier assigned to this DICOMDIR object by the **MC_Create_File** or **MC_Create_Empty_File** functions.
ItemID	The item object identifier of the directory record to retrieve the next lower directory entity from.
NextEntityID	Upon successful completion, the entity object identifier of the next lower directory entity referenced by *ItemID* is returned here.
NextItemID	Upon successful completion, the item object identifier of the first directory record in the directory entity *NextEntityID* is returned here.
NextItemName	Upon successful completion, a pointer to the string name of the item associated with the item object *NextItemID* is returned here.
IsLast	Upon successful completion, this parameter is set to true (non-zero) if the first record in the directory entity *NextEntityID* is also the last record (i.e., it is the only element)

Remarks

MC_Dir_Next_Entity returns *NextEntityID*, the entity object identifier of the next lower directory entity linked to *ItemID*, and the identifier *NextItemID* and type *NextItemName* of the first directory record of *NextEntityID*.

The item type for *NextItemID* is also returned in *NextItemName*. This item type can be used to create another record with **MC_Dir_Add_Record** or a new record in a new entity with **MC_Dir_Add_Entity**

If *ItemID* does not contain a referenced entity, *NextEntityID* and *NextItemID* are set to zero, and ItemName is set to NULL.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_DICOMDIR_ID	The *DirID* value is not a valid DICOMDIR object ID.
MC_INVALID_ITEM_ID	The *ItemID* value is not a valid item object ID.
MC_NULL_POINTER_PARM	The *NextEntityID*, *NextItemID*, or *NextItemName* parameter was NULL.

MC_UNABLE_TO_GET_ITEM_ID	MergeCOM-3 found a corrupted record ID in a directory record offset attribute while trying to traverse the DICOMDIR object pointed to by *DirID*.

See Also

MC_Dir_First_Record
MC_Dir_Next_Record
MC_Dir_Root_Entity

MC_Dir_Next_Record

Retrieves the next directory record in a directory entity

Synopsis

```
#include "mc3media.h"

MC_STATUS MC_Dir_Next_Record (
      int DirID,
      int EntityID,
      int* ItemID,
      char** ItemName,
      int* IsLast
)
```

DirID	The identifier assigned to this DICOMDIR object by the **MC_Create_File** or **MC_Create_Empty_File** functions.
EntityID	The identifier assigned to this directory entity object.
ItemID	Upon successful completion, the item object identifier of the directory record is returned here.
ItemName	Upon successful completion, a pointer to the string name of the item associated with the item object *ItemID* is returned here.
IsLast	Upon successful completion, this parameter is set to true (non-zero) if the retrieved directory record is the last record in *EntityID*.

Remarks

MC_Dir_Next_Record retrieves the next directory record in the directory entity specified by *EntityID*. If no records have been fetched from this entity, **MC_Dir_Next_Record** behaves the same as **MC_Dir_First_Record**. If one or more records have been fetched from this entity, the next record will be returned.

IsLast is nonzero if the retrieved directory record is the last record in *EntityID*. If **MC_Dir_First_Record** is called after the last record was retrieved, **IsLast* will be non-zero, **ItemID* will be set to 0, and **ItemName* will be NULL. If *EntityID* is an empty root directory entity, **IsLast* will be non-zero, **ItemID* will be set to 0, and **ItemName* will be NULL.

The first call to **MC_Dir_Next_Record** after **MC_Dir_Root_Entity** or **MC_Dir_Next_Entity** was called for *EntityID* will return the second record in *EntityID*.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_DICOMDIR_ID	The *DirID* value is not a valid DICOMDIR object ID.
MC_INVALID_ITEM_ID	The *ItemID* value is not a valid item or record object ID.
MC_NULL_POINTER_PARM	The *ItemID*, *ItemName*, or *isLast* parameter was NULL.
MC_UNABLE_TO_GET_ITEM_ID	MergeCOM-3 found a corrupted record ID in a directory record offset attribute while

trying to traverse the DICOMDIR object pointed to by *DirID*.

See Also

MC_Dir_First_Record
MC_Dir_Next_Entity
MC_Dir_Root_Entity

MC_Dir_Open_MRDR

Creates a DICOMDIR multiple reference directory record(MRDR) and references it with a directory record

Synopsis

```
#include "mc3media.h"

MC_STATUS MC_Dir_Open_MRDR (
      int DirID,
      int ItemID,
      char* MrdrName,
      int* MrdrID
)
```

DirID	The identifier assigned to this DICOMDIR object by the **MC_Create_File** or **MC_Create_Empty_File** functions.
ItemID	The identifier of the directory record that will reference the MRDR.
MrdrName	The string name of the item to be associated with the MRDR.
MrdrID	Upon successful completion, the item object identifier of the MRDR created will be returned here.

Remarks

MC_Dir_Open_MRDR creates a new MRDR and associates it with the directory record specified by *ItemID*. The use count attribute of the MRDR is initialized and the internal links are adjusted by MergeCOM-3. To remain DICOM conformant, the parameter *MrdrName* should always be the default data dictionary name for an MRDR item.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_DICOMDIR_ID	The *DirID* value is not a valid DICOMDIR object ID.
MC_INVALID_ITEM_ID	The *ItemID* value is not a valid item or record object ID.
MC_NULL_POINTER_PARM	The *MrdrID* or *MrdrName* parameter was NULL.
MC_UNABLE_TO_GET_ITEM_ID	MergeCOM-3 found a corrupted record ID in a directory record offset attribute while trying to traverse the DICOMDIR object pointed to by *DirID*.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Dir_Reference_MRDR **MC_Dir_Remove_Ref_MRDR**

MC_Dir_Reference_MRDR

Causes a DICOMDIR directory record to reference a multiple reference directory record(MRDR)

Synopsis

```
#include "mc3media.h"

MC_STATUS MC_Dir_Reference_MRDR (
      int DirID,
      int ItemID,
      int MrdrID
)
```

DirID	The identifier assigned to this DICOMDIR object by the **MC_Create_File** or **MC_Create_Empty_File** functions.
ItemID	The item object identifier of the directory record that will reference the MRDR.
MrdrID	The identifier of the MRDR that *ItemID* will reference.

Remarks

MC_Dir_Reference_MRDR references an MRDR with a standard directory record. MergeCOM-3 adjusts the use count attribute of the MRDR and the internal links within the affected directory records. The user is responsible for filling the other attributes within the MRDR and the directory record *ItemID*.

If *ItemID* already references another MRDR, **MC_Dir_Reference_MRDR** deletes that reference.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_DICOMDIR_ID	The *DirID* value is not a valid DICOMDIR object ID.
MC_INVALID_ITEM_ID	The *ItemID* value is not a valid item object ID.
MC_INVALID_MRDR_ID	The *MrdrID* value is not a valid MRDR object ID.
MC_UNABLE_TO_GET_ITEM_ID	While trying to traverse the DICOMDIR object pointed to by *DirID*, MergeCOM-3 found an invalid item ID contained in a directory record offset attribute.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Dir_Open_MRDR **MC_Dir_Remove_Ref_MRDR**

MC_Dir_Remove_Ref_MRDR

Removes a reference from a DICOMDIR directory record to a multiple reference directory record(MRDR)

Synopsis

```
#include "mc3media.h"

MC_STATUS MC_Dir_Remove_Ref_MRDR (
      int DirID,
      int ItemID
)
```

DirID — The identifier assigned to this DICOMDIR object by the **MC_Create_File** or **MC_Create_Empty_File** functions.

ItemID — The item object identifier of the directory record that the reference to an MRDR will be removed from.

Remarks

MC_Dir_Remove_Ref_MRDR removes the reference from a directory record to an MRDR. MergeCOM-3 decrements the use count attribute of the MRDR. If the use count equals zero, the MRDR will be automatically freed.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_DICOMDIR_ID	The *DirID* value is not a valid DICOMDIR object ID.
MC_INVALID_ITEM_ID	The *ItemID* value is not a valid item or record object ID.
MC_UNABLE_TO_GET_ITEM_ID	While trying to traverse the DICOMDIR object pointed to by *DirID*, MergeCOM-3 found an invalid item ID contained in a directory record offset attribute.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Dir_Open_MRDR **MC_Dir_Reference_MRDR**

MC_Dir_Root_Count

Returns the total number of items contained within a DICOMDIR.

Synopsis

```
#include "mc3media.h"

MC_STATUS MC_Dir_Root_Count (
      int DirID,
      int* Count
)
```

DirID	The identifier assigned to this DICOMDIR object by the **MC_Create_File** or **MC_Create_Empty_File** functions.
Count	Upon successful completion, the total number of items contained with a DICOMDIR is returned here.

Remarks

The count returned by **MC_Dir_Root_Count** is maintained internally whenever items and entities are added or deleted from a DICOMDIR. Because this count is maintained internally, there is no performance penalty associated with this function call.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_DICOMDIR_ID	The *DirID* value is not a valid DICOMDIR object ID.
MC_UNABLE_TO_GET_ITEM_ID	The *DirID* value does not contain a reference to a valid root item.
MC_NULL_POINTER_PARM	The *Count* parameter was NULL.
MC_MSGFILE_ERROR	An error occurred attempting to access the configuration data file for the directory record.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Dir_Entity_Count **MC_Dir_Item_Count**

MC_Dir_Root_Entity

Retrieves a pointer to the root entity of a specified DICOMDIR

Synopsis

```
#include "mc3media.h"

MC_STATUS MC_Dir_Root_Entity (
      int DirID,
      int* RootEntityID,
      int* ItemID,
      char** ItemType,
      int* IsLast
)
```

DirID	The identifier assigned to this DICOMDIR object by the **MC_Create_File** or **MC_Create_Empty_File** functions.
RootEntityID	Upon successful completion, the root entity object identifier is returned here.
ItemID	Upon successful completion, the item object identifier of the first directory record in the root entity is returned here.
ItemName	Upon successful completion, a pointer to the string name of the item associated with the item object *ItemID* is returned here.
IsLast	Upon successful completion, this parameter is set to true (non-zero) if the first record in the directory entity *RootEntityID* is also the last record (i.e., it is the only element)

Remarks

MC_Dir_Root_Entity returns the identifier of the root entity of the DICOMDIR. It also returns the identifier, *ItemID*, and type, *ItemName*, of the first directory record in *RootEntityID*.

The item type returned in *ItemName* can be used to create another record with **MC_Dir_Add_Record** or a new record in a new entity with **MC_Dir_Add_Entity** If the root entity is empty, *ItemID* is set to zero, and *ItemName* is set to NULL.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_DICOMDIR_ID	The *DirID* value is not a valid DICOMDIR object ID.
MC_NULL_POINTER_PARM	The *RootEntityID*, *ItemID*, or *ItemName* parameter was NULL.
MC_UNABLE_TO_GET_ITEM_ID	While trying to traverse the DICOMDIR object pointed to by *DirID*, MergeCOM-3 found an invalid item ID contained in a directory record offset attribute.

See Also

MC_Dir_First_Record
MC_Dir_Next_Entity
MC_Dir_Next_Record

MC_Dir_Sort

Sorts a DICOMDIR. See remarks section for specific ordering.

Synopsis

```
#include "mc3media.h"

MC_STATUS MC_Dir_Sort (
      int DirID
)
```

DirID The ID number of a DICOMDIR to sort

Remarks

Each DICOMDIR Entity will first sort by record type in the following order:

PATIENT
STUDY
SERIES
IMAGE
OVERLAY
MODALITY LUT
VOI LUT
CURVE
TOPIC
VISIT
RESULTS
INTERPRETATION
STUDY COMPONENT
STORED PRINT
RT DOSE
RT STRUCTURE SET
RT PLAN
RT TREAT RECORD
PRESENTATION
WAVEFORM
SR DOCUMENT
PRIVATE
MRDR

Within each record type the record entries shall be sorted in the in the following order:

PATIENT

Sorted first by Patient Name (0010,0010) ascending alphabetically if present

If Patient Name not present these records shall be placed at the end of the list.

Second the records within each unique patient name shall be sorted by the Patient ID.

STUDY

Sorted first by Study Date ascending

Sorted second by Study Time ascending within each Study Date

SERIES

Sorted first by Modality ascending alphabetically.

Sorted within each unique Modality by series number.

IMAGE

Sorted by instance number.

OVERLAY

Sorted by overlay number.

MODALITY LUT

Sorted by Lookup table number.

VOI LUT

Sorted by lookup table number.

CURVE

Sorted by curve number.

TOPIC

Sorted by Topic Title

VISIT

Sorted by Admitting Date if present. Those records without an admitting date shall be grouped as entered by the application as the end of the list.

RESULTS

Instance Creation Date if present. Those records without an instance creation date will be placed at the end of the list in the order as supplied by the application.

INTERPRETATION

Sorted by the Interpretation transcription date.

STUDY COMPONENT

Shall be sorted by Modality

STORED PRINT

Sorted by Instance Number if present.

RT DOSE

Sorted by Instance Number

RT STRUCTURE SET

Sorted by Instance Number

RT PLAN

Sorted by Instance Number

RT TREAT RECORD

Sorted by Instance Number

PRESENTATION

First sorted by Presentation Creation Date ascending

Second sorted by Presentation Creation Time ascending within each unique date

WAVEFORM

Sorted first sorted by Content Date ascending.

Second sorted by Content Time ascending within each unique Content Date.

SR DOCUMENT

Sorted first sorted by Content Date ascending.

Second sorted by Content Time ascending within each unique Content Date.

Sorted third by Instance number.

PRIVATE

Not Sorted

MRDR

Not Sorted.

Calls to the library to retrieve records will receive them in sorted order after calling this function. Also, when the DICOMDIR is written to a file, the records will be streamed sorted, depth first. The sort is not maintained if new items are added.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_LIBRARY_NOT_INITIALIZED	The library has not been properly initialized.
MC_INVALID_DICOMDIR_ID	*DirID* value is not a valid DICOMDIR object ID.
MC_INVALID_LOWER_DIR_RECORD	While trying to traverse the DICOMDIR object pointed to by *DirID*, MergeCOM-3 found an invalid item ID contained in a lower-level directory entity attribute.
MC_UNABLE_TO_GET_ITEM_ID	While trying to traverse the DICOMDIR object pointed to by *DirID*, MergeCOM-3 found an invalid item ID contained in a directory record offset attribute.
MC_VALUE_OUT_OF_RANGE	While trying to traverse the DICOMDIR object pointed to by *DirID*, MergeCOM-3 found an item ID that contained an attribute with an invalid value.

MC_SYSTEM_ERROR	An unrecoverable error occurred (i.e. no memory available, no disk space, etc). A log has been written to the log file describing the problem.

See Also

MC_Duplicate_Message

Creates a message identical to the source message. If the transfer syntax on the destination differs from the source, the new message will be changed to the new transfer syntax. This includes changing from one compression transfer syntax to another. This function will also register the compressor/decompressor with the new message.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Duplicate_Message (
      int SourceMsgID,
      int* DestMsgID,
      TRANSFER_SYNTAX DestMsgTransferSyntax,
      MC_STATUS (*YourDestMsgCompressionCallback),
      MC_STATUS (*YourDestMsgDecompressionCallback)
)
```

SourceMsgID	The identifier assigned to the source message by the MC_Open_Message or MC_Open_Empty_Message function.
DestMsgID	Upon successful completion, the message object identifier is returned here.
DestMsgTransferSyntax	The DICOM transfer syntax to be used for the destination message. One of the enumerated TRANSFER_SYNTAX types defined in "mc3msg.h".
YourDestMsgCompressionCallback	Name of a function to be registered as the Compression callback function for the destination message. To use the built in compressor, set this to **MC_Standard_Compressor**.
YourDestMsgDecompressionCallback	Name of a function to be registered as the Decompression callback function for the destination message. To use the built in decompressor, set this to **MC_Standard_Decompressor**.

The functions must be prototyped as follows:

```
MC_STATUS YourDestMsg(De)CompressionCallback (
      int CBmsgID,
      void** CBcontext,
      unsigned long CBdataLength,
      void* CBdataValue,
      unsigned long* CBoutdataLength,
      void** CBoutdataValue,
      int CBisFirst,
      int CBisLast
      int CBrelease
)
```

CBmsgID	The identifier assigned to this message object by the **MC_Open_Message**, **MC_Open_Empty_Message** functions.
CBcontext	A data structure that contains user data and has to be preserved between compression calls. The first call to the callback function should initialize this structure.

CBdataLength	The length of the incoming data pointed to by CBdataValue.
CBdataValue	Pointer to incoming data.
CBoutdataLength	The length of the outgoing data pointed to by CBoutdataValue.
CBoutdataValue	Pointer to the outgoing data.
CBisFirst	This is TRUE (not zero) the first time ***YourDestMsg(De)CompressionCallback*** is being called.
CBisLast	This is TRUE (not zero) the last time ***YourDestMsg(De)CompressionCallback*** is being called.
CBrelease	This is TRUE (not zero) if the callback should release all the context memory and return.

Remarks

Registering **MC_Standard_Compressor** and/or **MC_Standard_Decompressor** for the destination message will use the standard compressor/decompressor built into the library. You may also register your own (de)compression callbacks for the destination message. The source message can have a decompressor/compressor registered prior to duplication by calling **MC_Register_Compression_Callbacks** for the source message.

The **MC_Duplicate_Message** function is used to duplicate a message, however the transfer syntax may be different on the destination message. The following syntaxes may be switched when using the built in compressor/decompressor **MC_Standard_Compressor/MC_Standard_Decompressor**:
IMPLICIT_LITTLE_ENDIAN
IMPLICIT_BIG_ENDIAN
EXPLICIT_LITTLE_ENDIAN
EXPLICIT_BIG_ENDIAN
DEFLATED_EXPLICIT_LITTLE_ENDIAN
JPEG_BASELINE
JPEG_EXTENDED_2_4
JPEG_LOSSLESS_HIER_14
JPEG_2000
JPEG_2000_LOSSLESS_ONLY

NOTE: Only the transfer syntax may be different when using the built in compressor/decompressor **MC_Standard_Compressor/MC_Standard_Decompressor.**

The destination message should be freed via MC_Free_Message when it is no longer needed.

Performance Tuning:

If an encapsulated transfer syntax is being duplicated into a message with the same encapsulated transfer syntax, it is possible to bypass the compressor/decompressor by ensuring that ALL of the following cases are true:
1) The source message does not have a decompressor registered
2) A decompressor is not being passed in for the destination message.
3) The transfer syntaxes are identical

YourDestinationMsg(De)CompressionCallback
Each block of data is returned in *CBdataBuffer* and the number of bytes in the block is specified by *CBdataSize*.

CBisFirst is set to TRUE the first time ***YourDesMsg(De)CompressionCallback*** is called. This provides a clear mechanism for the function to know it is being called the first time.

CBisLast is set to TRUE the last time ***YourDestMsg(De)CompressionCallback*** is called. This provides a clear mechanism for the function to know it is being called the last time.

YourDestMsg(De)CompressionCallback must return **MC_NORMAL_COMPLETION** if all went well. If not, it must return **MC_CANNOT_COMPLY**. Both of these are defined in "mc3msg.h".

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_COMPRESSOR_REQUIRED	A compression function must be passed in for the destination message.
MC_DECOMPRESSOR_REQUIRED	A decompression function must be registered with the source message.
MC_INVALID_MESSAGE_ID	*SourceMsgID* is not a valid message object ID.
MC_CALLBACK_CANNOT_COMPLY	***YourDestMsg(De)CompressionCallback*** returned with MC_CANNOT_COMPLY, or the callback function registered for this attribute, and providing data, returned MC_CANNOT_COMPLY.

See Also

MC_Free_Message **MC_Register_Compression_Callbacks**

MC_Empty_File

Sets all the attributes in a DICOM file to empty

Synopsis

```
#include "mc3media.h"

MC_STATUS MC_Empty_File (
      int FileID
)
```

FileID The identifier assigned to this item object by the **MC_Create_File** function.

Remarks

MC_Empty_File sets the value of each attribute in the file identified by *FileID* to empty" (i.e. value length of zero). This is equivalent to calling **MC_Set_Value_To_Empty** for each attribute in an item. This function is useful for client applications which reuse a file and want to insure there are no attribute values remaining from the last instance of the file.

NOTE: If the file contains an attribute with a value representation of SQ, setting the file values to empty causes the SQ attribute's sequence item objects to be freed.

The DICOM prefix for the file is set to "DICM". All bytes in the file preamble are set to 00H.

When the file object identified by *FileID* was a DICOMDIR object, after each attribute is set to empty, an empty root directory entity is created in the object and the root directory record offset attributes are initialized. If the file object was not a DICOMDIR object, but it is to be used as a DICOMDIR object, **MC_Set_Service_Command** should be called for the file object before the **MC_Dir_...** functions are called for the object.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_FILE_ID	The *FileID* value is not a valid file object ID.

See Also

MC_Empty_Message
MC_Empty_Item
MC_Set_Value_To_Empty
MC_Set_pValue_To_Empty

MC_Empty_Item

Sets all the attributes in an item to empty.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Empty_Item (
      int ItemID
)
```

ItemID The identifier assigned to this item object by the **MC_Open_Item** function.

Remarks

MC_Empty_Item sets the value of each attribute in the item identified by *ItemID* to "empty" (i.e. value length of zero). This is equivalent to calling **MC_Set_Value_To_Empty** for each attribute in an item. This function is useful for client applications which reuse an item and want to insure there are no attribute values remaining from the last instance of the item.

NOTE: If the item contains an attribute with a value representation of SQ, setting the item values to empty causes the SQ attribute's sequence item objects to be freed (i.e. **MC_Free_Item** is automatically called for any items used by the item being emptied).

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_ITEM_ID	The *ItemID* value is not a valid item object ID.

See Also

MC_Empty_Message
MC_Set_Value_To_Empty
MC_Set_pValue_To_Empty

MC_Empty_Message

Sets all the attributes in a message to empty.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Empty_Message (
      int MessageID
)
```

MessageID The identifier assigned to this object by the MC_Open_Message function or the MC_Open_Item function.

Remarks

MC_Empty_Message sets the value of each attribute in the message identified by *MessageID* to "empty" (i.e. value length of zero). This is equivalent to calling **MC_Set_Value_To_Empty** for each attribute in a message. This function is useful for client applications which reuse a message and want to insure there are no attribute values remaining from the last instance of the message.

NOTE: If the message contains an attribute with a value representation of SQ, setting the message values to empty causes the SQ attribute's sequence item objects to be freed (i.e. **MC_Free_Item** is automatically called for any items used by the message being emptied).

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_MESSAGE_ID	The *MessageID* value is not a valid message object ID.

See Also

MC_Empty_Item
MC_Set_Value_To_Empty
MC_Set_pValue_To_Empty

MC_Error_Message

Returns a string description of a MergeCOM-3 status code.

Synopsis

```
#include "mc3msg.h"

char* MC_Error_Message (
      MC_STATUS StatusCode
)
```

StatusCode A MergeCOM-3 status code.

Remarks

It is often useful to log a descriptive message when an error status is received. **MC_Error_Message** returns a string message describing a given **MC_STATUS** code.

Return Value

Pointer to the descriptive string.

MC_File_To_Message

Converts a file object into a message object.

Synopsis

```
#include "mc3media.h"

MC_STATUS MC_File_To_Message (
      int FileID
)
```

FileID The identifier assigned to this object by the **MC_Create_Empty_File** or **MC_Create_File** function.

Remarks

MC_File_To_Message changes the file object pointed to by *FileID* into a message object. In the process, the DICOM File Meta Information attributes are removed from the object, and the "command type" attributes used by most DICOM services are added to the new message object. While MergeCOM-3 sets the values of many of these "command type" attributes automatically, some services require the application to set them. (Refer to the description of **MC_Send_Request_Message** for more information.)

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_FILE_ID	The *FileID* value is not a valid file object ID.

See Also

MC_Message_To_File

MC_Free_Item

Releases a MergeCOM-3 item object.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Free_Item (
      int* ItemID
)
```

ItemID A pointer to the identifier assigned to this item object by the **MC_Open_Item** function.

Remarks

The **MC_Free_Item** function frees the system resources used by the item object. The variable pointed to by *ItemID* is set to -1 (an invalid item ID).

NOTE: If the item object being freed contains an attribute with a value representation of SQ, freeing the item causes the SQ attribute's sequence item objects to be freed also (i.e. **MC_Free_Item** is automatically called for any items used by the item identified by *ItemID*). If any of the items thus freed also contain sequences of items, those item in turn are recursively freed.).

NOTE: The **ELIMINATE_ITEM_REFERENCES** configuration value will determine if a freed item object that is in use as a value in another sequence of items should be cleared.

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_ITEM_ID	The **ItemID* value is not a valid item object ID.
MC_NULL_POINTER_PARM	*ItemID* was NULL.

See Also

MC_Free_File
MC_Free_Message
MC_Open_Item

MC_Free_File

Releases a MergeCOM-3 file object

Synopsis

```
#include "mc3media.h"

MC_STATUS MC_Free_File (
      int* FileID
)
```

FileID A pointer to the identifier assigned to this object by the **MC_Create_File** or **MC_Create_Empty_File** function.

Remarks

The **MC_Free_File** function frees the system resources used by the file object. The variable pointed to by *FileID* is set to -1 (an invalid file ID).

NOTE: If the file contains an attribute with a value representation of SQ, freeing the file causes the SQ attribute's sequence item objects to be freed also. If any of the items thus freed also contain sequences of items, those items in turn are recursively freed.

NOTE: The **ELIMINATE_ITEM_REFERENCES** configuration value will determine if a freed item object that is in use as a value in another sequence of items should be cleared.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_FILE_ID	The **FileID* value is not a valid file object ID.
MC_NULL_POINTER_PARM	*FileID* was NULL.

See Also

MC_Free_Item
MC_Free_Message

MC_Free_Message

Releases a MergeCOM-3 message object.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Free_Message (
      int* MessageID
)
```

MessageID A pointer to The identifier assigned to this object by the MC_Open_Message function or the MC_Open_Item function.

Remarks

The **MC_Free_Message** function frees the system resources used by the message object. The variable pointed to by *MessageID* is set to -1 (an invalid message ID);

NOTE: If the message object being freed contains an attribute with a value representation of SQ, freeing the message causes the SQ attribute's sequence item objects to be freed also (i.e. **MC_Free_Item** is automatically called for any items used by the message identified by *MessageID*).

NOTE: If any of the items thus freed also contain sequences of items, those item in turn are recursively freed.

NOTE: The **ELIMINATE_ITEM_REFERENCES** configuration value will determine if a freed item object that is in use as a value in another sequence of items should be cleared.

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_MESSAGE_ID	The **MessageID* value is not a valid message object ID.
MC_NULL_POINTER_PARM	*MessageID* was NULL.

See Also

MC_Open_Message
MC_Free_Item
MC_Free_File

MC_FreeService

Releases a MergeCOM-3 service that was dynamically created using MC_NewService.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_FreeService (
      char* ServiceName
)
```

ServiceName Service name assigned by the application in the MC_NewService call

Remarks

The **MC_FreeService** function frees the system resources used by the service objects.

NOTE: If the service being freed is already in use in a service list an error is returned. The service list must be freed prior to freeing the service

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_SERVICE_IN_USE	The service is in use in a service list
MC_INVALID_SERVICE_NAME	The service name does not represent a name currently defined.

See Also

MC_NewService
MC_NewProposedServiceList
MC_FreeServiceList

MC_FreeServiceList

Releases a MergeCOM-3 service list.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_FreeServiceList(
      char* ServicelistName
)
```

ServiceListName Service list name assigned by the application in the MC_NewProposedServiceList call

Remarks

The **MC_FreeServiceList** function frees the system resources used by service list objects that were dynamically created using the MC_NewProposedServiceList function.

NOTE: Extreme care must be taken to assure that no association is currently using the service list.

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_SERVICE_LIST_NAME	The service list name does not represent a name currently defined.

See Also

MC_NewService
MC_NewProposedServiceList
MC_FreeService

MC_FreeSyntaxList

Releases a MergeCOM-3 message object.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_FreeSyntaxList(
      char* SyntaxlistName
)
```

SyntaxListName Syntax list name assigned by the application in the MC_NewSyntaxList call

Remarks

The **MC_FreeSyntaxList** function frees the system resources used by the syntax list objects.

NOTE: If the syntax list is currently in use in a service an error is returned. The service must be freed prior to freeing the syntax list.

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_SYNTAX_LIST_NAME	The syntax list name does not represent a name currently defined.
MC_SYNTAX_IN_USE	The syntax list is in use by a service definition.

See Also

MC_NewSyntaxList
MC_ServiceServiceList
MC_FreeService

MC_Get_Association_Info

Retrieves information about a given association; including the number of proposed and accepted services, the remote application title, and remote host name.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Get_Association_Info (
      int AssociationID,
      AssocInfo* AssociationInfo
)
```

AssociationID The identifier assigned to this association object by the **MC_Open_Association, MC_Open_Secure_Association, MC_Wait_For_Association** or **MC_Wait_For_Secure_Association** function.

AssociationInfo A structure of type AssocInfo you have declared that is filled with information about the association. AssocInfo is defined in mergecom.h as:

```
typedef struct MC_Assoc_Info {
       int     NumberOfProposedServices;     /* from service list */
       int     NumberOfAcceptableServices;   /* Acceptable both sides */
       char    RemoteApplicationTitle[20];   /* 16-characters max */
       char    RemoteHostName[40];           /* Network node name */
       int     Tcp_socket;                   /* the TCP socket the
                                                association is using */
       char    RemoteIPAddress[40];          /* Network IP address */
       char    LocalApplicationTitle[20];    /* 16-characters max */
       char    RemoteImplementationClassUID[66];/* 64-characters max */
       char    RemoteImplementationVersion[20]; /* 16-characters max */
       unsigned long LocalMaximumPDUSize;
       unsigned long RemoteMaximumPDUSize;
       unsigned short MaxOperationsInvoked; /* Negotiated Max operations
                                             * invoked by the assoc
                                               requestor */
       unsigned short MaxOperationsPerformed;/* Negotiated Max operations
                                                performed by the assoc
                                                requestor */
} AssocInfo;
```

Remarks

MC_Get_Association_Info retrieves information about an established association identified by *AssociationID*. This association information is returned in the structure pointed to by *AssociationInfo* detailed above. Specific information regarding the accepted services is not specified in this structure and needs to be obtained through the **MC_Get_First_Acceptable_Service** and **MC_Get_Next_Acceptable_Service** calls.

This function is to be called after an **MC_Open_Association, MC_Open_Secure_Association, MC_Wait_For_Association** or **MC_Wait_For_Secure_Association** call, and before any **MC_Read_Message**, **MC_Send_Request_Message**, or **MC_Send_Response_Message** calls.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.

MC_INVALID_ASSOC_ID	*AssociationID* is not a valid association object ID.
MC_NULL_POINTER_PARM	The *AssociationInfo* pointer had a null value.
MC_STATE_VIOLATION	The call was made after messages have been exchanged over the association.
MC_LIBRARY_NOT_INITIALIZED	The library has not been properly initialized.

See Also

MC_Get_First_Acceptable_Service
MC_Get_Next_Acceptable_Service
MC_Library_Initialization
MC_Get_Listen_Socket
MC_Get_Listen_Socket_For_Port

MC_Get_Attribute_Info

Retrieves information about a given attribute in a message.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Get_Attribute_Info (
      int MsgFileItemID,
      unsigned long Tag,
      MC_VR* ValueRep,
      int* NumValues
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
Tag	A tag identifying a message attribute.
ValueRep	The attribute's value representation code is returned here. Possible return codes (defined in "mc3msg.h") are: **AE, AS, CS, DA, DS, DT, IS, LO, LT, PN, SH, ST, TM, UI, SS, US, AT, SL, UL, FL, FD, OB, OW, OF, SQ, UNKNOWN_VR**
NumValues	The number of values assigned to the attribute is placed here. (**NOTE**: will be one (1) if the attribute's value is NULL.)

Remarks

MC_Get_Attribute_Info retrieves information about a message attribute identified by *Tag*. The attribute's Value Representation code is returned at *ValueRep*, and the number of values stored for the attribute is returned at *NumValues*. *ValueRep* and *NumValues* parameters may be NULL, in which case the corresponding information is not returned. If the attribute is "empty", *NumValues* will contain zero(0). If the attribute's value is NULL, *NumValues* will contain one(1).

MC_Get_Attribute_Info sets an internal "attribute pointer" such that **MC_Get_Next_Attribute** will retrieve the attribute following this one.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.
MC_INVALID_TAG	The message does not contain an attribute with an ID of *Tag*.

See Also

MC_Get_First_Attribute
MC_Get_Next_Attribute
MC_Get_pAttribute_Info
MC_Get_Tags_Dict_Info

MC_Get_Bool_Config_Value

Used to get the value of a boolean tool kit configuration parameter at runtime.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Get_Bool_Config_Value (
      BoolParm Parm, int* Value
)
```

Parm An enumerated constant identifying the boolean configuration parameter to get. *Parm* can have any of the following values:

LOG_FILE_BACKUP,
ACCEPT_ANY_CONTEXT_NAME,
ACCEPT_ANY_APPLICATION_TITLE,
ACCEPT_ANY_PROTOCOL_VERSION,
ACCEPT_DIFFERENT_IC_UID,
ACCEPT_DIFFERENT_VERSION,
AUTO_ECHO_SUPPORT,
SEND_SOP_CLASS_UID,
SEND_SOP_INSTANCE_UID,
SEND_LENGTH_TO_END,
SEND_RECOGNITION_CODE,
SEND_MSG_ID_RESPONSE,
SEND_ECHO_PRIORITY,
SEND_RESPONSE_PRIORITY,
INSURE_EVEN_UID_LENGTH,
BLANK_FILL_LOG_FILE,
ELIMINATE_ITEM_REFERENCES,
HARD_CLOSE_TCP_IP_CONNECTION,
FORCE_DICM_IN_PREFIX,
ACCEPT_ANY_PRESENTATION_CONTEXT,
ACCEPT_ANY_HOSTNAME,
ALLOW_OUT_OF_ORDER_TAGS,
EMPTY_PRIVATE_CREATOR_CODES,
FORCE_OPEN_EMPTY_ITEM,
FORCE_JAVA_BIG_ENDIAN,
ACCEPT_MULTIPLE_PRES_CONTEXTS,
ALLOW_INVALID_PRIVATE_CREATOR_CODES,
REMOVE_PADDING_CHARS,
PRIVATE_SYNTAX_1_LITTLE_ENDIAN,
PRIVATE_SYNTAX_1_EXPLICIT_VR,
PRIVATE_SYNTAX_1_ENCAPSULATED,
PRIVATE_SYNTAX_2_LITTLE_ENDIAN,
PRIVATE_SYNTAX_2_EXPLICIT_VR,
PRIVATE_SYNTAX_2_ENCAPSULATED,
EXPORT_UN_VR_TO_MEDIA,
EXPORT_UN_VR_TO_NETWORK,
EXPORT_PRIVATE_ATTRIBUTES_TO_MEDIA,
EXPORT_PRIVATE_ATTRIBUTES_TO_NETWORK,
ALLOW_INVALID_PRIVATE_ATTRIBUTES,
RETURN_COMMA_IN_DS_FL_FD_STRINGS,
ALLOW_COMMA_IN_DS_FL_FD_STRINGS,
DEFLATE_ALLOW_FLUSH,
COMPRESSION_ALLOW_FRAGS,

NETWORK_CAPTURE,
DUPLICATE_ENCAPSULATED_ICON,
COMPRESSION_WHEN_J2K_USE_LOSSY,
COMPRESSION_J2K_LOSSY_USE_QUALITY,
TCPIP_DISABLE_NAGLE,
COMBINE_DATA_WITH_HEADER,
REWRITE_CAPTURE_FILES,
MSG_FILE_ITEM_OBJ_TRACE,
EXPORT_UNDEFINED_LENGTH_SQ,
EXPORT_UNDEFINED_LENGTH_SQ_IN_DICOMDIR,
EXPORT_GROUP_LENGTHS_TO_NETWORK,
EXPORT_GROUP_LENGTHS_TO_MEDIA

These names are the same as those given to the parameters in the tool kit configuration files.

Value — The boolean value to which *Parm* is set is returned here.

Remarks

The MergeCOM-3 Advanced Library accesses several configuration files at startup. This call allows your application to get boolean configurable parameters specified in these files at runtime. Please see the detailed description of tool kit configuration elsewhere in this manual.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	*Value* was NULL.
MC_INVALID_PARAMETER_NAME	Parameter specified is not one of the above listed parameters.
MC_LIBRARY_NOT_INITIALIZED	The library has not been properly initialized.

See Also

MC_Get_Int_Config_Value
MC_Get_Log_Destination
MC_Get_Long_Config_Value
MC_Get_String_Config_Value
MC_Set_Bool_Config_Value
MC_Set_Int_Config_Value
MC_Set_Log_Destination
MC_Set_Long_Config_Value
MC_Set_String_Config_Value

MC_Get_Encapsulated_Value_To_Function

Retrieves the first frame of encapsulated pixel data from a message. If a decompressor is registered, then the data will be decompressed.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Get_Encapsulated_Value_To_Function (
      int MsgFileItemID,
      unsigned long Tag,
     void* UserInfo,
      void* YourGetFunction,
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
Tag	A tag identifying a message attribute containing pixel data.
UserInfo	Address of data which will be passed on to *YourGetFunction* each time it is called. This may be NULL.
YourGetFunction	Name of a function which will be called repeatedly to provide blocks of data of the attribute's value

The function must be prototyped as follows:

```
MC_STATUS YourGetFunction (
      int CBMsgFileItemID,
      unsigned long CBtag,
      void* CBuserInfo,
      int CBdataSize,
      void* CBdataBuffer,
      int CBisFirst,
      int CBisLast
)
```

CBMsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
CBtag	DICOM tag which identifies the attribute.
CBuserInfo	Address of user data which is being passed from the **MC_Get_Encapsulated_Value_To_Function** function. This may be NULL.
CBdataSize	The number of bytes in *CBdataBuffer*.
CBdataBuffer	Address of the buffer containing a portion of the attribute's value.
CBisFirst	This is TRUE (not zero) the first time ***YourGetFunction*** is being called for this attribute.
CBisLast	This is TRUE (not zero) the last time ***YourGetFunction*** is being called for this attribute.

Remarks

The **MC_Get_Encapsulated_Value_To_Function** function is used to fetch the value of the first frame of an attribute which has a value representation of OB or OW and is

encapsulated. If a decompressor is registered, the data will be decompressed. Such attributes tend to have values of great length. To accommodate this, one passes the **MC_Get_Encapsulated_Value_To_Function** function the name of a function (***YourGetFunction***) which MergeCOM-3, in turn, calls. This "callback" function is called repeatedly to provide blocks of the attribute's data value. If a decompressor is registered for this message, the data will be passed back decompressed. An optional *UserInfo* parameter may be used to pass information between the **MC_Get_Encapsulalated_Value_To_Function** caller and ***YourGetFunction*** which receives the data in its *CBuserInfo* parameter. If no such exchange of data is necessary, you may use NULL for *UserInfo*.

MC_Get_Encapsulated_Value_To_Function sets an internal frame pointer such that **MC_Get_Next_Encapsulated_Value_To_Function** will retrieve the next frame following this one.

YourGetFunction

Each block of data is returned in *CBdataBuffer* and the number of bytes in the block is specified by *CBdataSize*.

CBisFirst is set to TRUE the first time ***YourGetFunction*** is called. This provides a clear mechanism for the function to know it is being called the first time for the attribute identified by *CBtag* in the message identified by *CBMsgFileItemID*. Users often open a file at this time.

CBisLast is set to TRUE the last time ***YourGetFunction*** is called. This provides a clear mechanism for the function to know it is being called the last time for the attribute identified by *CBtag* in the message identified by *CBMsgFileItemID*. Users often close a file at this time.

YourGetFunction must return **MC_NORMAL_COMPLETION** if all went well. If not, it must return **MC_CANNOT_COMPLY**. Both of these are defined in "mc3msg.h".

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally, and there are no more frames in the message.
MC_INVALID_TRANSFER_SYNTAX	The message's transfer syntax is non-encapsulated.
MC_END_OF_FRAME	The function completed normally, and there are more frames in the message.
MC_NULL_POINTER_PARM	*YourGetFunction* is NULL
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.
MC_INVALID_TAG	The message does not contain an attribute with an ID of *Tag*.
MC_INCOMPATIBLE_VR	The attribute does not contain OB or OW data
MC_EMPTY_VALUE	No value has been assigned to this attribute yet.

MC_NULL_VALUE	The attribute has a value of NULL.
MC_NO_MORE_VALUES	The attribute has no more values.
MC_CALLBACK_CANNOT_COMPLY	*YourGetFunction* returned MC_CANNOT_COMPLY.
MC_COMPRESSION_FAILURE	There was an error while trying to decompress the data.
MC_MISSING_DELIMITER	The expected delimiter can not be detected

See Also

MC_Get_Next_Encapsulated_Value_To_Function
MC_Register_Compression_Callbacks

MC_Get_Enum_From_Transfer_Syntax

Converts the DICOM UID for a transfer syntax into an enumerated value.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Get_Enum_From_Transfer_Syntax (
      char* Uid,
      TRANSFER_SYNTAX* SyntaxType

)
```

Uid — The transfer syntax UID for conversion is specified here.

SyntaxType — The DICOM transfer syntax is returned here. One of the enumerated TRANSFER_SYNTAX types defined in "mc3msg.h" is returned:

IMPLICIT_LITTLE_ENDIAN
IMPLICIT_BIG_ENDIAN
EXPLICIT_LITTLE_ENDIAN
EXPLICIT_BIG_ENDIAN
DEFLATED_EXPLICIT_LITTLE_ENDIAN
RLE
JPEG_BASELINE
JPEG_EXTENDED_2_4
JPEG_EXTENDED_3_5
JPEG_SPEC_NON_HIER_6_8
JPEG_SPEC_NON_HIER_7_9
JPEG_FULL_PROG_NON_HIER_10_12
JPEG_FULL_PROG_NON_HIER_11_13
JPEG_LOSSLESS_NON_HIER_14
JPEG_LOSSLESS_NON_HIER_15
JPEG_EXTENDED_HIER_16_18
JPEG_EXTENDED_HIER_17_19
JPEG_SPEC_HIER_20_22
JPEG_SPEC_HIER_21_23
JPEG_FULL_PROG_HIER_24_26
JPEG_FULL_PROG_HIER_25_27
JPEG_LOSSLESS_HIER_28
JPEG_LOSSLESS_HIER_29
JPEG_LOSSLESS_HIER_14
JPEG_2000_LOSSLESS_ONLY
JPEG_2000
JPEG_LS_LOSSLESS
JPEG_LS_LOSSY
MPEG2_MPML
PRIVATE_SYNTAX_1 or
PRIVATE_SYNTAX_2

Remarks

MC_Get_Enum_From_Transfer_Syntax converts a DICOM transfer syntax UID into a MergeCOM-3 enumerated value**.** The UID values used by **MC_Get_Enum_From_Transfer_Syntax** are specified in MergeCOM-3's configuration files. Please see the detailed description of tool kit configuration elsewhere in this manual.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	One or more of the pointer-type parameters was NULL.
MC_INVALID_TRANSFER_SYNTAX	*Uid* is not registered in the MergeCOM-3 configuration files.

See Also

MC_Get_Transfer_Syntax_From_Enum
MC_Get_MergeCOM_Service
MC_Get_UID_From_MergeCOM_Service

MC_Get_File_Length

Gets the length in bytes of a file object.

Synopsis

```
#include "mc3media.h"

MC_STATUS MC_Get_File_Length (
      int FileID,
      unsigned long* FileLength
)
```

FileID	The identifier assigned to this object by the **MC_Create_File** or **MC_Create_Empty_File** function.
FileLength	The file object size is returned here.

Remarks

MC_Get_File_Length returns the number of bytes it would take to write a file object in DICOM Part 10 format. This function assumes the transfer syntax UID is contained in the group 2 elements. Note that a possible value for *FileLength* is undefined length (i.e., 0xffffffff), when the file object is encoded in an encapsulated transfer syntax. Also note that the file length may change. The padding is specified when calling **MC_Write_File**.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_FILE_ID	The *FileID* value is not a valid file object ID.
MC_INVALID_TRANSFER_SYNTAX	The transfer syntax UID contained in the file objects group 2 elements was incorrect.
MC_LIBRARY_NOT_INITIALIZED	The library has not been properly initialized.
MC_NULL_POINTER_PARM	*FileLength* was NULL.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment.

MC_Get_File_Preamble

Gets the preamble for a file object.

Synopsis

```
#include "mc3media.h"

MC_STATUS MC_Get_File_Preamble (
      int FileID,
      char* Preamble
)
```

FileID	The identifier assigned to this object by the **MC_Create_File** or **MC_Create_Empty_File** function.
Preamble	A pointer to a user allocated 128 byte buffer in which the preamble associated with the file object *FileID* is copied.

Remarks

MC_Get_File_Preamble retrieves the preamble associated with a file object. The function stores the DICOM file preamble for the file object *FileID* in the buffer pointed to by *Preamble*.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_FILE_ID	The *FileID* value is not a valid file object ID.
MC_NULL_POINTER_PARM	*Preamble* was NULL.

See Also

MC_Set_File_Preamble

MC_Get_Filename

Gets the filename associated with a file object.

Synopsis

```
#include "mc3media.h"

MC_STATUS MC_Get_Filename (
      int FileID,
      char* FileName,
      int FileSize,

)
```

FileID	The identifier assigned to this object by the **MC_Create_Empty_File** or **MC_Create_File** function.
Filename	A pointer to a string where the filename will be returned.
FileSize	The length in bytes of *Filename*.

Remarks

The **MC_Get_Filename** function retrieves the filename associated with a file object.

The filename is passed to the user's callback function when the **MC_Write_File** function is called.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_FILE_ID	The *FileID* value is not a valid file object ID.
MC_NULL_POINTER_PARM	*Filename* was NULL.
MC_BUFFER_TOO_SMALL	*Filename* was too small to contain the complete filename.

See Also

MC_Write_File **MC_Empty_File**

MC_Get_First_Acceptable_Service

Retrieves the first of possibly many services that have been accepted over an association.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Get_First_Acceptable_Service (
      int AssocID,
      ServiceInfo* AserviceInfo
)
```

AssocID The identifier assigned to this object by the **MC_Open_Association, MC_Open_Secure_Association, MC_Wait_For_Association** or **MC_Wait_For_Secure_Association** function.

AserviceInfo A structure of type ServiceInfo you have declared that is filled with information about the accepted service. ServiceInfo is defined in mergecom.h as:

```
typedef struct MC_Service_Info {
       char    ServiceName[50];      /* MergeCOM-3 service name */
       TRANSFER_SYNTAX SyntaxType;   /* transfer syntax
                                        negotiated */
       ROLE_TYPE RoleNegotiated;     /* The role negotiated for the
                                        service */
       int    PresentationContextID;
} ServiceInfo;
```

"TRANSFER_SYNTAX" and "ROLE_TYPE" are defined in mc3msg.h.

Remarks

The MergeCOM service name and transfer syntax negotiated is returned for the first service negotiated for and acceptable to both sides. These values are returned in the structure detailed above.

MC_Get_First_Acceptable_Service sets an internal "service pointer" such that **MC_Get_Next_Acceptable_Service** will retrieve the service following this one.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_ASSOC_ID	*AssocID* is not a valid association object ID.
MC_NULL_POINTER_PARM	One or more of the pointer-type parameters was NULL.
MC_STATE_VIOLATION	Call was made after reading or sending a message.
MC_LIBRARY_NOT_INITIALIZED	The library has not been properly initialized.
MC_END_OF_LIST	No services were accepted.

See Also

MC_Open_Association
MC_Open_Secure_Association
MC_Wait_For_Association
MC_Wait_For_Secure_Association
MC_Get_Next_Acceptable_Service

MC_Get_First_Attribute

Retrieves information about the first attribute in a message.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Get_First_Attribute (
      int MsgFileItemID,
      unsigned long* Tag,
      MC_VR* ValueRep,
      int* NumValues
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
Tag	The attribute's tag is returned here.
ValueRep	The attribute's value representation code is returned here. Possible return codes (defined in "mc3msg.h") are: **AE, AS, CS, DA, DS, DT, IS, LO, LT, PN, SH, ST, TM, UI, SS, US, AT, SL, UL, FL, FD, OB, OW, OF, SQ, UNKNOWN_VR**
NumValues	The number of values assigned to the attribute is returned here. (**NOTE**: will return one (1) if the attribute's value is NULL.)

Remarks

MC_Get_First_Attribute retrieves information about the first attribute in a message. The attribute's tag is returned at *Tag*, the attribute's Value Representation code is returned at *ValueRep* and the number of values stored for the attribute is returned at *NumValues*. Any of the *Tag*, *ValueRep* or *NumValues* parameters may be NULL, in which case the corresponding information is not returned. If the attribute is "empty" *NumValues* will contain zero(0). If the attribute's value is NULL, *NumValues* will contain 1.

MC_Get_First_Attribute sets an internal "attribute pointer" such that **MC_Get_Next_Attribute** will retrieve the attribute following this one.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.
MC_NULL_POINTER_PARM	One or more of the pointer-type parameters was NULL.
MC_MESSAGE_EMPTY	There are no attributes in the message.

See Also

MC_Get_Next_Attribute
MC_Get_Attribute_Info
MC_Get_pAttribute_Info

MC_Get_Int_Config_Value

Used to get the value of an integer tool kit configuration parameter at runtime.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Get_Int_Config_Value (
      IntParm Parm,
      int* Value
)
```

Parm — An enumerated constant identifying the integer configuration parameter to get. *Parm* can have any of the following values:

LOG_FILE_SIZE,
LOG_MEMORY_SIZE,
ARTIM_TIMEOUT,
ASSOC_REPLY_TIMEOUT,
RELEASE_TIMEOUT,
WRITE_TIMEOUT,
CONNECT_TIMEOUT,
INACTIVITY_TIMEOUT,
LARGE_DATA_SIZE,
DESIRED_LAST_PDU_SIZE,
OBOW_BUFFER_SIZE,
FLATE_GROW_OUTPUT_BUF_SIZE,
DEFLATE_COMPRESSION_LEVEL,
COMPRESSION_LUM_FACTOR,
COMPRESSION_CHROM_FACTOR,
COMPRESSION_J2K_LOSSY_RATIO,
COMPRESSION_J2K_LOSSY_QUALITY,
WORK_BUFFER_SIZE,
TCPIP_LISTEN_PORT,
MAX_PENDING_CONNECTIONS,
TCPIP_SEND_BUFFER_SIZE,
TCPIP_RECEIVE_BUFFER_SIZE,
NUM_HISTORICAL_LOG_FILES,
LOG_FILE_LINE_LENGTH,
NUMBER_OF_CAP_FILES,
IGNORE_JPEG_BAD_SUFFIX

These names are the same as those given to the parameters in the tool kit configuration files. A description of the options can be found in Appendix B.

Value — *Parm*'s integer value is returned here.

Remarks

The MergeCOM-3 Advanced Tool Kit Library accesses several configuration files at startup. This call allows your application to get integer configurable parameters specified in these files at runtime. Please see the detailed description of tool kit configuration elsewhere in this manual.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	*Value* was NULL.
MC_INVALID_PARAMETER_NAME	Parameter specified is not one of the above listed parameters.
MC_MUST_BE_POSITIVE	Parameter value cannot be negative.
MC_LIBRARY_NOT_INITIALIZED	The library has not been properly initialized.

See Also

MC_Set_Int_Config_Value
MC_Get_Log_Destination **MC_Set_Log_Destination**
MC_Get_Long_Config_Value **MC_Set_Long_Config_Value**
MC_Get_String_Config_Value **MC_Set_String_Config_Value**

MC_Get_Listen_Socket

Returns the id of the socket being listened on

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Get_Listen_Socket (
      int* AlistenSocket
)
```

AlistenSocket The socket being listened on.

Remarks

MC_Get_Listen_Socket returns the DICOM listen socket. It will return the current listen port used by calls to **MC_Wait_For_Association** or **MC_Wait_For_Secure_Association. MC_Get_Listen_Socket_For_Port** must be used when listening is done with **MC_Wait_For_Association_On_Port** or **MC_Wait_For_Secure_Association_On_Port.**

In specialized cases where the server application is waiting on several asynchronous events, not just the association event, the **MC_Get_Listen_Socket** call can be made to request the file descriptor for the DICOM listen socket. In this way the server application can do a **`select( )`** system call on this and other file descriptors. When the **`select`** returns with an event on the DICOM listen socket descriptor, the application can call **MC_Wait_For_Association** or **MC_Wait_For_Secure_Association** and get an immediate response.

Similarly, once the association is established, both the client and server applications can use the **MC_Get_Association_Info** call to get the file descriptor for the socket over which message exchange will occur. Again, **`select( )`** can be used to wait asynchronously for a DICOM request or response message.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	*AlistenSocket* was NULL.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment.

See Also

MC_Get_Association_Info

MC_Get_Listen_Socket_For_Port

Returns the id of the socket being listened on for a specific port

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Get_Listen_Socket_For_Port (
      int Port
      int* ListenSocket
)
```

Port	The port number to check for the listen socket
ListenSocket	The socket being listened on.

Remarks

MC_Get_Listen_Socket_For_Port returns the DICOM listen socket for a specific port. It can be used to get the default listen port from a call to **MC_Wait_For_Association** or **MC_Wait_For_Secure_Association**. It can also be used to get the listen socket for a specific listen port from calls to **MC_Wait_For_Association_On_Port** and **MC_Wait_For_Secure_Association_On_Port**.

In specialized cases where the server application is waiting on several asynchronous events, not just the association event, the **MC_Get_Listen_Socket_For_Port** call can be made to request the file descriptor for the DICOM listen socket. In this way the server application can do a **`select( )`** system call on this and other file descriptors. When the **`select`** returns with an event on the DICOM listen socket descriptor, the application can call **MC_Wait_For_Association** or **MC_Wait_For_Secure_Association** and get an immediate response.

Similarly, once the association is established, both the client and server applications can use the **MC_Get_Association_Info** call to get the file descriptor for the socket over which message exchange will occur. Again, **`select( )`** can be used to wait asynchronously for a DICOM request or response message.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	*LlistenSocket* was NULL.
MC_INVALID_PORT_NUMBER	*Port* is not currently being listened on .
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment.

See Also

MC_Get_Association_Info

MC_Get_Log_Destination

Used to get the value of a integer tool kit configuration parameter at runtime.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Get_Log_Destination (
      LogParm Parm,
      int* Value
)
```

Parm — An enumerated constant identifying the class of logging that is to be retreived. *Parm* can have any of the following values:

ERROR_DESTINATIONS,
WARNING_DESTINATIONS,
INFO_DESTINATIONS,
T1_DESTINATIONS,
T2_DESTINATIONS,
T3_DESTINATIONS,
T4_DESTINATIONS,
T5_DESTINATIONS,
T6_DESTINATIONS,
T7_DESTINATIONS,
T8_DESTINATIONS
T9_DESTINATIONS

Value — A defined term specifying where the logging is being directed. *Value* can have any of the following values:

File_Destination,
Memory_Destination,
Screen_Destination,
Bitbucket_Destination

These values can also be OR'ed together to indicate multiple destinations.

Remarks

This call allows you to retrieve the logging of error, warning, and info messages at runtime. The [DEFAULT_LIBRARY] section of the MergeCOM-3 initialization file contains the setting used at startup. Please see the detailed description of tool kit configuration elsewhere in this manual.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	*Value* was NULL.
MC_INVALID_PARAMETER_NAME	Parameter specified is not one of the above listed parameters.
MC_LIBRARY_NOT_INITIALIZED	The library has not been properly initialized.

See Also

MC_Get_Int_Config_Value
MC_Set_Int_Config_Value
MC_Set_Log_Destination
MC_Get_Long_Config_Value
MC_Set_Long_Config_Value
MC_Get_String_Config_Value
MC_Set_String_Config_Value

MC_Get_Long_Config_Value

Used to get the value of a long integer tool kit configuration parameter at runtime.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Get_Long_Config_Value (
      LongParm Parm,
      long int* Value
)
```

Parm — An enumerated constant identifying the long integer configuration parameter to get. *Parm* can have only the following values:

PDU_MAXIMUM_LENGTH,
CALLBACK_MIN_DATA_SIZE,
CAPTURE_FILE_SIZE

This name is the same as those given to the parameters in the tool kit configuration files.

Value — *Parm*'s long integer value is returned here.

Remarks

The MergeCOM-3 Advanced Tool Kit Library accesses several configuration files at startup. This call allows your application to get long integer configurable parameters specified in these files at runtime. Please see the detailed description of tool kit configuration elsewhere in this manual.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	*Value* was NULL.
MC_INVALID_PARAMETER_NAME	Parameter specified is not one of the above listed parameters.
MC_LIBRARY_NOT_INITIALIZED	The library has not been properly initialized.

See Also

MC_Get_Int_Config_Value
MC_Set_Int_Config_Value
MC_Get_Log_Destination
MC_Set_Log_Destination
MC_Set_Long_Config_Value
MC_Get_String_Config_Value
MC_Set_String_Config_Value

MC_Get_MergeCOM_Service

Returns the MergeCOM-3 service name associated with a DICOM SOP Class UID.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Get_MergeCOM_Service (
      char* UID,
      char* ServiceName,
      int BufferSize
)
```

UID	The SOP class UID for which to find the associated MergeCOM-3 service name.
ServiceName	The name of the MergeCOM-3 service associated with *UID* is returned here.
BufferSize	The length of *ServiceName* in bytes.

Remarks

MC_Get_MergeCOM_Service returns the service name for *UID*. This function is often useful in conjunction with **MC_Set_Service_Command** to set the message type for a message before validation or sending across the network.

Reference the "DICOM V3.0 Standard, Final Text - October 29, 1993" for more information about these SOP Class UIDs.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	*UID* or *ServiceName* was NULL.
MC_BUFFER_TOO_SMALL	*ServiceName* was not large enough to contain the MergeCOM-3 service name.
MC_INVALID_SOP_CLASS_UID	*UID* does not contain a MergeCOM-3 supported SOP Class UID.

See Also

MC_Set_Service_Command
MC_Get_Tag_Info
MC_Get_pTag_Info
MC_Get_Enum_From_Transfer_Syntax
MC_Get_Transfer_Syntax_From_Enum
MC_Get_UID_From_MergeCOM_Service

MC_Get_Message_Service

Returns the service name and command ID association with a message.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Get_Message_Service (
      int MessageID,
      char** ServiceName,
      MC_COMMAND* Command
)
```

MessageID	The message object's identification number.
ServiceName	The name of the service associated with the message is returned here.
Command	The command associated with this message is returned here. The **MC_COMMAND** enumerated values are defined in "mergecom.h".

Remarks

MC_Get_Message_Service returns the service name and command which were specified when the message identified by *MessageID* was opened. This function is often useful in callback functions which are supporting more than one service or command. Note well that this memory is de-allocated when the **MC_Free_Message** API call is made for *MessageID*.

The following commands are supported by the Advanced MergeCOM-3 Tool Kit:

Command	Description
C_STORE_RQ	DICOM Composite Store Service Request
C_STORE_RSP	DICOM Composite Store Service Response
C_ECHO_RQ	DICOM Verification Service Request
C_ECHO_RSP	DICOM Verification Service Response
C_FIND_RQ	DICOM Composite Find Service Request
C_FIND_RSP	DICOM Composite Find Service Response
C_CANCEL_FIND_RQ	Cancel DICOM Composite Find Service Request
C_GET_RQ	DICOM Composite Get Service Request
C_GET_RSP	DICOM Composite Get Service Response
C_CANCEL_GET_RQ	Cancel DICOM Composite Get Service Request
C_MOVE_RQ	DICOM Composite Move Service Request
C_MOVE_RSP	DICOM Composite Move Service Response
C_CANCEL_MOVE_RQ	Cancel DICOM Composite Move Service Request
N_EVENT_REPORT_RQ	DICOM Normalized Report Service Request
N_EVENT_REPORT_RSP	DICOM Normalized Report Service Response
N_GET_RQ	DICOM Normalized Get Service Request
N_GET_RSP	DICOM Normalized Get Service Request

N_SET_RQ	DICOM Normalized Set Service Request
N_SET_RSP	DICOM Normalized Set Service Response
N_ACTION_RQ	DICOM Normalized Action Service Request
N_ACTION_RSP	DICOM Normalized Action Service Response
N_CREATE_RQ	DICOM Normalized Create Service Request
N_CREATE_RSP	DICOM Normalized Create Service Response
N_DELETE_RQ	DICOM Normalized Delete Service Request
N_DELETE_RSP	DICOM Normalized Delete Service Response

Reference the “DICOM V3.0 Standard, Final Text - October 29, 1993” for more information about these commands.

Return Value

One of these enumerated **MC_STATUS** codes defined in **“mcstatus.h”**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	*ServiceName* or *Command* was NULL.
MC_INVALID_MESSAGE_ID	*MessageID* is not a valid message object ID.

See Also

MC_Register_Callback_Function

MC_Get_Message_Transfer_Syntax

Returns the transfer syntax over which a message was received over the network.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Get_Message_Transfer_Syntax (
      int MessageID,
      TRANSFER_SYNTAX* TransferSyntax
)
```

MessageID	The message object's identification number.
TransferSyntax	The name of the transfer syntax associated with the message is returned here.

Remarks

MC_Get_Message_Transfer_Syntax returns the DICOM transfer syntax over which a message was received on the network. This function is often useful when encapsulated (compressed) and standard transfer syntaxes are both negotiated for a service. The transfer syntax is needed in this case to determine how the pixel data in a message should be decoded

Reference the *MergeCOM-3 Advanced Tool Kit User's Manual* for a discussion on negotiating multiple transfer syntaxes for a service.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	*TransferSyntax* was NULL.
MC_INVALID_MESSAGE_ID	*MessageID* is not a valid message object ID.
MC_STATE_VIOLATION	The call was made on a message that has not been received over the network.

See Also

MC_Set_Message_Transfer_Syntax **MC_Read_Message**

MC_Get_Meta_ServiceName

Returns the meta service name corresponding to the presentation context in which a message was received over the network.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Get_Meta_ServiceName (
      int AssociationID,
      char* Value,
      int BufferSize
)
```

AssociationID	An association identification number returned by a **MC_Wait_For_Association** or **MC_Wait_For_Secure_Association** call.
Value	The name of the meta service of the most recent request message is returned here.
BufferSize	The size of the Value buffer in bytes

Remarks

MC_Get_Meta_ServiceName returns the DICOM meta service name for a request message that was received on the network. This function is only useful on the receiving end of a **MC_Send_Request_Message_For_Service**. An example is the Filmbox when both BASIC_GRAYSCALE_PRINT_MANAGEMENT and BASIC_COLOR_PRINT_MANAGEMENT are negotiated. The **MC_Send_Request_Message_For_Service** would specify which of the services should be used. On the receiving end, **MC_Read_Message** returns BASIC_FILM_BOX as the service name, but does not contain any information on the meta SOP. This function will return the meta SOP based on the most recent received message's presentation context.

This function should only be used immediately after receiving a request message.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_ASSOC_ID	The association ID passed in was not valid.
MC_NULL_POINTER_PARM	*Value* was NULL.
MC_BUFFER_TOO_SMALL	*Value* is not large enough.
MC_NOT_META_SOP	The most recent received message was not sent on a presentation context that belongs to a Meta SOP.
MC_NOT_FOUND	The meta SOP for the most recently received message could not be found.

See Also

MC_Send_Request_Message_For_Service **MC_Read_Message**

MC_Get_Negotiation_Info

Retrieves received extended negotiation information.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Get_Negotiation_Info (
      int AssociationID,
      char* ServiceName,
      void** ExtInfoBuffer,
      int* ExtInfoLength
)
```

AssociationID	An association identification number returned by a **MC_Wait_For_Association** or **MC_Wait_For_Secure_Association** call.
ServiceName	The name given to a valid DICOM service.
ExtInfoBuffer	A pointer to the buffer containing extended negotiation information will be returned here.
ExtInfoLength	The number of bytes contained in the *ExtInfoBuffer* will be returned here.

Remarks

The DICOM standard allows application entities to exchange "extended negotiation information" when establishing an association. The contents of the negotiation information must be known to both the association requestor application and the association acceptor application. Such extended negotiation is not often used for DICOM services, but some services may require it.

MC_Get_Negotiation_Info allows the caller to retrieve any extended negotiation information which may have been received during a **MC_Wait_For_Association** or **MC_Wait_For_Secure_Association** call. If the remote application sent negotiation information for *ServiceName*, the negotiation information is returned at *ExtInfoBuffer* and its length at *ExtInfoLength*. If no negotiation information was received, the function will return a status of **MC_EXT_INFO_UNAVAILABLE**.

MC_Set_Negotiation_Info_For_Association may be used to set the extended negotiation information before calling **MC_Accept_Association** to accept the association. MergeCOM-3 will only return extended negotiation information to the remote application that has been registered with **MC_Set_Negotiation_Info_For_Association**.

Note also that **MC_Set_Negotiation_Info** may be used to set extended negotiation for all associations accepted with **MC_Accept_Association**. Any service with extended negotiation information in the association request will have the extended negotiation information registered in **MC_Set_Negotiation_Info** returned by **MC_Accept_Association**. Use of **MC_Set_Negotiation_Info** is depracated.

Use **MC_Clear_Negotiation_Info** to remove extended information registered using the **MC_Set_Negotiation_Info** function call.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning

MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_ASSOC_ID	*AssociationID* is not a valid association identifier.
MC_NULL_POINTER_PARM	Either *ServiceName* or *ExtInfoBuffer* or *ExtInfoLength* was NULL.
MC_UNKNOWN_SERVICE	*ServiceName* was not registered in the MergeCOM-3 configuration files.
MC_EXT_INFO_UNAVAILABLE	No extended negotiation information was received for *ServiceName*.

See Also

MC_Set_Negotiation_Info **MC_Clear_Negotiation_Info**
MC_Set_Negotiation_Info_For_Association

MC_Get_Next_Acceptable_Service

Retrieves additional services that have been accepted over an association, after **MC_Get_First_Acceptable_Service** has been called to retrieve the first.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Get_Next_Acceptable_Service (
      int AssocID,
      ServiceInfo* AserviceInfo
)
```

AssocID The identifier assigned to this object by the **MC_Open_Association, MC_Open_Secure_Association, MC_Wait_For_Association** or **MC_Wait_For_Secure_Association** function.

AserviceInfo A structure of type ServiceInfo you have declared that is filled with information about the accepted service. ServiceInfo is defined in mergecom.h as:

```
typedef struct MC_Service_Info {
       char    ServiceName[50];      /* MergeCOM-3 service name */
       TRANSFER_SYNTAX SyntaxType;   /* transfer syntax
                                        negotiated */
       ROLE_TYPE RoleNegotiated;     /* The role negotiated for the
                                        service */
       int    PresentationContextID;
} ServiceInfo;
```

"TRANSFER_SYNTAX" and "ROLE_TYPE" are defined in mc3msg.h.

Remarks

The MergeCOM service name and transfer syntax negotiated is returned for the first service negotiated for and acceptable to both sides. These values are returned in the structure detailed above.

When there are no more services in the list of accepted services **MC_END_OF_LIST** is returned.

MC_Get_Next_Acceptable_Service sets an internal "service pointer" such that **MC_Get_Next_Acceptable_Service** can be called again to retrieve the service following the current one.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_ASSOC_ID	*AssocID* is not a valid association object ID.
MC_NULL_POINTER_PARM	One or more of the pointer-type parameters was NULL.
MC_STATE_VIOLATION	Call was made after reading or sending a message.
MC_LIBRARY_NOT_INITIALIZED	The library has not been properly initialized.
MC_END_OF_LIST	No more services in list of accepted services.

See Also

MC_Open_Association
MC_Open_Secure_Association
MC_Wait_For_Association
MC_Wait_For_Secure_Association
MC_Get_First_Acceptable_Service

MC_Get_Next_Attribute

Retrieves information about the next attribute in a message.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Get_Next_Attribute (
      int MsgFileItemID,
      unsigned long* Tag,
      MC_VR* ValueRep,
      int* NumValues
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
Tag	The attribute's tag is returned here.
ValueRep	The attribute's value representation code is returned here. Possible return codes (defined in "mc3msg.h") are: **AE, AS, CS, DA, DS, DT, IS, LO, LT, PN, SH, ST, TM, UI, SS, US, AT, SL, UL, FL, FD, OB, OW, OF, SQ, UNKNOWN_VR**
NumValues	The number of values assigned to the attribute is returned here. (**NOTE**: will be one (1) if the attribute's value is NULL.)

Remarks

MC_Get_Next_Attribute retrieves information about the next attribute in a message. **MC_Get_First_Attribute**, **MC_Get_Attribute_Info** or **MC_Get_pAttribute_Info** must be used first to set a current attribute location. The attribute's tag is returned at *Tag*, the attribute's Value Representation code is returned at *ValueRep*, and the number of values stored for the attribute is returned at *NumValues*. Any of the *Tag*, *ValueRep* or *NumValues* parameters may be NULL, in which case the corresponding information is not returned. If the attribute is "empty", *NumValues* will contain zero(0). If the attribute's value is NULL, *NumValues* will contain one(1).

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.
MC_MESSAGE_EMPTY	The message contains no attributes.
MC_NO_MORE_ATTRIBUTES	There are no more attributes in the message.

See Also

MC_Get_First_Attribute
MC_Get_Attribute_Info
MC_Get_pAttribute_Info

MC_Get_Next_Encapsulated_Value_To_Function

Retrieves the next frame of encapsulated pixel data from a message. If a decompressor is registered, then the data will be decompressed.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Get_Next_Encapsulated_Value_To_Function (
      int MsgFileItemID,
      unsigned long Tag,
     void* UserInfo,
      void* YourGetFunction,
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
Tag	A tag identifying a message attribute containing pixel data.
UserInfo	Address of data which will be passed on to *YourGetFunction* each time it is called. This may be NULL.
YourGetFunction	Name of a function which will be called repeatedly to provide blocks of data of the attribute's value

The function must be prototyped as follows:

```
MC_STATUS YourGetFunction (
      int CBMsgFileItemID,
      unsigned long CBtag,
      void* CBuserInfo,
      int CBdataSize,
      void* CBdataBuffer,
      int CBisFirst,
      int CBisLast
)
```

CBMsgFileItemID	The message identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message** functions.
CBtag	DICOM tag which identifies the attribute.
CBuserInfo	Address of user data which is being passed from the **MC_Get_Next_Encapsulated_Value_To_Function** function. This may be NULL.
CBdataSize	The number of bytes in *CBdataBuffer*.
CBdataBuffer	Address of the buffer containing a portion of the attribute's value.
CBisFirst	This is TRUE (not zero) the first time ***YourGetFunction*** is being called for this attribute.
CBisLast	This is TRUE (not zero) the last time ***YourGetFunction*** is being called for this attribute.

Remarks

The **MC_Get_Next_Encapsulated_Value_To_Function** function is used to fetch the value of the next frame of an attribute which has a value representation of OB or OW and is encapsulated. If a decompressor is registered, the data will be decompressed. Such attributes tend to have values of great length. To accommodate this, one passes the **MC_Get_Next_Encapsulated_Value_To_Function** function the name of a function

(***YourGetFunction***) which MergeCOM-3, in turn, calls. This "callback" function is called repeatedly to provide blocks of the attribute's data value. If a decompressor is registered for this message, the data will be passed back decompressed. An optional *UserInfo* parameter may be used to pass information between the **MC_Get_Next_Encapsulalated_Value_To_Function** caller and ***YourGetFunction*** which receives the data in its *CBuserInfo* parameter. If no such exchange of data is necessary, you may use NULL for *UserInfo*.

MC_Get_Next_Encapsulated_Value_To_Function sets an internal frame pointer such that subsequent calls will retrieve the next frame following this one.

YourGetFunction

Each block of data is returned in *CBdataBuffer* and the number of bytes in the block is specified by *CBdataSize*.

CBisFirst is set to TRUE the first time ***YourGetFunction*** is called. This provides a clear mechanism for the function to know it is being called the first time for the attribute identified by *CBtag* in the message identified by *CBMsgFileItemID*. Users often open a file at this time.

CBisLast is set to TRUE the last time ***YourGetFunction*** is called. This provides a clear mechanism for the function to know it is being called the last time for the attribute identified by *CBtag* in the message identified by *CBMsgFileItemID*. Users often close a file at this time.

YourGetFunction must return **MC_NORMAL_COMPLETION** if all went well. If not, it must return **MC_CANNOT_COMPLY**. Both of these are defined in "mc3msg.h".

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally, and this was the last frame in the message.
MC_INVALID_TRANSFER_SYNTAX	The message's transfer syntax is non-encapsulated.
MC_END_OF_FRAME	The function completed normally, and there are more frames in the message.
MC_NULL_POINTER_PARM	*YourGetFunction* is NULL
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.
MC_INVALID_TAG	The message does not contain an attribute with an ID of *Tag*.
MC_INCOMPATIBLE_VR	The attribute does not contain OB or OW data
MC_EMPTY_VALUE	No value has been assigned to this attribute yet.
MC_NULL_VALUE	The attribute has a value of NULL.
MC_NO_MORE_VALUES	The attribute has no more values.
MC_CALLBACK_CANNOT_COMPLY	*YourGetFunction* returned MC_CANNOT_COMPLY.
MC_COMPRESSION_FAILURE	There was an error while trying to decompress the data.

See Also

MC_Get_Encapsulated_Value_To_Function
MC_Register_Compression_Callbacks

MC_Get_Next_pValue... Functions

Retrieves the next value of a private attribute in a message object

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Get_Next_pValue (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      MC_DT DataType,
      int BufferSize,
      void* Value,
      int* ValueSize
)
MC_STATUS MC_Get_Next_pValue_To_Double (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      double* Value
)
MC_STATUS MC_Get_Next_pValue_To_Float (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      float* Value
)
MC_STATUS MC_Get_Next_pValue_To_Int (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      int* Value
)
MC_STATUS MC_Get_Next_pValue_To_ShortInt (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      short int* Value
)
MC_STATUS MC_Get_Next_pValue_To_LongInt (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      long int* Value
)
MC_STATUS MC_Get_Next_pValue_To_String (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      int BufferSize,
```

```
      char* Value
)
MC_STATUS MC_Get_Next_pValue_To_UInt (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      unsigned int* Value
)
MC_STATUS MC_Get_Next_pValue_To_UShortInt (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      unsigned short* Value
)
MC_STATUS MC_Get_Next_pValue_To_ULongInt (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      unsigned long* Value
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
PrivateCode	The code string which identifies which block in the private *Group* "owns" the attribute.
Group	The number identifying the private group containing the private attribute. It must be an odd number.
ElementByte	The number identifying the private attribute within the private *Group* for this *PrivateCode*.
DataType	One of the enumerated codes identifying the data type of the value in *Value*. The MC_DT enumerated type is defined in "mc3msg.h". They are:

String_Type	Null-terminated character string
Int_Type	Binary integer number
UInt_Type	Binary unsigned integer number
ShortInt_Type	Binary short integer number
UShortInt_Type	Binary unsigned short integer number
LongInt_Type	Binary long integer number
ULongInt_Type	Binary unsigned long integer number
Float_Type	Binary Floating point number

Value	The attribute's value will be returned here.
ValueSize	The size of the returned value is returned here.
BufferSize	The size of the *Value* buffer in bytes.

Remarks

These functions fetch the next value of a multi-valued private attribute. If no value has yet been fetched from this attribute, these function behaves identically to the functions of similar names, with "_Next" removed from the name. If one or more values have already been fetched from this attribute, the next value with be returned.

If **MC_Get_Next_pValue** is used, the data type of the fetched value is specified by the *DataType* parameter. The other function names imply the data type requested. For example **MC_Get_Next_pValue_To_Int** is the same as calling **MC_Get_Next_pValue** with *DataType* specified as **Int_Type**. Each function will return the value requested in *Value*, which must be prototyped as the appropriate type. In the case of **MC_Get_Next_pValue_To_String**, an additional parameter (*BufferSize*) is required to specify the size of the buffer to receive the string value. For **MC_Get_Next_pValue,** *BufferSize* must be large enough to contain the data type requested.

Any reasonable conversion will be made from the attribute's value representation to the data type requested. However, some conversions are illogical (see the table below). An appropriate error code will be returned if the conversion attempt failed.

NOTE: Attributes with a value representation of **SQ** (sequence of items) have values stored internally as integers - each value is the *ItemID* of an item object opened previously using **MC_Open_Item**.

	Function may be used to retrieve values from attributes with these Value Representations
MC_Get_Next_pValue_To_Int	**DS, FD, FL, IS, SL, SS, UL, US, SQ**
MC_Get_Next_pValue_To_UInt	**DS, FD, FL, IS, SL, SS, UL, US, SQ**
MC_Get_Next_pValue_To_ShortInt	**DS, FD, FL, IS, SL, SS, UL, US, SQ**
MC_Get_Next_pValue_To_UShortInt	**DS, FD, FL, IS, SL, SS, UL, US, SQ**
MC_Get_Next_pValue_To_LongInt	**AT, DS, FD, FL, IS, SL, SS, UL, US, SQ**
MC_Get_Next_pValue_To_ULongInt	**AT, DS, FD, FL, IS, SL, SS, UL, US, SQ**
MC_Get_Next_pValue_To_Float	**DS, FD, FL, IS, SL, SS, UL, US, SQ**
MC_Get_Next_pValue_To_Double	**DS, FD, FL, IS, SL, SS, UL, US, SQ**
MC_Get_Next_pValue_To_String	**AS, AT, CS, DA, DS, DT, FD, FL, IS, LO, LT, PN, SH, SL, SS, ST, TM, UI, UL, US, SQ**

The same rules apply to the **MC_Get_Next_pValue** function, based on the value used in the *DataType* parameter.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_EMPTY_VALUE	The attribute has no value assigned to it.
MC_NULL_VALUE	The attribute's value is NULL (i.e. it's length is zero).
MC_NO_MORE_VALUES	There are no more values for this attribute.

MC_NULL_POINTER_PARM	One or more of the pointer-type parameters was NULL.
MC_INVALID_TAG	The message does not contain an attribute identified by *ElementByte* in *Group* for *PrivateCode*.
MC_INVALID_PRIVATE_CODE	The message does not contain an attributes in *Group* for *PrivateCode.*
MC_INVALID_GROUP	*Group* was not an odd number.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.
MC_BUFFER_TOO_SMALL	Buffer is not large enough to contain the value. (**MC_Get_Next_pValue** and **MC_Get_Next_pValue_To_String** only)
MC_INCOMPATIBLE_VR	The function was called to retrieve an attribute value whose value representation(VR) is not one of those listed for the function in the table above.
MC_INCOMPATIBLE_VALUE	The attribute's value is not consistent with its VR. (This should never happen.)
MC_VALUE_OUT_OF_RANGE	The attribute numeric value is too large to be accommodated by the receiving data type. (E.g. Using **MC_Get_pValue_To_Int** when the value is 123.45; or instances where the value's sign(+|–) would be affected)
MC_INVALID_DATA_TYPE	The *DataType* parameter is invalid.

See Also

MC_Get_pValue... Functions
MC_Get_Value... Functions
MC_Get_Next_Value... Functions

MC_Get_Next_Validate_Error

Used to retrieve additional validation error blocks created by the **MC_Validate_Message** function.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Get_Next_Validate_Error (
      int MessageID,
      VAL_ERR** ErrorInfo
)
```

MessageID The message object's identification number

ErrorInfo The address of a validation error block describing the violation encountered is returned here if a status of **MC_NORMAL_COMPLETION** is returned. The block is the **VAL_ERR** type defined in "mc3msg.h".

```
typedef struct ValErr_struct
{
       unsigned long Tag;     /* Tag of attribute with validation
                                 violation */
       int MsgItemID;         /* ID of message or item object containing
                                 the attribute */
       int ValueNumber;       /* Value number involved - zero if no value
                                 involved */
       MC_STATUS Status;      /* Validation violation  status code */

} VAL_ERR;
```

MC_Get_Next_Validate_Error returns additional validation error blocks created by a previous call to **MC_Validate_Message**. The next validation violation error block is returned at **ErrorInfo* if another violation exists. If not, this function returns **MC_END_OF_LIST**.

The VAL_ERR data structure is de-allocated when MC_Free_File, MC_Empty_File or MC_Empty_Message are called.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	Another validation violation exists and its information is returned at **ErrorInfo*.
MC_INVALID_MESSAGE_ID	The *MessageID* value is not a valid message object ID.
MC_NULL_POINTER_PARM	*ErrorInfo* was NULL.
MC_END_OF_LIST	The error list has been exhausted, or no **MC_Validate_Message** call has been made for this message.

Validation Violations

If a status of **MC_NORMAL_COMPLETION** is returned, the status of the next validation violation is returned at `ErrorInfo->Status`. The violation status codes are defined in **"mcstatus.h"**. They are arranged by violation type below:

INFO MESSAGES:

MC_NO_CONDITION_FUNCTION	The attribute's DICOM type is "1C" or "2C" and no function was available to check the condition. A user-defined condition function was not correctly linked with the program.
MC_UNABLE_TO_CHECK_CONDITION	Not enough information is available to check whether or not a DICOM type "1C" or "2C" attribute is required.

WARNINGS:

MC_NOT_ONE_OF_DEFINED_TERMS	A value set for this attribute is not one of the valid defined terms assigned to this attribute.
MC_NON_SERVICE_ATTRIBUTE	The attribute is not one of those defined for the message's service. Note that private attributes will not cause this violation

ERRORS:

MC_INVALID_VR_CODE	The attribute's value representation is unknown.
MC_REQUIRED_ATTRIBUTE_MISSING	An attribute which is required for the message service has been deleted from the message object.
MC_REQUIRED_VALUE_MISSING	The attribute is required to have a value and does not.
MC_VALUE_MAY_NOT_BE_NULL	The attribute is DICOM type "1" or type "1C" and it has been encoded with a NULL value.
MC_VALUE_NOT_ALLOWED	This DICOM type "1C" or type "2C" attribute may not have a value under current conditions.
MC_TOO_FEW_VALUES	The attribute is required to have more values set.
MC_TOO_MANY_VALUES	The attribute has more values set than are allowed.
MC_INVALID_ITEM_ID	This sequence of items (SQ) attribute has an invalid value assigned to it.
MC_NOT_ONE_OF_ENUMERATED_VALUES	A value set for this attribute is not one of the valid enumerated values assigned to this attribute.
MC_INVALID_VALUE_FOR_VR	A value for this attribute does not conform to the requirements of its value representation.
MC_INVALID_CHARS_IN_VALUE	A value for this attribute does not contain valid characters for its value representation.

See Also

MC_Validate_Message

MC_Get_Next_Value... Functions

Retrieves the next value of an attribute in a message object

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Get_Next_Value (
      int MsgFileItemID,
      unsigned long Tag,
      MC_DT DataType,
      int BufferSize,
      void* Value,
      int* ValueSize
)
MC_STATUS MC_Get_Next_Value_To_Double (
      int MsgFileItemID,
      unsigned long Tag,
      double* Value
)
MC_STATUS MC_Get_Next_Value_To_Float (
      int MsgFileItemID,
      unsigned long Tag,
      float* Value
)
MC_STATUS MC_Get_Next_Value_To_Int (
      int MsgFileItemID,
      unsigned long Tag,
      int* Value
)
MC_STATUS MC_Get_Next_Value_To_LongInt (
      int MsgFileItemID,
      unsigned long Tag,
      long int* Value
)
MC_STATUS MC_Get_Next_Value_To_ShortInt (
      int MsgFileItemID,
      unsigned long Tag,
      short int* Value
)
MC_STATUS MC_Get_Next_Value_To_String (
      int MsgFileItemID,
      unsigned long Tag,
      int BufferSize,
      char* Value
)
MC_STATUS MC_Get_Next_Value_To_UInt (
      int MsgFileItemID,
      unsigned long Tag,
      unsigned int* Value
)
MC_STATUS MC_Get_Next_Value_To_UShortInt (
      int MsgFileItemID,
      unsigned long Tag,
      unsigned short* Value
)
MC_STATUS MC_Get_Next_Value_To_ULongInt (
      int MsgFileItemID,
```

```
    unsigned long Tag,
    unsigned long* Value
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
Tag	DICOM tag which identifies the attribute.
DataType	One of the enumerated codes identifying the data type of the value in *Value*. The MC_DT enumerated type is defined in "mc3msg.h". They are:

String_Type	Null-terminated character string
Int_Type	Binary integer number
UInt_Type	Binary unsigned integer number
ShortInt_Type	Binary short integer number
UShortInt_Type	Binary unsigned short integer number
LongInt_Type	Binary long integer number
ULongInt_Type	Binary unsigned long integer number
Float_Type	Binary Floating point number

Value	The attribute's value will be returned here.
ValueSize	The size of the returned value is returned here.
BufferSize	The size of *Value* buffer in bytes.

Remarks

These functions fetch the next value of a multi-valued attribute. If no value has yet been fetched from this attribute, these function behaves identically to the functions of similar names, with "_Next" removed from the name. If one or more values have already been fetched from this attribute, the next value with be returned.

If **MC_Get_Next_Value** is used, the data type of the fetched value is specified by the *DataType* parameter. The other function names imply the data type requested. For example **MC_Get_Next_Value_To_Int** is the same as calling **MC_Get_Next_Value** with *DataType* specified as **Int_Type**. Each function will return the value requested in *Value*, which must be prototyped as the appropriate type. In the case of **MC_Get_Next_Value_To_String**, an additional parameter (*BufferSize*) is required to specify the size of the buffer to receive the string value. For **MC_Get_Next_Value,** *BufferSize* must be large enough to contain the data type requested.

Any reasonable conversion will be made from the attribute's value representation to the data type requested. However, some conversions are illogical (see the table below). An appropriate error code will be returned if the conversion attempt failed.

NOTE: Attributes with a value representation of **SQ** (sequence of items) have values stored internally as integers - each value is the *ItemID* of an item object opened previously using **MC_Open_Item**.

	Function may be used to retrieve values from attributes with these Value Representations
MC_Get_Next_Value_To_Int	**DS, FD, FL, IS, SL, SS, UL, US, SQ**
MC_Get_Next_Value_To_UInt	**DS, FD, FL, IS, SL, SS, UL, US, SQ**
MC_Get_Next_Value_To_ShortInt	**DS, FD, FL, IS, SL, SS, UL, US, SQ**
MC_Get_Next_Value_To_UShortInt	**DS, FD, FL, IS, SL, SS, UL, US, SQ**

MC_Get_Next_Value_To_LongInt	**AT, DS, FD, FL, IS, SL, SS, UL, US, SQ**
MC_Get_Next_Value_To_ULongInt	**AT, DS, FD, FL, IS, SL, SS, UL, US, SQ**
MC_Get_Next_Value_To_Float	**DS, FD, FL, IS, SL, SS, UL, US, SQ**
MC_Get_Next_Value_To_Double	**DS, FD, FL, IS, SL, SS, UL, US, SQ**
MC_Get_Next_Value_To_String	**AS, AT, CS, DA, DS, DT, FD, FL, IS, LO, LT, PN, SH, SL, SS, ST, TM, UI, UL, US, SQ**

The same rules apply to the **MC_Get_Next_Value** function, based on the value used in the *DataType* parameter.

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_EMPTY_VALUE	The attribute has no value assigned to it.
MC_NULL_VALUE	The attribute's value is NULL (i.e. it's length is zero).
MC_NO_MORE_VALUES	There are no more values for this attribute.
MC_NULL_POINTER_PARM	One or more of the pointer-type parameters was NULL.
MC_INVALID_TAG	The message does not contain an attribute with an ID of *Tag*.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.
MC_BUFFER_TOO_SMALL	Buffer is not large enough to contain the value. (**MC_Get_Next_Value** and **MC_Get_Next_Value_To_String** only)
MC_INCOMPATIBLE_VR	The function was called to retrieve an attribute value whose value representation(VR) is not one of those listed for the function in the table above.
MC_INCOMPATIBLE_VALUE	The attribute's value is not consistent with its VR. (This should never happen.)
MC_VALUE_OUT_OF_RANGE	The attribute numeric value is too large to be accommodated by the receiving data type. (E.g. Using **MC_Get_Value_To_Int** when the value is 123.45; or instances where the value's sign(+|–) would be affected)
MC_INVALID_DATA_TYPE	The *DataType* parameter is invalid.

See Also

MC_Get_pValue... Functions
MC_Get_Next_pValue... Functions
MC_Get_Next_Value... Functions
MC_Get_Value_To_Function

MC_Get_pAttribute_Info

Retrieves information about a given private attribute in a message.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Get_pAttribute_Info (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      MC_VR* ValueRep,
      int* NumValues
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
PrivateCode	The code string which identifies which block in the private *Group* "owns" the attribute.
Group	The number identifying the private group containing the private attribute. It must be an odd number.
ElementByte	The number identifying the private attribute within the private *Group* for this *PrivateCode*.
ValueRep	The attribute's value representation code is returned here. Possible return codes (defined in "mc3msg.h") are: **AE, AS, CS, DA, DS, DT, IS, LO, LT, PN, SH, ST, TM, UI, SS, US, AT, SL, UL, FL, FD, OB, OW, OF, SQ, UNKNOWN_VR**
NumValues	The number of values assigned to the attribute is placed here. (**NOTE**: will be one (1) if the attribute's value is NULL.)

Remarks

MC_Get_pAttribute_Info retrieves information about a message attribute identified by *ElementByte* for the *PrivateCode* within the given *Group*. The attribute's Value Representation code is returned at *ValueRep*, and the number of values stored for the attribute is returned at *NumValues*. If the attribute is "empty", *NumValues* will contain zero(0). If the attribute's value is NULL, *NumValues* will contain one(1).

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.
MC_INVALID_TAG	The message does not contain an attribute identified by *ElementByte* in *Group* for *PrivateCode*.
MC_INVALID_PRIVATE_CODE	The message does not contain attributes in *Group* for *PrivateCode.*
MC_INVALID_GROUP	*Group* was not an odd number.

MC_NULL_POINTER_PARM	One or more of the pointer-type parameters was NULL.

See Also

MC_Get_First_Attribute
MC_Get_Next_Attribute
MC_Get_Attribute_Info

MC_Get_pTag_Info

Retrieves a descriptive string for a given private tag.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Get_pTag_Info (
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      char* Name,
      int NameLength
)
```

PrivateCode	The code string which identifies which block in the private *Group* "owns" the attribute.
Group	The number identifying the private group containing the private attribute. It must be an odd number.
ElementByte	The number identifying the private attribute within the private *Group* for this *PrivateCode*.
Name	The tag's name is returned here.
NameLength	The length in bytes of *Name*.

Remarks

MC_Get_pTag_Info retrieves information about an attribute identified by *ElementByte* for the *PrivateCode* within the given *Group*. A descriptive test string for the attribute is returned at *Name*.

This function will only work with private attributes added to the MergeCOM-3's data dictionary through use of an extended tool kit.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_TAG	The MergeCOM-3 data dictionary does not contain an attribute identified by *ElementByte* in *Group* for *PrivateCode*.
MC_INVALID_GROUP	*Group* was not an odd number.
MC_NULL_POINTER_PARM	One or more of the pointer-type parameters was NULL.
MC_DDFILE_ERROR	An error occurred while trying to access the MergeCOM-3 data dictionary. A message describing the error has been written to the MergeCOM-3 log file.
MC_MISSING_CONFIG_PARM	This error occurs when the tool kit can not obtain the name of the data dictionary.
MC_BUFFER_TOO_SMALL	*Name* is not large enough to contain the descriptive string.

See Also

MC_Get_Tag_Info
MC_Get_Attribute_Info
MC_Get_pAttribute_Info

MC_Get_pValue... Functions

Retrieves the first or only value of a private attribute in a message object.

Synopsis

#include "mc3msg.h"

```
MC_STATUS MC_Get_pValue (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      MC_DT DataType,
      int BufferSize,
      void* Value,
      int* ValueSize
)
MC_STATUS MC_Get_pValue_To_Double (
(
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      double* Value
)
MC_STATUS MC_Get_pValue_To_Float (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      float* Value
)
MC_STATUS MC_Get_pValue_To_Int (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      int* Value
)
MC_STATUS MC_Get_pValue_To_ShortInt (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      short int* Value
)
MC_STATUS MC_Get_pValue_To_LongInt (
(
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      long int* Value
)
MC_STATUS MC_Get_pValue_To_String (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
```

```
      int BufferSize,
      char* Value
)
MC_STATUS MC_Get_pValue_To_UInt (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      unsigned int* Value
)
MC_STATUS MC_Get_pValue_To_UShortInt (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      unsigned short* Value
)
MC_STATUS MC_Get_pValue_To_ULongInt (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      unsigned long* Value
)
MC_STATUS MC_Get_pValue_To_Buffer (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      int BufferSize,
      void* Value,
      int* ValueSize
)
```

MsgFileItemID — The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.

PrivateCode — The code string which identifies which block in the private *Group* "owns" the attribute.

Group — The number identifying the private group containing the private attribute. It must be an odd number.

ElementByte — The number identifying the private attribute within the private *Group* for this *PrivateCode*.

DataType — One of the enumerated codes identifying the data type which should be used for the returned value in *Value*. The MC_DT enumerated type is defined in "mc3msg.h". They are:

String_Type	Null-terminated character string
Int_Type	Binary integer number
UInt_Type	Binary unsigned integer number
ShortInt_Type	Binary short integer number
UShortInt_Type	Binary unsigned short integer number
LongInt_Type	Binary long integer number
ULongInt_Type	Binary unsigned long integer number
Float_Type	Binary Floating point number

Value — The attribute's value will be returned here.

BufferSize — The size of the *Value* buffer in bytes.

ValueSize The size (in bytes) of the value returned in the *Value* buffer.

Remarks

These functions fetch the first (or only) value of a private attribute identified by *ElementByte* for the *PrivateCode* within the given private *Group*. If more than one value exists for this attribute, the first one will be returned. If **MC_Get_pValue** is used, the data type of the fetched value is specified by the *DataType* parameter. The other function names imply the data type requested. For example **MC_Get_pValue_To_Int** is the same as calling **MC_Get_pValue** with *DataType* specified as **Int_Type**. Each function will return the value requested in *Value*, which must be prototyped as the appropriate type. For **MC_Get_pValue_To_String and MC_Get_pValue_To_Buffer**, an additional parameter (*BufferSize*) is required to specify the size of the buffer to receive the string value. For **MC_Get_pValue,** *BufferSize* must be large enough to contain the data type requested.

MC_Get_pValue_To_Buffer is ONLY used to retrieve the value of a private attribute whose Value Representation is unknown. This may occur if **MC_Stream_To_Message** is used and one or more of the stream attributes is unknown. Using **MC_Get_pValue_To_Buffer** for attributes whose Value Representation is known will result in an error. The attribute's value is simply copied (memcpy) to the *Value* buffer and its length is placed at *ValueSize*.

Any reasonable conversion will be made from the attribute's value representation to the data type requested. However, some conversions are illogical (see the table below). An appropriate error code will be returned if the conversion attempt failed.

NOTE: Attributes with a value representation of **SQ** (sequence of items) have values stored internally as integers - each value is the *ItemID* of an item object opened previously using **MC_Open_Item**.

Function	**Value Representations** to use for retrieving values from attributes
MC_Get_pValue_To_Int	DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Get_pValue_To_UInt	DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Get_pValue_To_ShortInt	DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Get_pValue_To_UShortInt	DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Get_pValue_To_LongInt	AT, DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Get_pValue_To_ULongInt	AT, DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Get_pValue_To_Float	DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Get_pValue_To_Double	DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Get_pValue_To_String	AS, AT, CS, DA, DS, DT, FD, FL, IS, LO, LT, PN, SH, SL, SS, ST, TM, UI, UL, US, SQ
MC_Get_pValue_To_Function	OB, OW, OL,OF,SL, SS, UL, US, AT, FL, FD
MC_Get_pValue_To_Buffer	UNKNOWN_VR

The same rules apply to the **MC_Get_pValue** function, based on the value used in the *DataType* parameter. The **MC_Get_pValue_To_Function** call is described in its own section.

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	One or more of the pointer-type parameters was NULL.
MC_INVALID_TAG	The message does not contain an attribute identified by *ElementByte* in *Group* for *PrivateCode*.
MC_INVALID_PRIVATE_CODE	The message does not contain an attributes in *Group* for *PrivateCode.*
MC_INVALID_GROUP	*Group* was not an odd number.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.
MC_EMPTY_VALUE	The attribute has no value assigned to it.
MC_NULL_VALUE	The attribute's value is NULL (i.e. it's length is zero).
MC_BUFFER_TOO_SMALL	Buffer is not large enough to contain the value. (**MC_Get_pValue, MC_Get_pValue_To_Buffer** and **MC_Get_pValue_To_String** only)
MC_INCOMPATIBLE_VR	The function was called to retrieve an attribute value whose value representation(VR) is not one of those listed for the function in the table above.
MC_INCOMPATIBLE_VALUE	The attribute's value is not consistent with its VR. (This should never happen.)
MC_VALUE_OUT_OF_RANGE	The attribute numeric value is too large to be accommodated by the receiving data type. (E.g. Using **MC_Get_pValue_To_Int** when the value is 123.45; or instances where the value's sign(+|–) would be affected)
MC_INVALID_DATA_TYPE	The *DataType* parameter is invalid.

See Also

MC_Get_pValue_To_Function
MC_Get_Value_To_Function
MC_Get_Next_pValue... Functions
MC_Get_Value... Functions
MC_Get_Next_Value... Functions

MC_Get_pValue_Count

Returns the number of values assigned to a private attribute in a message object.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Get_pValue_Count (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      int* CountPtr
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
PrivateCode	The code string which identifies which block in the private *Group* "owns" the attribute.
Group	The number identifying the private group containing the private attribute. It must be an odd number.
ElementByte	The number identifying the private attribute within the private *Group* for this *PrivateCode*.
CountPtr	The number of values the attribute contains is returned here.

Remarks

MC_Get_pValue_Count returns the number of values assigned to a private attribute identified by *ElementByte* for the *PrivateCode* within the given *Group*. The number of returned at **CountPtr*.

Only certain attributes are allowed to contain multiple values and be conformant to the DICOM standard. The developer usually knows ahead of time which attributes will contain multiple values and which will not. Some attributes, such as those with value representations of OB or OW, may never contain more than one value.

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	*CountPtr* was NULL.
MC_INVALID_TAG	The message does not contain an attribute identified by *ElementByte* in *Group* for *PrivateCode*.
MC_INVALID_PRIVATE_CODE	The message does not contain an attributes in *Group* for *PrivateCode*.
MC_INVALID_GROUP	*Group* was not an odd number.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.
MC_NULL_VALUE	The attribute contains a NULL value (i.e. its value is of length zero).

MC_EMPTY_VALUE	No value has been assigned to the attribute.

See Also

MC_Get_Value_Count

MC_Get_pValue_Length

Returns the length of a private attribute value.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Get_pValue_Length (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      int ValueNumber,
      unsigned long* Length
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
PrivateCode	The code string which identifies which block in the private *Group* "owns" the attribute.
Group	The number identifying the private group containing the private attribute. It must be an odd number.
ElementByte	The number identifying the private attribute within the private *Group* for this *PrivateCode*.
ValueNumber	Retrieve the length of this attribute value.
Length	The value's length is returned here.

Remarks

MC_Get_pValue_Length returns the length of a given value of an attribute in a message object. The size, in bytes, is returned at **Length*.

The function returns the length of value *ValueNumber*. The first value is number 1. Only certain attributes are allowed to contain multiple values and be conformant to the DICOM standard. The developer usually knows ahead of time which attributes will contain multiple values and which will not. Some attributes, such as those with value representations of OB or OW, may never contain more than one value.

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_TAG	The message does not contain an attribute identified by *ElementByte* in *Group* for *PrivateCode*.
MC_NULL_POINTER_PARM	*Length* or *PrivateCode* was NULL.
MC_INVALID_PRIVATE_CODE	The message does not contain an attributes in *Group* for *PrivateCode.*
MC_INVALID_GROUP	*Group* was not an odd number.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.

MC_NULL_VALUE	The attribute contains a NULL value (i.e. its value is of length zero).
MC_EMPTY_VALUE	No value has been assigned to the attribute.
MC_INVALID_VALUE_NUMBER	*ValueNumber* was negative or the attribute does not have that many values.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.
MC_TEMP_FILE_ERROR	File I/O error accessing value information in temporary files.

See Also

MC_Get_Value_Length

MC_Get_pValue_To_Function

Retrieves the value of a private attribute which has a value representation of OB, OW , OL, or OF from a message object.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Get_pValue_To_Function (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      void* UserInfo,
      MC_STATUS (*YourGetFunction)()
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
PrivateCode	The code string which identifies which block in the private *Group* "owns" the attribute.
Group	The number identifying the private group containing the private attribute. It must be an odd number.
ElementByte	The number identifying the private attribute within the private *Group* for this *PrivateCode*.
UserInfo	Address of data which will be passed on to *YourGetFunction* each time it is called. This may be NULL.
YourGetFunction	Name of a function which will be called repeatedly to provide the function with blocks of the attribute's value.

The function must be prototyped as follows:

```
MC_STATUS YourGetFunction (
      int CBMsgFileItemID,
      unsigned long CBtag,
      void* CBuserInfo,
      int CBdataSize,
      void* CBdataBuffer,
      int CBisFirst,
      int CBisLast
)
```

CBMsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
CBtag	DICOM tag which identifies the attribute.
CBuserInfo	Address of user data which is being passed from the **MC_Get_pValue_To_Function** function. This may be NULL.
CBdataSize	The number of bytes in *CBdataBuffer*.
CBdataBuffer	Address of the buffer containing a portion of the attribute's value.
CBisFirst	This is TRUE (not zero) the first time ***YourGetFunction*** is being called for this attribute.

CBisLast	This is TRUE (not zero) the last time ***YourGetFunction*** is being called for this attribute.

Remarks

The **MC_Get_pValue_To_Function** function is used to fetch the value of a private attribute which has a value representation of OB, OW, or OF. Such attributes tend to have values of great length. To accommodate this, one uses the **MC_Get_pValue_To_Function** function is specify the name of a function (***YourGetFunction***) which MergeCOM-3, in turn, calls. This "callback" function is called repeatedly to provide blocks of the attribute's data value.

An optional *UserInfo* parameter may be used to pass information between the **MC_Get_pValue_To_Function** caller and ***YourGetFunction*** which receives the data in its *CBuserInfo* parameter. If no such exchange of data is necessary, you may use NULL for *UserInfo*.

YourGetFunction

Each block of data is returned in *CBdataBuffer* and the number of bytes in the block is specified by *CBdataSize*.

CBisFirst is set to TRUE the first time ***YourGetFunction*** is called. This provides a clear mechanism for the function to know it is being called the first time for the attribute identified by *CBtag* in the message identified by *CBMsgFileItemID*. Users often open a file at this time.

CBisLast is set to TRUE the last time ***YourGetFunction*** is called. This provides a clear mechanism for the function to know it is being called the last time for the attribute identified by *CBtag* in the message identified by *CBMsgFileItemID*. Users often close a file at this time.

YourGetFunction must return **MC_NORMAL_COMPLETION** if all went well. If not, it must return **MC_CANNOT_COMPLY**. Both of these are defined in "mc3msg.h".

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	One or more of the pointer-type parameters was NULL.
MC_INVALID_TAG	The message does not contain an attribute identified by *ElementByte* in *Group* for *PrivateCode*.
MC_INVALID_PRIVATE_CODE	The message does not contain an attributes in *Group* for *PrivateCode.*
MC_INVALID_GROUP	*Group* was not an odd number.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.
MC_INCOMPATIBLE_VR	The attribute's value representation was not OB, OW, or OF.
MC_CALLBACK_REGISTERED	This function is not allowed when the **MC_Register_Callback_Function** function was issued for the same attribute.

MC_EMPTY_VALUE	No value has been assigned to this attribute yet.
MC_CALLBACK_CANNOT_COMPLY	***YourGetFunction*** returned with **MC_CANNOT_COMPLY.**

See Also

MC_Get_Value_To_Function

MC_Get_Value
MC_Get_Value_To_Float **MC_Get_Value_To_Double**
MC_Get_Value_To_ShortInt **MC_Get_Value_To_UShortInt**
MC_Get_Value_To_LongInt **MC_Get_Value_To_Int**
MC_Get_Value_To_ULongInt **MC_Get_Value_To_Uint**
MC_Get_Value_To_NULL **MC_Get_Value_To_Empty**

MC_Get_pValue
MC_Get_pValue_To_Float **MC_Get_pValue_To_Double**
MC_Get_pValue_To_ShortInt **MC_Get_pValue_To_UShortInt**
MC_Get_pValue_To_LongInt **MC_Get_pValue_To_Int**
MC_Get_pValue_To_ULongInt **MC_Get_pValue_To_Uint**
MC_Get_pValue_To_NULL **MC_Get_pValue_To_Empty**

MC_Get_Next_Value
MC_Get_Next_Value_To_Float **MC_Get_Next_Value_To_Double**
MC_Get_Next_Value_To_ShortInt **MC_Get_Next_Value_To_UShortInt**
MC_Get_Next_Value_To_LongInt **MC_Get_Next_Value_To_Int**
MC_Get_Next_Value_To_ULongInt **MC_Get_Next_Value_To_UInt**

MC_Get_Next_pValue
MC_Get_Next_pValue_To_Float **MC_Get_Next_pValue_To_Double**
MC_Get_Next_pValue_To_ShortInt **MC_Get_Next_pValue_To_UShortInt**
MC_Get_Next_pValue_To_LongInt **MC_Get_Next_pValue_To_Int**
MC_Get_Next_pValue_To_ULongInt **MC_Get_Next_pValue_To_UInt**

MC_Get_Stream_Length

Returns the size of a DICOM stream which would result from using the **MC_Message_To_Stream** function.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Get_Stream_Length (
      int MessageID,
      unsigned long StartTag,
      unsigned long StopTag,
      unsigned long* LengthPtr,
      TRANSFER_SYNTAX SyntaxType
)
```

MessageID — The identifier assigned to this object by the MC_Open_Message function or the MC_Open_Item function.

StartTag — The DICOM tag identifying the first attribute which would be streamed.

StopTag — The DICOM tag identifying the last attribute which would be streamed

LengthPtr — Location where MergeCOM-3 will return the stream length.

SyntaxType — Specify which DICOM transfer syntax is to be assumed for the stream data. Use one of the enumerated TRANSFER_SYNTAX types defined in "mc3msg.h":

IMPLICIT_LITTLE_ENDIAN
IMPLICIT_BIG_ENDIAN
EXPLICIT_LITTLE_ENDIAN
EXPLICIT_BIG_ENDIAN
DEFLATED_EXPLICIT_LITTLE_ENDIAN
RLE
JPEG_BASELINE
JPEG_EXTENDED_2_4
JPEG_EXTENDED_3_5
JPEG_SPEC_NON_HIER_6_8
JPEG_SPEC_NON_HIER_7_9
JPEG_FULL_PROG_NON_HIER_10_12
JPEG_FULL_PROG_NON_HIER_11_13
JPEG_LOSSLESS_NON_HIER_14
JPEG_LOSSLESS_NON_HIER_15
JPEG_EXTENDED_HIER_16_18
JPEG_EXTENDED_HIER_17_19
JPEG_SPEC_HIER_20_22
JPEG_SPEC_HIER_21_23
JPEG_FULL_PROG_HIER_24_26
JPEG_FULL_PROG_HIER_25_27
JPEG_LOSSLESS_HIER_28
JPEG_LOSSLESS_HIER_29
JPEG_LOSSLESS_HIER_14
JPEG_2000_LOSSLESS_ONLY
JPEG_2000
JPEG_LS_LOSSLESS
JPEG_LS_LOSSY
MPEG2_MPML

PRIVATE_SYNTAX_1 or
PRIVATE_SYNTAX_2

Remarks

MC_Get_Stream_Length returns the size of a DICOM stream which would result from using the **MC_Message_To_Stream** function for the message identified by *MessageID*. The calculation is based on the assumption that the stream would be encoded using *SyntaxType* DICOM transfer syntax. It is also assumed that the stream would contain attributes with tags greater than or equal to *StartTag* and less than or equal to *StopTag*. The length is returned to the unsigned long at **LengthPtr*.

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_MESSAGE_ID	The *MessageID* value is not a valid message object ID.
MC_INVALID_TRANSFER_SYNTAX	*SyntaxType* is not one of the **TRANSFER_SYNTAX** codes defined in "mc3msg.h".
MC_NULL_POINTER_PARM	*LengthPtr* was NULL.

See Also

MC_Message_To_Stream

MC_Get_String_Config_Value

Used to get the value of a character string tool kit configuration parameter at runtime.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Get_String_Config_Value (
      StringParm Parm,
      int ValueLength,
      char* Value
)
```

Parm An enumerated constant identifying the character string configuration parameter to get. *Parm* can have any of the following values:

LOG_FILE,
LICENSE,
IMPLEMENTATION_CLASS_UID,
IMPLEMENTATION_VERSION,
LOCAL_APPL_CONTEXT_NAME,
IMPLICIT_LITTLE_ENDIAN_SYNTAX,
IMPLICIT_BIG_ENDIAN_SYNTAX,
EXPLICIT_LITTLE_ENDIAN_SYNTAX,
EXPLICIT_BIG_ENDIAN_SYNTAX,
DEFLATED_EXPLICIT_LITTLE_ENDIAN_SYNTAX,
RLE_SYNTAX,
JPEG_BASELINE_SYNTAX,
JPEG_EXTENDED_2_4_SYNTAX,
JPEG_EXTENDED_3_5_SYNTAX,
JPEG_SPEC_NON_HIER_6_8_SYNTAX,
JPEG_SPEC_NON_HIER_7_9_SYNTAX,
JPEG_FULL_PROG_NON_HIER_10_12_SYNTAX,
JPEG_FULL_PROG_NON_HIER_11_13_SYNTAX,
JPEG_LOSSLESS_NON_HIER_14_SYNTAX,
JPEG_LOSSLESS_NON_HIER_15_SYNTAX,
JPEG_EXTENDED_HIER_16_18_SYNTAX,
JPEG_EXTENDED_HIER_17_19_SYNTAX,
JPEG_SPEC_HIER_20_22_SYNTAX,
JPEG_SPEC_HIER_21_23_SYNTAX,
JPEG_FULL_PROG_HIER_24_26_SYNTAX,
JPEG_FULL_PROG_HIER_25_27_SYNTAX,
JPEG_LOSSLESS_HIER_28_SYNTAX,
JPEG_LOSSLESS_HIER_29_SYNTAX,
JPEG_LOSSLESS_HIER_14_SYNTAX,
JPEG_2000_LOSSLESS_ONLY_SYNTAX,
JPEG_2000_SYNTAX,
JPEG_LS_LOSSLESS_SYNTAX,
JPEG_LS_LOSSY_SYNTAX,
MPEG2_MPML_SYNTAX,
INITIATOR_NAME,
RECEIVER_NAME,
DICTIONARY_FILE,
MSG_INFO_FILE,
TEMP_FILE_DIRECTORY,
UNKNOWN_VR_CODE,

	PRIVATE_SYNTAX_1_SYNTAX, PRIVATE_SYNTAX_2_SYNTAX, CAPTURE_FILE, PEGASUS_OP_LIP3PLUS_NAME, PEGASUS_OP_LIE3PLUS_NAME, PEGASUS_OP_D2SEPLUS_NAME PEGASUS_OP_SE2DPLUS_NAME, PEGASUS_OP_J2KP_NAME, PEGASUS_OP_J2KE_NAME, PEGASUS_OP_LIP3PLUS_REGISTRATION, PEGASUS_OP_LIE3PLUS_REGISTRATION, PEGASUS_OP_D2SEPLUS_REGISTRATION, PEGASUS_OP_SE2DPLUS_REGISTRATION, PEGASUS_OP_J2KP_REGISTRATION, PEGASUS_OP_J2KE_REGISTRATION, LARGE_DATA_STORE, DICTIONARY_ACCESS, NULL_TYPE3_VALIDATION These names are the same as those given to the parameters in the tool kit configuration files.
ValueLength	The length of *Value*.
Value	A character string to copy the value of *Parm* into.

Remarks

The MergeCOM-3 Advanced Tool Kit Library accesses several configuration files at startup. This call allows your application to get character string configurable parameters specified in these files at runtime. Please see the detailed description of tool kit configuration elsewhere in this manual.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_PARAMETER_NAME	Parameter specified is not one of the above listed parameters.
MC_NULL_POINTER_PARM	*Value* was a null pointer.
MC_BUFFER_TOO_SMALL	*Value* is not large enough to contain the configuration string.
MC_LIBRARY_NOT_INITIALIZED	The library has not been properly initialized.

See Also

MC_Get_Int_Config_Value
MC_Get_Log_Destination
MC_Get_Long_Config_Value
MC_Set_String_Config_Value
MC_Set_Int_Config_Value
MC_Set_Log_Destination
MC_Set_Long_Config_Value

MC_Get_Tag_Info

Retrieves a descriptive string for a given tag.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Get_Tag_Info (
      unsigned long Tag,
      char* Name,
      int NameLength
)
```

Tag	The identifier assigned to this object by the **MC_Open_Message** function or the **MC_Open_Item** function.
Name	The tag's name is returned here.
NameLength	The length in bytes of *Name*.

Remarks

MC_Get_Tag_Info retrieves information about an attribute identified by *Tag*. A descriptive text string for the attribute is returned at *Name*.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_TAG	The attribute identified by *Tag* is not a valid DICOM attribute.
MC_NULL_POINTER_PARM	One or more of the pointer-type parameters was NULL.
MC_DDFILE_ERROR	An error occurred while trying to access the MergeCOM-3 data dictionary. A message describing the error has been written to the MergeCOM-3 log file.
MC_MISSING_CONFIG_PARM	This error occurs when the tool kit can not obtain the name of the data dictionary.
MC_BUFFER_TOO_SMALL	*Name* is not large enough to contain the descriptive string.

See Also

MC_Get_pTag_Info
MC_Get_Attribute_Info
MC_Get_pAttribute_Info

MC_Get_Tags_Dict_Info

Retrieves all dictionary information for a given tag.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Get_Tags_Dict_Info (
      unsigned long Tag,
      char* Name,
      int NameSize,
      MC_VR* ValueRep,
      unsigned short* ValueMult_low,
      unsigned short* ValueMult_high
)
```

Tag	A tag identifying a message attribute.
Name	A buffer to contain the tag's name.
NameSize	The size, in bytes, of *Name*
ValueRep	The attribute's value representation code is returned here. Possible return codes (defined in "mc3msg.h") are: **AE, AS, CS, DA, DS, DT, IS, LO, LT, PN, SH, ST, TM, UI, SS, US, AT, SL, UL, FL, FD, OB, OW, OF, SQ, UNKNOWN_VR**
ValueMult_low	The minimum multiplicity will be returned here
ValueMult_high	The maximum multiplicity will be returned here

Remarks

MC_Get_Tags_Dict_Info retrieves information about an attribute identified by *Tag*. The attribute's Name is returned at *Name, v*alue representation code is returned at *ValueRep*, and the value multiplicities are returned in *ValueMult_low* and *ValueMult_high*. The values returned are from the dictionary file.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	*Name, ValueRep, ValueMult_low, ValueMult_High, and/*or *Tag* are invalid.
MC_BUFFER_TOO_SMALL	*NameSize* is not large enough to contain *Name.*
MC_MISSING_CONFIG_PARM	DICTIONARY_FILE is invalid
MC_DDFILE_ERROR	Data Dictionary problem
MC_INVALID_TAG	Tag is not found

See Also

MC_Get_Attribute_Info

MC_Get_Transfer_Syntax_From_Enum

Gets the DICOM UID for a transfer syntax.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Get_Transfer_Syntax_From_Enum (
      TRANSFER_SYNTAX SyntaxType,
      char* Uid,
      int UidLength
)
```

SyntaxType — Specify the DICOM transfer syntax for which to find a UID. Use one of the enumerated TRANSFER_SYNTAX types defined in mc3msg.h:

IMPLICIT_LITTLE_ENDIAN
IMPLICIT_BIG_ENDIAN
EXPLICIT_LITTLE_ENDIAN
EXPLICIT_BIG_ENDIAN
DEFLATED_EXPLICIT_LITTLE_ENDIAN
RLE
JPEG_BASELINE
JPEG_EXTENDED_2_4
JPEG_EXTENDED_3_5
JPEG_SPEC_NON_HIER_6_8
JPEG_SPEC_NON_HIER_7_9
JPEG_FULL_PROG_NON_HIER_10_12
JPEG_FULL_PROG_NON_HIER_11_13
JPEG_LOSSLESS_NON_HIER_14
JPEG_LOSSLESS_NON_HIER_15
JPEG_EXTENDED_HIER_16_18
JPEG_EXTENDED_HIER_17_19
JPEG_SPEC_HIER_20_22
JPEG_SPEC_HIER_21_23
JPEG_FULL_PROG_HIER_24_26
JPEG_FULL_PROG_HIER_25_27
JPEG_LOSSLESS_HIER_28
JPEG_LOSSLESS_HIER_29
JPEG_LOSSLESS_HIER_14
JPEG_2000_LOSSLESS_ONLY
JPEG_2000
PRIVATE_SYNTAX_1 or
PRIVATE_SYNTAX_2

Uid — The transfer syntax's UID is returned here.

UidLength — The length in bytes of *Uid*.

Remarks

MC_Get_Transfer_Syntax_From_Enum converts a MergeCOM-3 enumerated value that represents a DICOM transfer syntax into the UID for that transfer syntax**.** The UID values used by **MC_Get_Transfer_Syntax_From_Enum** are specified in MergeCOM-3's configuration files. Please see the detailed description of tool kit configuration elsewhere in this manual.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	One or more of the pointer-type parameters was NULL.
MC_INVALID_TRANSFER_SYNTAX	*SyntaxName* is not registered in the MergeCOM-3 configuration files.
MC_BUFFER_TOO_SMALL	*Uid* is not large enough to contain the unique identifier.

See Also

MC_Get_Enum_From_Transfer_Syntax
MC_Get_MergeCOM_Service
MC_Get_UID_From_MergeCOM_Service

MC_Get_UID_From_MergeCOM_Service

Converts a MergeCOM-3 service name into a DICOM SOP Class UID.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Get_UID_From_MergeCOM_Service (
      char* ServiceName,
      char* Uid,
      int UidLength
)
```

ServiceName	String name of a DICOM service
Uid	The service's SOP Class UID is returned here.
UidLength	The length in bytes of *Uid*.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	One or more of the pointer-type parameters was NULL.
MC_UNKNOWN_SERVICE	*ServiceName* was not registered in the MergeCOM-3 configuration files.
MC_BUFFER_TOO_SMALL	*Uid* is not large enough to contain the unique identifier.

See Also

MC_Get_MergeCOM_Service
MC_Get_Enum_From_Transfer_Syntax
MC_Get_Transfer_Syntax_From_Enum

MC_Get_Value... Functions

Retrieves the first or only value of an attribute in a message object.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Get_Value (
      int MsgFileItemID,
      unsigned long Tag,
      MC_DT DataType,
      int BufferSize,
      void* Value,
      int* ValueSize
)
MC_STATUS MC_Get_Value_To_Double (
      int MsgFileItemID,
      unsigned long Tag,
      double* Value
)
MC_STATUS MC_Get_Value_To_Float (
      int MsgFileItemID,
      unsigned long Tag,
      float* Value
)
MC_STATUS MC_Get_Value_To_Int (
      int MsgFileItemID,
      unsigned long Tag,
      int* Value
)
MC_STATUS MC_Get_Value_To_ShortInt (
      int MsgFileItemID,
      unsigned long Tag,
      short int* Value
)
MC_STATUS MC_Get_Value_To_LongInt (
      int MsgFileItemID,
      unsigned long Tag,
      long int* Value
)
MC_STATUS MC_Get_Value_To_String (
      int MsgFileItemID,
      unsigned long Tag,
      int BufferSize,
      char* Value
)
MC_STATUS MC_Get_Value_To_UInt (
      int MsgFileItemID,
      unsigned long Tag,
      unsigned int* Value
)
MC_STATUS MC_Get_Value_To_UShortInt (
      int MsgFileItemID,
      unsigned long Tag,
      unsigned short* Value
)
```

```
MC_STATUS MC_Get_Value_To_ULongInt (
      int MsgFileItemID,
      unsigned long Tag,
      unsigned long* Value
)
MC_STATUS MC_Get_Value_To_Buffer (
      int MsgFileItemID,
      unsigned long Tag,
      int BufferSize,
      char* Value,
      int* ValueSize
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
Tag	DICOM tag which identifies the attribute.
DataType	One of the enumerated codes identifying the data type which should be used for the returned value in *Value*. The MC_DT enumerated type is defined in "mc3msg.h". They are:

String_Type	Null-terminated character string
Int_Type	Binary integer number
UInt_Type	Binary unsigned integer number
ShortInt_Type	Binary short integer number
UShortInt_Type	Binary unsigned short integer number
LongInt_Type	Binary long integer number
ULongInt_Type	Binary unsigned long integer number
Float_Type	Binary Floating point number

Value	The attribute's value will be returned here.
BufferSize	The size of the *Value* buffer in bytes.
ValueSize	The size (in bytes) of the value returned in the *Value* buffer. If *DataType* is **String_Type**, the length of the string (not including the trailing null) is returned.

Remarks

These functions fetch the first (or only) value of an attribute with the given *Tag*. If more than one value exists for this attribute, the first one will be returned. If **MC_Get_Value** is used, the data type of the fetched value is specified by the *DataType* parameter. The other function names imply the data type requested. For example **MC_Get_Value_To_Int** is the same as calling **MC_Get_Value** with *DataType* specified as **Int_Type**. Each function will return the value requested in *Value*, which must be prototyped as the appropriate type. For **MC_Get_Value_To_String** and **MC_Get_Value_To_Buffer**, an additional parameter (*BufferSize*) is required to specify the size of the buffer to receive the string value. For **MC_Get_Value,** *BufferSize* must be large enough to contain the data type requested.

MC_Get_Value_To_Buffer is ONLY used to retrieve the value of an attribute whose Value Representation is unknown. This may occur if **MC_Stream_To_Message** is used and one or more of the stream attributes is unknown. Using **MC_Get_Value_To_Buffer** for attributes whose Value Representation is known will result in an error. The attribute's value is simply copied (memcpy) to the *Value* buffer and its length is placed at *ValueSize*.

Any reasonable conversion will be made from the attribute's value representation to the data type requested. However, some conversions are illogical (see the table below). An appropriate error code will be returned if the conversion attempt failed.

NOTE: Attributes with a value representation of **SQ** (sequence of items) have values stored internally as integers - each value is the *ItemID* of an item object opened previously using **MC_Open_Item**.

Function	**Value Representations** to use for retrieving values from attributes
MC_Get_Value_To_Int	DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Get_Value_To_UInt	DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Get_Value_To_ShortInt	DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Get_Value_To_UShortInt	DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Get_Value_To_LongInt	AT, DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Get_Value_To_ULongInt	AT, DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Get_Value_To_Float	DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Get_Value_To_Double	DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Get_Value_To_String	AS, AT, CS, DA, DS, DT, FD, FL, IS, LO, LT, PN, SH, SL, SS, ST, TM, UI, UL, US, SQ
MC_Get_Value_To_Function	OB, OW, OF, SL, SS, UL, US, AT, FL, FD
MC_Get_Value_To_Buffer	UNKNOWN_VR

The same rules apply to the **MC_Get_Value** function, based on the value used in the *DataType* parameter. The **MC_Get_Value_To_Function** call is described in its own section.

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	**Meaning**
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	One or more of the pointer-type parameters was NULL.
MC_INVALID_TAG	The message does not contain an attribute with an ID of *Tag*.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.
MC_EMPTY_VALUE	The attribute has no value assigned to it.
MC_NULL_VALUE	The attribute's value is NULL (i.e. it's length is zero).
MC_BUFFER_TOO_SMALL	Buffer is not large enough to contain the value. (**MC_Get_Value, MC_Get_Value_To_Buffer** and **MC_Get_Value_To_String** only)
MC_INCOMPATIBLE_VR	The function was called to retrieve an attribute value whose value representation(VR) is not one of those listed for the function in the table above.

MC_INCOMPATIBLE_VALUE	The attribute's value is not consistent with its VR. (This should never happen.)
MC_VALUE_OUT_OF_RANGE	The attribute numeric value is too large to be accommodated by the receiving data type. (E.g. Using **MC_Get_Value_To_Int** when the value is 123.45; or instances where the value's sign(+|–) would be affected)
MC_INVALID_DATA_TYPE	The *DataType* parameter is invalid.

See Also

MC_Get_pValue... Functions
MC_Get_pValue_To_Function
MC_Get_Next_pValue... Functions
MC_Get_Value_To_Function
MC_Get_Next_Value... Functions

MC_Get_Value_Count

Returns the number of values assigned to an attribute in a message object.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Get_Value_Count (
      int MsgFileItemID,
      unsigned long Tag,
      int* CountPtr
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
Tag	DICOM tag which identifies the attribute.
CountPtr	The number of values the attribute contains is returned here.

Remarks

MC_Get_Value_Count returns the number of values assigned to an attribute in a message object. The number is returned at **CountPtr*.

Only certain attributes are allowed to contain multiple values and be conformant to the DICOM standard. The developer usually knows ahead of time which attributes will contain multiple values and which will not. Some attributes, such as those with value representations of OB or OW, may never contain more than one value.

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	*CountPtr* was NULL.
MC_INVALID_TAG	The message does not contain an attribute with an ID of *Tag*.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.
MC_NULL_VALUE	The attribute contains a NULL value (i.e. its value is of length zero).
MC_EMPTY_VALUE	No value has been assigned to the attribute.

See Also

MC_Get_pValue_Count

MC_Get_Value_Length

Returns the length of an attribute value.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Get_Value_Length (
      int MsgFileItemID,
      unsigned long Tag,
      int ValueNumber,
      unsigned long* Length
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
Tag	DICOM tag which identifies the attribute.
ValueNumber	Retrieve the length of this attribute value.
Length	The value's length is returned here.

Remarks

MC_Get_Value_Length returns the length of a given value of an attribute in a message object. The size, in bytes, is returned at **Length*.

The function returns the length of value *ValueNumber*. The first value is number 1. Only certain attributes are allowed to contain multiple values and be conformant to the DICOM standard. The developer usually knows ahead of time which attributes will contain multiple values and which will not. Some attributes, such as those with value representations of OB or OW, may never contain more than one value.

If the attribute has a value representation of SQ, the number of items in the sequence is returned in **Length*.

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	*Length* was NULL.
MC_INVALID_TAG	The message does not contain an attribute with an ID of *Tag*.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.
MC_NULL_VALUE	The attribute contains a NULL value (i.e. its value is of length zero).
MC_EMPTY_VALUE	No value has been assigned to the attribute.
MC_INVALID_VALUE_NUMBER	*ValueNumber* was negative or the attribute does not have that many values.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

MC_TEMP_FILE_ERROR	File I/O error accessing value information in temporary files.

See Also

MC_Get_pValue_Length

MC_Get_Value_To_Function

Retrieves the value of an attribute which has a value representation of OB, OW, or OF from a message object.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Get_Value_To_Function (
      int MsgFileItemID,
      unsigned long Tag,
      void* UserInfo,
      MC_STATUS (*YourGetFunction)()
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
Tag	DICOM tag which identifies the attribute.
UserInfo	Address of data which will be passed on to *YourGetFunction* each time it is called. This may be NULL.
YourGetFunction	Name of a function which will be called repeatedly to provide the function with blocks of the attribute's value.

The function must be prototyped as follows:

```
MC_STATUS YourGetFunction (
      int CBMsgFileItemID,
      unsigned long CBtag,
      void* CBuserInfo,
      int CBdataSize,
      void* CBdataBuffer,
      int CBisFirst,
      int CBisLast
)
```

CBMsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
CBtag	DICOM tag which identifies the attribute.
CBuserInfo	Address of user data which is being passed from the **MC_Get_Value_To_Function** function. This may be NULL.
CBdataSize	The number of bytes in *CBdataBuffer*.
CBdataBuffer	Address of the buffer containing a portion of the attribute's value.
CBisFirst	This is TRUE (not zero) the first time ***YourGetFunction*** is being called for this attribute.
CBisLast	This is TRUE (not zero) the last time ***YourGetFunction*** is being called for this attribute.

Remarks

The **MC_Get_Value_To_Function** function is used to fetch the value of an attribute which has a value representation of OB, OW, or OF. Such attributes tend to have values of great length. To accommodate this, one uses the **MC_Get_Value_To_Function**

function is specify the name of a function (***YourGetFunction***) which MergeCOM-3, in turn, calls. This "callback" function is called repeatedly to provide blocks of the attribute's data value. An optional *UserInfo* parameter may be used to pass information between the **MC_Get_Value_To_Function** caller and ***YourGetFunction*** which receives the data in its *CBuserInfo* parameter. If no such exchange of data is necessary, you may use NULL for *UserInfo*.

YourGetFunction

Each block of data is returned in *CBdataBuffer* and the number of bytes in the block is specified by *CBdataSize*.

CBisFirst is set to TRUE the first time ***YourGetFunction*** is called. This provides a clear mechanism for the function to know it is being called the first time for the attribute identified by *CBtag* in the message identified by *CBMsgFileItemID*. Users often open a file at this time.

CBisLast is set to TRUE the last time ***YourGetFunction*** is called. This provides a clear mechanism for the function to know it is being called the last time for the attribute identified by *CBtag* in the message identified by *CBMsgFileItemID*. Users often close a file at this time.

YourGetFunction must return **MC_NORMAL_COMPLETION** if all went well. If not, it must return **MC_CANNOT_COMPLY**. Both of these are defined in "mc3msg.h".

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	One or more of the pointer-type parameters was NULL.
MC_INVALID_TAG	The message does not contain an attribute with an ID of *Tag*.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.
MC_INCOMPATIBLE_VR	The attribute's value representation was not OB, OW, or OF.
MC_CALLBACK_REGISTERED	This function is not allowed when the **MC_Register_Callback_Function** function was issued for the same attribute.
MC_EMPTY_VALUE	No value has been assigned to this attribute yet.
MC_CALLBACK_CANNOT_COMPLY	***YourGetFunction*** returned with **MC_CANNOT_COMPLY.**

See Also

MC_Get_Value... Functions
MC_Get_pValue... Functions
MC_Get_pValue_To_Function
MC_Get_Next_Value... Functions
MC_Get_Next_pValue... Functions

MC_Get_Version_String

Retrieves a descriptive string containing the tool kit version.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Get_Version_String (
      int VersionLength,
      char* Version
)
```

VersionLength	The length in bytes of *Version*.
Version	The MergeCOM-3 version is returned here.

Remarks

MC_Get_Version_String retrieves a text string containing MergeCOM-3's version.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	*Version* was NULL.
MC_BUFFER_TOO_SMALL	*Version* is not large enough to contain the descriptive string.

See Also

MC_Library_Initialization

MC_Library_Initialization

Prepares the MergeCOM library and provides optional information to the library.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Library_Initialization (
      void* (*CfgFunction) (void),
      void* (*DictionaryFunction) (void),
      void* (*FutureFunction) (void)
)
```

CfgFunction	The name of the function generated by the genconf utility, or NULL. (see below)
DictionaryFunction	The name of the function generated by the gendict utility, or NULL. (see below)
FutureFunction	NULL. This argument is reserved for future use.

Remarks

MC_Library_Initialization prepares the library to accept other API function calls. If any of the three parameters are set to NULL, no special initialization options have been selected. If a parameter is not NULL, **MC_Library_Initialization** calls the functions specified to initialize configuration or dictionary data structures.

MC_Library_Initialization must be the first library function call made!

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_CONFIG_INFO_ERROR	An error has occurred while trying to initialize configuration information. A message describing the error has been written to the MergeCOM-3 log file.
MC_DDFILE_ERROR	An error has occurred while trying to access the MergeCOM-3 data dictionary. A message describing the error has been written to the MergeCOM-3 log file.
MC_LIBRARY_ALREADY_INITIALIZED	The **MC_Library_Initialization** has already been called and the library is already initialized.
MC_MISSING_CONFIG_PARM	This error occurs when the tool kit can not obtain the name of the data dictionary.
MC_NO_FILE_SYSTEM	This error occurs only in tool kits running on platforms that do not have a file system. When this error occurs, no configuration function has been specified for the initialization of the configuration information.

MC_NO_MERGE_INI	This error occurs when the tool kit can not open the tool kit initialization file. See the section titled “Configuration” in this manual and the *MergeCOM-3 Advanced Tool Kit Users Manual* for a detailed discussion of the initialization file(s).
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Library_Reset
MC_Library_Release
MC_Set_MergeINI

MC_Library_Release

Releases all resources used by the MergeCOM-3 library.

Synopsis

```
#include "mc3msg.h"
MC_STATUS MC_Library_Release (void)
```

Remarks

MC_Library_Release releases all resources used by the library. This option is normally used before exiting an application. **MC_Library_Initialization** must be called before the library can be used again.

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_LIBRARY_NOT_INITIALIZED	The library has not been initialized by a call to **MC_Library_Initialization**.

See Also

MC_Library_Initialization **MC_Library_Reset** MC_Set_MergeINI

MC_Library_Reset

Sets the MergeCOM-3 library back to its initial state.

Synopsis

```
#include "mc3msg.h"
MC_STATUS MC_Library_Reset (void)
```

Remarks

MC_Library_Reset resets the library to its initial state. This call is normally used when the library is used for an embedded application. The same options specified by the **MC_Library_Initialization** function call will be in effect. This function is not normally called.

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_LIBRARY_NOT_INITIALIZED	The library has not been initialized by a call to **MC_Library_Initialization**.
MC_CONFIG_INFO_ERROR	An error occurred while trying to initialize configuration information. A message describing the error is written to the MergeCOM-3 log file.
MC_DDFILE_ERROR	An error occurred while trying to access the MergeCOM-3 data dictionary. A message describing the error is written to the MergeCOM-3 log file.
MC_MISSING_CONFIG_PARM	This error occurs when the tool kit cannot obtain the name of the data dictionary.
MC_NO_FILE_SYSTEM	This error occurs only in toolkits running on platforms that do not have a file system. When this error occurs, no configuration function has been specified for the initialization of the configuration information.
MC_NO_MERGE_INI	This error occurs when the tool kit cannot open the toolkit initialization file. Refer to *Configuration* in this manual and the *MergeCOM-3 Advanced Tool Kit Users Manual* for more initialization-file details.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error is written to the MergeCOM-3 log file.

See Also

MC_Library_Initialization
MC_Library_Release
MC_Set_MergeINI

MC_List_File (All toolkits except Windows versions)

Prepares a listing of the current contents of a file object.

Synopsis

```
#include "mc3media.h"

void MC_List_File (
      int FileID,
      FILE* StreamHandle
)
```

FileID	The identifier assigned to this object by the **MC_Create_File** or **MC_Create_Empty_File** function.
StreamHandle	The handle of an open stream. If NULL, the listing will be directed to STDOUT.

Remarks

MC_List_File prepares a report listing the current contents of the file object identified by *FileID*. The filename and preamble associated with the file object will also be listed. The report will be written to the stream identified by *StreamHandle*. Use NULL to direct the report to STDOUT.

NOTE: If the file object contains an attribute with a value representation of SQ (sequence of items), each item in the sequence will be listed. Each sequence of items is indented in the listing four spaces to the right of its owner DICOMDIR, file, message, or item.

Return Value

There is no return value.

See Also

MC_List_Message
MC_List_Item

MC_List_File (Windows toolkit versions)

Prepares a listing of the current contents of a file object.

Synopsis

```
#include "mc3media.h"

void MC_List_File (
      int FileID,
      char* Afilename
)
```

FileID	The identifier assigned to this object by the **MC_Create_File** or **MC_Create_Empty_File** function.
Afilename	The name of file to open. If NULL and using the static library, the listing will be directed to STDOUT. If NULL and using the DLL library, an exception will occur.

Remarks

Because `FILE*` variables cannot be passed to a DLL, we have changed this function on Windows platforms to pass a filename instead. With the static library, the output will be sent to stdout when the Afilename parameter is set to NULL. If it is set to NULL when using the DLL, an exception error will occur. When Afilename is set to a filename, the message, item, or file object is listed to this text file.

MC_List_File prepares a report listing the current contents of the file object identified by *FileID*. The filename and preamble associated with the file object will also be listed. The report will be written to the file identified by *Afilename*. Use NULL to direct the report to STDOUT **only** when linked with the static toolkit library.

NOTE: If the file object contains an attribute with a value representation of SQ (sequence of items), each item in the sequence will be listed. Each sequence of items is indented in the listing four spaces to the right of its owner DICOMDIR, file, message, or item.

Return Value

There is no return value.

See Also

MC_List_Message
MC_List_Item

MC_List_Item (All toolkits except Windows versions)

Prepares a listing of the current contents of an item object.

Synopsis

```
#include "mc3msg.h"

void MC_List_Item (
      int ItemID,
      FILE* StreamHandle
)
```

ItemID	The identifier assigned to this message object by the **MC_Open_Item** function.
StreamHandle	The handle of an open stream. If NULL, the listing will be directed to STDOUT.

Remarks

MC_List_Item prepares a report listing the contents of the current contents of the item object identified by *ItemID*. The report will be written to the stream identified by *StreamHandle*. Use NULL to direct the report to STDOUT.

NOTE: An item is listed automatically if the owning message object is listed by **MC_List_Message**, or if the owning item object is listed by **MC_List_Item**.

Return Value

There is no return value.

See Also

MC_List_Message

MC_List_Item (Windows toolkit versions)

Prepares a listing of the current contents of an item object.

Synopsis

```
#include "mc3msg.h"

void MC_List_Item (
      int ItemID,
      char* Afilename
)
```

ItemID	The identifier assigned to this message object by the **MC_Open_Item** function.
Afilename	The name of file to open. If NULL and using the static library, the listing will be directed to STDOUT. If NULL and using the DLL library, an exception will occur.

Remarks

Because `FILE*` variables cannot be passed to a DLL, we have changed this function on Windows platforms to pass a filename instead. With the static library, the output will be sent to stdout when the Afilename parameter is set to NULL. If it is set to NULL when using the DLL, an exception error will occur. When Afilename is set to a filename, the message, item, or file object is listed to this text file.

MC_List_Item prepares a report listing the contents of the current contents of the item object identified by *ItemID*. The report will be written to the stream identified by *Afilename*. Use NULL to direct the report to STDOUT **only** when linked with the static toolkit library.

NOTE: An item is listed automatically if the owning message object is listed by **MC_List_Message**, or if the owning item object is listed by **MC_List_Item**.

Return Value

There is no return value.

See Also

MC_List_Message

MC_List_Message (All toolkits except Windows versions)

Prepares a listing of the current contents of a message object.

Synopsis

```
#include "mc3msg.h"

void MC_List_Message (
      int MessageID,
      FILE* StreamHandle
)
```

MessageID	The identifier assigned to this object by the MC_Open_Message function or the MC_Open_Item function.
StreamHandle	The handle of an open stream. If NULL, the listing will be directed to STDOUT.

Remarks

MC_List_Message prepares a report listing the contents of the current contents of the message object identified by *MessageID*. The report will be written to the stream identified by *StreamHandle*. Use NULL to direct the report to STDOUT.

NOTE: If the message object contains an attribute with a value representation of SQ (sequence of items), each item in the sequence will be listed. Each sequence of items is indented in the listing four spaces to the right of its owning message or item.

Return Value

There is no return value.

See Also

MC_List_Item

MC_List_Message (Windows toolkit versions)

Prepares a listing of the current contents of a message object.

Synopsis

```
#include "mc3msg.h"

void MC_List_Message (
      int MessageID,
      char* Afilename
)
```

MessageID	The identifier assigned to this object by the MC_Open_Message function or the MC_Open_Item function.
Afilename	The name of file to open. If NULL and using the static library, the listing will be directed to STDOUT. If NULL and using the DLL library, an exception will occur.

Remarks

Because `FILE*` variables cannot be passed to a DLL, we have changed this function on Windows platforms to pass a filename instead. With the static library, the output will be sent to stdout when the Afilename parameter is set to NULL. If it is set to NULL when using the DLL, an exception error will occur. When Afilename is set to a filename, the message, item, or file object is listed to this text file.

MC_List_Message prepares a report listing the contents of the current contents of the message object identified by *MessageID*. The report will be written to the stream identified by *Afilename*. Use NULL to direct the report to STDOUT **only** when linked with the static toolkit library.

NOTE: If the message object contains an attribute with a value representation of SQ (sequence of items), each item in the sequence will be listed. Each sequence of items is indented in the listing four spaces to the right of its owning message or item.

Return Value

There is no return value.

See Also

MC_List_Item

MC_Message_To_File

Changes a message object into a file object

Synopsis

```
#include "mc3media.h"

MC_STATUS MC_Message_To_File (
      int MessageID,
      char* Filename
)
```

MessageID	The identifier assigned to this object by the **MC_Open_Message** function.
Filename	A pointer to a string containing the filename to be associated with the new file object.

Remarks

MC_Message_To_File changes the message object pointed to by *MessageID* into a file object. In the process, the "command type" attributes are removed from the message object, and the DICOM File Meta Information attributes are added to the new file object. These new attributes must be given values by the user. If the message was opened as an empty message and the command and service were not set, the command and service must be set for the converted file object before validating.

The DICOM prefix for the file is set to "DICM". All bytes in the file preamble are set to 00H.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_MESSAGE_ID	The *MessageID* value is not a valid message object ID.
MC_NULL_POINTER_PARM	*Filename* has a value of NULL.

See Also

MC_File_To_Message

MC_Message_To_SR

Converts a message created with the toolkit's message API calls (MC_Open_Message, etc) into an SR tree management object.

Synopsis

```
#include "mc3sra.h"

MC_STATUS MC_Message_To_SR (
      int MsgID
)
```

MsgID The identifier assigned to a SR message object by the **MC_Open_Message** functions.

Remarks

MC_Message_To_SR converts a message that was created with the toolkit's message API calls into an SR tree management object.

For example, a STANDARD_BASIC_TEXT_SR, C_STORE_RQ message has been created and manipulated by the toolkit messaging API calls, **MC_Message_To_SR** can be used to convert this message object into an SR tree management object. Once this function call has been issued, further changes to the SR toolkit message can be accomplished through the use of the MC_SR… API calls.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_MESSAGE_ID	The *MsgID* value is not a valid object ID.
MC_MSGFILE_ERROR	An error occurred attempting to access the configuration data file for the directory record.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_SR_Add_Child
MC_SR_Get_First_Child
MC_ SR_Get_Next_Child
MC_ SR_Get_Root
MC_SR_Delete_Child

MC_Message_To_Stream

Request that the values of a message object be returned as a DICOM stream.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Message_To_Stream (
      int MessageID,
      unsigned long StartTag,
      unsigned long StopTag,
      TRANSFER_SYNTAX SyntaxType,
      void* UserInfo,
      MC_STATUS (*YourReceiveStreamFunction)()
)
```

MessageID	The identifier assigned to this object by the MC_Open_Message function or the MC_Open_Item function.
StartTag	The DICOM tag identifying the first attribute which should be streamed.
StopTag	The DICOM tag identifying the last attribute which should be streamed
SyntaxType	Specify which DICOM transfer syntax is to be used to encode the stream data. Use one of the enumerated TRANSFER_SYNTAX types defined in "mc3msg.h": IMPLICIT_LITTLE_ENDIAN IMPLICIT_BIG_ENDIAN EXPLICIT_LITTLE_ENDIAN EXPLICIT_BIG_ENDIAN DEFLATED_EXPLICIT_LITTLE_ENDIAN RLE JPEG_BASELINE JPEG_EXTENDED_2_4 JPEG_EXTENDED_3_5 JPEG_SPEC_NON_HIER_6_8 JPEG_SPEC_NON_HIER_7_9 JPEG_FULL_PROG_NON_HIER_10_12 JPEG_FULL_PROG_NON_HIER_11_13 JPEG_LOSSLESS_NON_HIER_14 JPEG_LOSSLESS_NON_HIER_15 JPEG_EXTENDED_HIER_16_18 JPEG_EXTENDED_HIER_17_19 JPEG_SPEC_HIER_20_22 JPEG_SPEC_HIER_21_23 JPEG_FULL_PROG_HIER_24_26 JPEG_FULL_PROG_HIER_25_27 JPEG_LOSSLESS_HIER_28 JPEG_LOSSLESS_HIER_29 JPEG_LOSSLESS_HIER_14 JPEG_2000_LOSSLESS_ONLY JPEG_2000 PRIVATE_SYNTAX_1 or PRIVATE_SYNTAX_2
UserInfo	Address of data which will be passed on to *YourReceiveStreamFunction* each time it is called. This may be

NULL.

YourReceiveStreamFunction	Name of a function which will be called repeatedly to provide blocks of streamed DICOM message data.

The function must be prototyped as follows:

```
MC_STATUS YourReceiveStreamFunction (
      int CBmessageID,
      void* CBuserInfo,
      int CBdataSize,
      void* CBdataBuffer,
      int CBisFirst,
      int CBisLast
)
```

CBmessageID	The identifier assigned to the message object by the **MC_Open_Message** function.
CBuserInfo	Address of data which is being passed from the **MC_Message_To_Stream** function. This may be NULL.
CBdataSize	The number of bytes of stream data being provided to you in *CBdataBuffer*.
CBdataBuffer	The buffer containing stream data from the message object.
CBisFirst	Is TRUE (not zero) when MergeCOM-3 is providing the first block of stream data.
CBisLast	Is TRUE (not zero) when MergeCOM-3 is providing the last block of stream data.

Remarks

MC_Message_To_Stream requests that the contents of the message identified by *MessageID* be "streamed" (i.e. put in the form defined by the DICOM standard). The streamed message is passed to the user in blocks by calling ***YourReceiveStreamFunction*** repeatedly until the entire streamed message has been transferred.

MC_Message_To_Stream can pass data to ***YourReceiveStreamFunction*** by specifying the data's address in *UserInfo*. MergeCOM-3 passes the address to ***YourReceiveStreamFunction*** in *CBuserInfo* each time it is called. *UserInfo* may be NULL.

StartTag and *StopTag* specify which attributes in the message are to be placed in the stream. Any attributes in the message with tags less than *StartTag* or greater than *StopTag* will be ignored. Neither *StartTag* nor *StopTag* need be in the message.

SyntaxType must be set to **one of the values listed above.** The transfer syntax specifies the byte order used in the streamed message, whether or not each attribute's value representation is explicitly encoded in the stream, and how the pixel data is encoded in the message.

NOTE: If the message contains "group length" attributes (i.e. attributes with tags of the form gggg0000: any group, element zero), **MC_Message_To_Stream** will automatically calculate the group length value when the message is streamed.

YourReceiveStreamFunction

YourReceiveStreamFunction will be called repeatedly to pass blocks of data to it. MergeCOM-3 sets **CBisFirst* to TRUE(non-zero) the first time it calls

YourReceiveStreamFunction for this attribute and it sets **CBisLast* to TRUE(non-zero) when it calls ***YourReceiveStreamFunction*** with the final block of streamed data.

**CBdataBuffer* is set to the address of a buffer containing the stream data block, and **CBdataSize* is be set to the number of bytes placed at **CBdataBuffer*.

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	One or more of the pointer-type parameters was NULL.
MC_INVALID_MESSAGE_ID	The *MessageID* value is not a valid message object ID.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.
MC_MESSAGE_EMPTY	The message has no attributes in it.
MC_CALLBACK_CANNOT_COMPLY	***YourReceiveStreamFunction*** returned a value other than MC_NORMAL_COMPLETION.
MC_INVALID_TRANSFER_SYNTAX	An invalid code was used for the *SyntaxType* parameter.
MC_TEMPFILE_ERROR	A value for an attribute is stored temporarily on file and an I/O error occurred.

See Also

MC_Stream_To_Message
MC_Get_Stream_Length

MC_NewProposedServiceList
MC_NewProposedServiceListAsync

Creates a new service list for use in association negotiation.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_NewProposedServiceList(
      char *ServiceListName,
      char *ServiceNameArray[])

MC_STATUS MC_NewProposedServiceListAsync(
      char *ServiceListName,
      char *ServiceNameArray[]
      unsigned short MaxOperationsInvoked,
      unsigned short MaxOperationsPerformed)
```

ServiceListName	Application supplied name to associate with this list
ServiceNameArray	NULL terminated array of Service Names to be used in this list
MaxOperationsInvoked	The maximum operations invoked by the association requestor. Note that a setting of 0 means an unlimited number of operations can be invoked
MaxOperationsPerformed	The maximum operations performed by the association reqestor. Note that a setting of 0 means an unlimited number of operations can be performed

Remarks

MC_NewProposedServiceList and **MC_NewProposedServiceListAsync** establish a list of services to be used during the negotiation of an association. The ServiceListName can be used in **MC_Open_Association, MC_Open_Secure_Association** and **MC_Wait_For_Association**.

This functionality is used to dynamically create list normally found in the **mergecom.app** configuration file. This method of service list generation augments the existing configuration file.

Service names are generated using **MC_NewServiceFromName** or **MC_NewServiceFromUID**.

MC_NewProposedServiceListAsync allows setting of a value for Max Operations Performed and Max Operstions Invoked during association negotation. These settings allow negotiation of the DICOM Asynchronous Operations Window. The negotiated results for the association requestor can be examined by calling **MC_Get_Association_Info**.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating

	environment. A message describing the error has been written to the MergeCOM-3 log file.
MC_DUPLICATE_NAME	A Duplicate Name is present in the SeviceNameArray or the request service list name is already in use.

See Also

MC_Open_Association
MC_Open_Secure_Association
MC_Wait_For_Association
MC_Wait_For_Secure_Association
MC_NewServiceFromName
MC_NewServiceFromUID
MC_FreeServiceList
MC_Get_Association_Info

MC_NewSyntaxList

Creates a new syntax list for use in the creation of services for association negotiation.

Synopsis

```
#include "mergecom.h"
#include "mc3msg.h"

MC_STATUS MC_NewSyntaxList(
      char *SyntaxListName,
      TRANSFER_SYNTAX SyntaxArray[])
```

SyntaxListName	Application supplied name to associate with this list
SyntaxArray	NULL terminated array of TRANSFER_SYNTAXs to be used in this list

Remarks

MC_NewSyntaxList establishes a list of syntaxes to be used during the negotiation of an association. The SyntaxList name is supplied by the application and is used as a reference when creating service references

This functionality is used to dynamically create list normally found in the **mergecom.app** configuration file. This method of syntax list generation augments the existing configuration file. TRANSFER_SYNTAXs are an enumerated type found in **mc3msg.h**

Service names are generated using **MC_NewServiceFromName or MC_NewServiceFromUID**.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.
MC_DUPLICATE_NAME	The requested syntax list name is already in use.
MC_DUPLICATE_SYNTAX	A syntax id is duplicated in the array supplied.

See Also

MC_Open_Association
MC_Open_Secure_Association
MC_Wait_For_Association
MC_Wait_For_Secure_Association
MC_NewServiceFromName
MC_NewServiceFromUID
MC_FreeSyntaxList

MC_NewService… Functions

Creates a new service description for use in the creation of service lists.

Synopsis

```
#include "mergecom.h"
MC_STATUS MC_NewServiceFromName(
      char *ServiceName,
      char *SOPClassName,
      char *SyntaxListName,
      int SCURole,
      int SCPRole)
MC_STATUS MC_NewServiceFromUID (
      char *ServiceName,
      char *SOPClassUID,
      char *SyntaxListName,
      int SCURole,
      int SCPRole)
MC_STATUS MC_NewServiceWithExtInfoFromName(
      char *ServiceName,
      char *SOPClassName,
      char *SyntaxListName,
      int SCURole,
      int SCPRole,
      void* ExtInfoBuffer,
      int ExtInfoLength)
MC_STATUS MC_NewServiceWithExtInfoFromUID (
      char *ServiceName,
      char *SOPClassUID,
      char *SyntaxListName,
      int SCURole,
      int SCPRole,
      void* ExtInfoBuffer,
      int ExtInfoLength)
```

ServiceName	Application supplied name to associate with this service
SOPClassName	Name as defined in **mergecom.srv**
SOPClassUID	UID as defined by DICOM and found in **mergecom.srv**
SyntaxListName	Name of a syntax list as defined by **MC_NewSyntaxList** or in the **mergecom.app.** Can be NULL, if so the default syntax lists will be utilized.
SCURole	Sets the scu role negotiation parameters for this service, 0 or 1
SCPRole	Sets the scp role negotiation parameters for this service, 0 or 1
ExtInfoBuffer	A pointer to the buffer containing extended negotiation information.
ExtInfoLength	The number of bytes contained in the *ExtInfoBuffer*.

Remarks

These routines are utilized to create a service with a specific name to be used in a service list. Each service utilitized in a dynamic service list must be created with a call to one of these functions. The service lists created can then be used in subsequent calls to **MC_Open_Association, MC_Open_Secure_Association**,

MC_Wait_For_Association, or **MC_Wait_For_Secure_Association** to specify the services negotiated for an association.

When creating a new service, it can be identified by its MergeCOM-3 service name specified in the **mergecom.srv** file or by its SOP Class UID depending on which call is utilized. The SCU and SCP roles are specified for the service along with the name of a syntax list to be used. The **MC_NewServiceWithExtInfoFromUID** and **MC_NewServiceWithExtInfoFromName** routines can be used to optionally specify extended negotiatoin information for the service.

The application assigned name is utilized by **MC_NewProposedServiceList** .

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.
MC_INVALID_SOP_CLASS_UID	The SOPClassName or SOPClassUID do not reference a defined SOP in **mergecom.srv**

See Also

MC_NewSyntaxList **MC_NewProposedServiceList**
MC_FreeService
MC_Set_Negotiation_Information_For_Association

MC_Open_Association

Establishes a connection with a remote DICOM application.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Open_Association (
      int ApplicationID,
      int* AssociationID,
      const char* RemoteApplicationTitle,
      int* RemoteHostPortNumber,
      char* RemoteHostTCPIPName,
      char* ServiceList
)
```

ApplicationID	The application ID returned from the MC_Register_Application function.
AssociationID	The identification number of an association object is returned here.
RemoteApplicationTitle	The DICOM Application Title of the remote application.

Each of the following is optional. Use NULL if not used

RemoteHostPortNumber	The TCP/IP port used by the remote application to "listen" for DICOM associations
RemoteHostTCPIPName	The remote host's TCP/IP Name
ServiceList	Name of a service list in the MergeCOM-3 configuration file.

Remarks

MC_Open_Association establishes a DICOM association connection with a remote DICOM application.

Each application in a DICOM association has a publicly known name or "application title". This application's title was declared in the **MC_Register_Application** call. The application to which we are intending to connect is specified in *RemoteApplicationTitle*. The remote DICOM system waits for association requests on a given TCP/IP port. That port number is specified by *RemoteHostPortNumber*. The TCP/IP name of the remote host is specified by *RemoteHostTCPIPName*.

Starting a DICOM association is a negotiated process. One application provides one or more services, and the other uses one or more of the services provided. The **MC_Open_Association** function lets the remote DICOM process know which services this application wishes to use. This "service list" is defined in the MergeCOM-3 Application Configuration file, and the name of the appropriate service list is specified by the *ServiceList* parameter.

A few DICOM services require the applications to negotiate application-specific information. This is performed by sharing "extended negotiation information" when the association is negotiated. If one or more of the services in *ServiceList* requires such negotiation, **MC_Set_Negotiation_Info** should be called for each such service to supply the negotiation information.

If the remote system accepts the association request (i.e. it can support one or more of the services in the *ServiceList*), **MC_Open_Association** creates an association object, puts its ID in **AssociationID*, and returns MC_NORMAL_COMPLETION.

If the service requires the remote application to return extended negotiation information, **MC_Get_Negotiation_Info** should be called to retrieve such information. If the information is unacceptable, **MC_Close_Association** should be called.

NOTE: That the information registered with the **MC_Set_Negotiation_Info** call is not changed during the association process. Use **MC_Clear_Negotiation_Info** to prevent the information from being used in subsequent association negotiations for a given service.

MC_ASSOCIATION_REJECTED is returned if the remote system rejects the association request. **AssociationID* is NOT valid if the association is rejected or if an error occurs.

NOTE: Only the first three parameters (*ApplicationID, AssociationID and RemoteApplicationTitle*) are required. Each of the others will be defaulted if they are specified as NULL. The default values are obtained from the *RemoteApplicationTitle*'s section in the MergeCOM-3 Application Configuration file.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally. *AssociationID* has been set to the association's identification number.
MC_ASSOCIATION_REJECTED	The remote system rejected the open association request.
MC_NEGOTIATION_ABORTED	The association was aborted during negotiation
MC_CONNECTION_FAILED	The remote system could not be connected to at the TCP/IP level. Check that the remote host name and port number have been configured properly.
MC_NULL_POINTER_PARM	*AssociationID* and/or *RemoteApplicationTitle* was NULL.
MC_INVALID_APPLICATION_TITLE	*RemoteApplicationTitle* was not 1-16 bytes long.
MC_INVALID_APPLICATION_ID	*ApplicationID* does not identify a valid MergeCOM-3 application.
MC_INVALID_PORT_NUMBER	*RemoteHostPortNumber* was not NULL and pointed at a negative number.
MC_INVALID_SERVICE_LIST_NAME	*ServiceList* was not NULL and pointed at an empty string.
MC_INVALID_HOST_NAME	*RemoteHostTCPIPName* was not NULL and it was not 1-39 bytes long.
MC_UNKNOWN_HOST_NAME	The *RemoteHostTCPIPName* is unknown to the system.

MC_TIMEOUT	Timeout attempting the association.
MC_SYSTEM_CALL_INTERRUPTED	The operating system interrupted the network call. Retry the connection.
MC_NO_APPLICATIONS_REGISTERED	An application title for this application has not yet be registered.
MC_LIBRARY_NOT_INITIALIZED	This call was made prior to the initialization of the library
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Register_Application
MC_Open_Secure_Association
MC_Wait_For_Association
MC_Wait_For_Secure_Association
MC_Set_Negotiation_Info
MC_Get_Negotiation_Info
MC_Clear_Negotiation_Info
MC_Close_Association
MC_Abort_Association

MC_Open_Empty_Message

Creates a new "empty" MergeCOM-3 message object.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Open_Empty_Message (
      int* MessageIDPtr
)
```

MessageIDPtr Upon successful completion, the message object identifier is returned here.

Remarks

The **MC_Open_Empty_Message** function creates a "message object" which contains no attributes. The resulting message object is given an identification number which is returned in **MessageIDPtr*. All functions dealing with this message must provide this message ID number.

The opened message object is not associated with any particular DICOM service or command. If the message object is to be used to send a message to a network partner, or if **MC_Validate_Message** is to be called for the message object, **MC_Set_Service_Command** must be called first to associate the message object with a given DICOM service and command.

If a message is opened using **MC_Open_Empty_Message**, it is not necessary to add attributes to the message object before setting attribute values. If one of the set value functions (e.g. **MC_Set_Value_From_String**) is used for an attribute, the attribute will automatically be added to the message object before the value is set. (Note that this is NOT THE CASE if a message object is opened using **MC_Open_Message**. In that case the message IS associated with a given service/command pair and attributes other than those associated with that service and command must be explicitly added to the message before setting values for the added attributes.)

Reference the "DICOM V3.0 Standard, Final Text - October 29, 1993" for more information about these commands.

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	*MessageIDPtr* was NULL.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Open_Message
MC_Free_Message
MC_Free_Item

MC_Open_File
MC_Open_File_Bypass_OBOW
MC_Open_File_Upto_Tag

Requests that the values of a file object be retrieved from a DICOM file

Synopsis

```
#include "mc3media.h"

MC_STATUS MC_Open_File (
      int ApplicationID,
      int FileID,
      void* UserInfo,
      MC_STATUS (*YourFromMediaFunction() )
)

MC_STATUS MC_Open_File_Bypass_OBOW (
      int ApplicationID,
      int FileID,
      void* UserInfo,
      MC_STATUS (*YourFromMediaFunction() )
)

MC_STATUS MC_Open_File_Upto_Tag (
      int ApplicationID,
      int* FileID,
      void* UserInfo,
      unsigned long Tag,
      long* Offset,
      MC_STATUS (*YourFromMediaFunction() )
)
```

ApplicationID	The identifier assigned to this object by the **MC_Register_Application** function.
FileID	The identifier assigned to this object by the **MC_Create_File** or **MC_Create_Empty_File** functions.
UserInfo	Address of data which will be passed on to *YourFromMediaFunction* each time it is called. This may be NULL.
Tag	The attribute after which to stop reading the file from media.
Offset	Upon successful completion, the offset in bytes from the beginning of the file where the first attribute greater than *Tag* is returned here.
YourFromMediaFunction	Name of a function which will be called repeatedly to request blocks of DICOM file data from media.

The function must be prototyped as follows:

```
MC_STATUS YourFromMediaFunction (
      char* CBfilename,
      void* CBuserInfo,
      int* CBdataSizePtr,
      void** CBdataBufferPtr,
      int CBisFirst,
```

```
      int* CBisLastPtr
)
```

CBfilename	String filename associated with file object
CBuserInfo	Address of data which is being passed from the **MC_Open_File** function. This may be NULL.
CBdataSizePtr	Set **CBdataSizePtr* to the number of bytes you are providing. This must be an even number.
CBdataBufferPtr	Set **CBdataBufferPtr* to the address of the data you are providing.
CBisFirst	This is TRUE (non-zero) the first time MergeCOM-3 calls *YourFromMediaFunction* to request data blocks.
CBisLastPtr	Set **CBisLastPtr* to TRUE (non-zero) when you are returning with the last block of file data.

Remarks

MC_Open_File, **MC_Open_File_Bypass_OBOW**, and **MC_Open_File_Upto_Tag** request that the contents of a DICOM file be converted and placed into the file object *FileID*. The file is passed to MergeCOM-3 by ***YourFromMediaFunction***. MergeCOM-3 repeatedly calls ***YourFromMediaFunction*** until all of the file has been processed.

The MC_Open_File… functions can pass data to ***YourFromMediaFunction*** by specifying the data's address in *UserInfo*. MergeCOM-3 passes the address to ***YourFromMediaFunction*** in *CBuserInfo* each time it is called. *UserInfo* may be NULL.

When **MC_Open_File** retrieves attributes of type OB, OW, or OF and a callback is not registered for the attribute, the library stores the values in a configurable location (normally in temporary files). Then the user can use **MC_Get_Value_To_Function** to retrieve the attribute values. If a callback is registered for the attribute, the callback is supplied the data as it is read. See the description of **MC_Register_Callback_Function** for more details.

When **MC_Open_File_Bypass_OBOW** retrieves attributes of type OB, OW, or OF and a callback is not registered for the attribute, the library stores the values in a configurable location (normally in temporary files). Then the user can use **MC_Get_Value_To_Function** to retrieve the attribute values.

Performance Tuning:

MC_Open_File_Bypass_OBOW can be used to increase performance for handling attributes of type OB, OW, or OF. When a callback function is registered with **MC_Register_Callback_Function** for an attribute of type OB, OW, or OF, **MC_Open_File_Bypass_OBOW** will not read in the attribute's value. Instead, the data will be left on media and the offset of the attribute's value from the beginning of the file along with length of the value will be passed to the user's callback function. When the data is needed by the user or MergeCOM-3, the callback can retrieve it from media. See the description of **MC_Register_Callback_Function** for more details.

When **MC_Open_File_Upto_Tag** retrieves attributes of type OB, OW, or OF and a callback is not registered for the attribute, the library stores the values in a configurable location (normally in temporary files). Then the user can use **MC_Get_Value_To_Function** to retrieve the attribute values.

Performance Tuning:

MC_Open_File_Upto_Tag can be used to increase performance for handling attributes of type OB, OW, or OF. **MC_Open_File_Upto_Tag** will stop reading the file from media when it reaches the first attribute greater than or equal to *Tag*. As a convenience, the offset in bytes from the beginning of the file of the first attribute greater than *Tag* is returned in *Offset*. If the user wants to use *Offset* to access attributes greater then *Tag* directly from media, the user will have to parse the file directly.

If the file contains any attributes with a value representation of SQ (i.e. the file contains one or more sequence of items), an item object is automatically opened for each item in the file. The ItemID associated with each opened item object is used as the value for each item in the sequence attribute. Later, the **MC_Get_Value...** functions may be used to retrieve the ItemID's from the SQ attribute. Then, again using the **MC_Get_Value...** functions, the attributes of the ItemID object may be retrieved.

The MC_Open_File… functions also will read in DICOMDIRs. In order for a DICOMDIR to be properly read in, the file object must be associated with the service for a DICOMDIR. This can be accomplished by specifying the service when the object is created, or calling **MC_Set_Service_Command** on the object before one of the **MC_Open_File…** functions is called. When one of these conditions are met, the function will resolve the directory record file offset pointers within the DICOMDIR by assigning them the ItemID associated with each opened item object.

The **MC_Open_File…** functions read in the DICOM File Meta Information attributes in explicit VR little endian transfer syntax. The remainder of the file is read in with the transfer syntax specified in the attribute (0002,0010). If this attribute is not found, the whole file is read in using the explicit VR little endian transfer syntax.

NOTE: A runtime configuration parameter determines what will happen if the input file contains an invalid DICOM file prefix. The default is to change any invalid prefix to "DICM". If requested, however, an invalid prefix can be left in the file. If an invalid prefix is read in or wrote out a warning message will be logged.

NOTE: A runtime configuration parameter determines what will happen if the input file contains an attribute which is not in the service associated with the file. The default is to ignore such attributes (with a warning message logged). If requested, however, such attributes will be added to the file, along with their values. If the Value Representation of the attribute being added cannot be determined, the attribute is given a pseudo Value Representation of "**Unknown_VR**". The only way to retrieve the value of such attributes is to use the **MC_Get_Value_To_Buffer** function. To change the value of such attributes, **MC_Set_Value_Representation (**or **MC_Set_pValue_Representation**) must first be called to assign a valid Value Representation to the attribute. If **MC_Write_File** is used, attributes with unknown VR's are simply copied (memcpy) to the stream with no consideration given to byte ordering.

YourFileToFunction

It is the responsibility of ***YourFromMediaFunction*** to pass blocks of data back to MergeCOM-3 each time it is called. MergeCOM-3 sets *CBFirstCall* to TRUE(non-zero) the first time it calls ***YourFromMediaFunction*** for this file.

**CBdataBufferPtr* must be set to the address of the buffer containing the stream data block, and **CBdataSizePtr* must be set to the number of bytes placed at **CBdataBufferPtr*.

YourFromMediaFunction must set **CBisLast* to TRUE(non-zero) when it is providing the last block of the streamed message.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.

MC_NULL_POINTER_PARM	*FileIDPtr*, ***YourFromMediaFunction***, or *Filename* was NULL.
MC_INVALID_APPLICATION_ID	The application ID does not identify a valid MergeCOM-3 application.
MC_INVALID_FILE	An invalid DICOM Prefix was found within the file. A message describing the error has been written to the MergeCOM-3 log file.
MC_INVALID_TRANSFER_SYNTAX	An invalid transfer syntax code was found within the file's DICOM File Meta Information.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.
MC_CALLBACK_CANNOT_COMPLY	***YourFromMediaFunction*** returned with a status other than MC_NORMAL_COMPLETION.
MC_CALLBACK_DATA_SIZE_UNEVEN	The *CBdataSizePtr* parameter returned by ***YourFromMediaFunction*** was an uneven number.
MC_CALLBACK_PARM_ERROR	A callback function registered by your application returned an empty (NULL) data buffer when the buffers length was specified as being non-zero. See **MC_Register_Callback_Function** for details.
MC_OUT_OF_ORDER_TAG	A tag was found in the file that was not in ascending order. A message describing the error has been written to the MergeCOM-3 log file.
MC_INVALID_TAG	The file contains an invalid tag.
MC_VALUE_TOO_LARGE	An attribute in the file (other than those with value representations of OB, OW, or OF) was larger than that acceptable for its value representation.
MC_UNEXPECTED_EOD	***YourFromMediaFunction*** stopped without passing the entire value for an attribute.
MC_INVALID_LENGTH_FOR_VR	The value(s) for one of the file attributes was not legal for its value representation.

Also any of the status codes which may be returned by the **MC_Set_Value** call may also be returned.

See Also

MC_Register_Callback_Function **MC_Write_File**
MC_Write_File_By_Callback

MC_Open_Item

Creates a new MergeCOM-3 item object.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Open_Item (
      int* ItemIDPtr,
      char* ItemName
)
```

ItemIDPtr	Upon successful completion, the item object identifier is returned here.
ItemName	String name of the item to be associated with this item object.

Remarks

The **MC_Open_Item** function creates a "item object" which contains (or will contain) all of the attributes of a named item which will be used in a sequence of items in a message object or in another item object. The resulting item object is given an identification number which is returned in **ItemIDPtr*. All functions dealing with this item must provide this item ID number.

The *ItemName* is used to access configuration information which describes the attributes of the message. If such configuration information is not available, an empty item object is created, and a warning message is logged.

Return Value

One of these enumerated **MC_STATUS codes define in "mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	One or more of the pointer-type parameters was NULL.
MC_MSGFILE_ERROR	An error occurred attempting to access the configuration data file.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Open_Message

MC_Open_Message

Creates a new MergeCOM-3 message object.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Open_Message (
      int* MessageIDPtr,
      char* ServiceName,
      MC_COMMAND Command
)
```

MessageIDPtr	Upon successful completion, the message object identifier is returned here.
ServiceName	String name of a DICOM service to be associated with this message object.
Command	The command which is to be associated with this message. The **MC_COMMAND** enumerated values are defined in "mc3msg.h".

Remarks

The **MC_Open_Message** function creates a "message object" which contains (or will contain) all of the attributes of a DICOM message which will be used for the given *ServiceName* and *Command*. The resulting message object is given an identification number which is returned in **MessageIDPtr*. All functions dealing with this message must provide this message ID number.

The *ServiceName* and *Command* are used to access configuration information which describes the attributes of the message. If such configuration information is not available, an empty message object is created, and a warning message is logged.

MC_Open_Message generates in each message object created the set of "command type" attributes used by most DICOM services. While MergeCOM-3 sets the values of many of these "command type" attributes automatically, some services require the application to set them. (Refer to the description of **MC_Send_Request_Message** for more information.)

MC_Open_Empty_Message should be used if the service and command are not yet known, or if there is no need to validate that values will be set only for attributes assigned to a given service/command pair.

The following commands are supported by the Advanced MergeCOM-3 Tool Kit:

Command	Description
C_STORE_RQ	DICOM Composite Store Service Request
C_STORE_RSP	DICOM Composite Store Service Response
C_ECHO_RQ	DICOM Verification Service Request
C_ECHO_RSP	DICOM Verification Service Response
C_FIND_RQ	DICOM Composite Find Service Request
C_FIND_RSP	DICOM Composite Find Service Response
C_CANCEL_FIND_RQ	Cancel DICOM Composite Find Service Request

C_GET_RQ	DICOM Composite Get Service Request
C_GET_RSP	DICOM Composite Get Service Response
C_CANCEL_GET_RQ	Cancel DICOM Composite Get Service Request
C_MOVE_RQ	DICOM Composite Move Service Request
C_MOVE_RSP	DICOM Composite Move Service Response
C_CANCEL_MOVE_RQ	Cancel DICOM Composite Move Service Request
N_EVENT_REPORT_RQ	DICOM Normalized Report Service Request
N_EVENT_REPORT_RSP	DICOM Normalized Report Service Response
N_GET_RQ	DICOM Normalized Get Service Request
N_GET_RSP	DICOM Normalized Get Service Request
N_SET_RQ	DICOM Normalized Set Service Request
N_SET_RSP	DICOM Normalized Set Service Response
N_ACTION_RQ	DICOM Normalized Action Service Request
N_ACTION_RSP	DICOM Normalized Action Service Response
N_CREATE_RQ	DICOM Normalized Create Service Request
N_CREATE_RSP	DICOM Normalized Create Service Response
N_DELETE_RQ	DICOM Normalized Delete Service Request
N_DELETE_RSP	DICOM Normalized Delete Service Response

Reference the “DICOM V3.0 Standard, Final Text - October 29, 1993” for more information about these commands.

Return Value

One of these enumerated **MC_STATUS** codes defined in **“mcstatus.h”**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	One or more of the pointer-type parameters was NULL.
MC_INVALID_COMMAND	*Command* is not a supported command.
MC_MSGFILE_ERROR	An error occurred attempting to access the configuration data file.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Open_Empty_Message
MC_Free_Message
MC_Free_Item

MC_Open_Secure_Association

Establishes a connection with a remote DICOM application over a secure socket connection.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Open_Secure_Association (
      int ApplicationID,
      int* AssociationID,
      const char* RemoteApplicationTitle,
      int* RemoteHostPortNumber,
      char* RemoteHostTCPIPName,
      char* ServiceList,
      SecureSocketFunctions* SS_functions,
      void* SS_application_context
)
```

ApplicationID	The application ID returned from the MC_Register_Application function.
AssociationID	The identification number of an association object is returned here.
RemoteApplicationTitle	The DICOM Application Title of the remote application.
SS_functions	Pointer to a structure containing functions that will be called by MergeCOM-3 while processing network I/O over the secure connection. (see below)

Each of the following is optional. Use NULL if not used

RemoteHostPortNumber	The TCP/IP port used by the remote application to "listen" for DICOM associations
RemoteHostTCPIPName	The remote host's TCP/IP Name
ServiceList	Name of a service list in the MergeCOM-3 configuration file.
SS_context	An optional pointer to application-specific data that MergeCOM-3 passes to the functions declared in *SS_functions*.

Remarks

MC_Open_Secure_Association establishes a secure DICOM association connection with a remote DICOM application. Use **MC_Open_Association** if a secure network connection is not required.

Each application in a DICOM association has a publicly known name or "application title". This application's title was declared in the **MC_Register_Application** call. The application to which we are intending to connect is specified in *RemoteApplicationTitle*. The remote DICOM system waits for association requests on a given TCP/IP port. That port number is specified by *RemoteHostPortNumber*. The TCP/IP name of the remote host is specified by *RemoteHostTCPIPName*.

Starting a DICOM association is a negotiated process. One application provides one or more services, and the other uses one or more of the services provided. The **MC_Open_Secure_Association** function lets the remote DICOM process know which services this application wishes to use. This "service list" is defined in the MergeCOM-3

Application Configuration file, and the name of the appropriate service list is specified by the *ServiceList* parameter.

A few DICOM services require the applications to negotiate application-specific information. This is performed by sharing "extended negotiation information" when the association is negotiated. If one or more of the services in *ServiceList* requires such negotiation, **MC_Set_Negotiation_Info** should be called for each such service to supply the negotiation information.

If the remote system accepts the association request (i.e. it can support one or more of the services in the *ServiceList*), **MC_Open_Secure_Association** creates an association object, puts its ID in **AssociationID*, and returns MC_NORMAL_COMPLETION.

If the service requires the remote application to return extended negotiation information, **MC_Get_Negotiation_Info** should be called to retrieve such information. If the information is unacceptable, **MC_Close_Association** should be called.

NOTE: That the information registered with the **MC_Set_Negotiation_Info** call is not changed during the association process. Use **MC_Clear_Negotiation_Info** to prevent the information from being used in subsequent association negotiations for a given service.

MC_ASSOCIATION_REJECTED is returned if the remote system rejects the association request. **AssociationID* is NOT valid if the association is rejected or if an error occurs.

NOTE: Only four parameters (*ApplicationID, AssociationID, RemoteApplicationTitle and SS_functions*) are required. *RemoteHostPortNumber*, *RemoteHostTCPIPName* and *ServiceList* will be defaulted if they are specified as NULL. The default values are obtained from the *RemoteApplicationTitle*'s section in the MergeCOM-3 Application Configuration file.

SecureSocketFunctions

When using the **MC_Wait_For_Secure_Association** call, MergeCOM-3 establishes a TCP/IP connection and then calls the functions provided by the *SS_functions* parameter to establish the secure connection and to pass data through the secure connection. The *SS_functions* are responsible for sending and receiving all data through the secure connection, thus allowing them to do so using a secure protocol such as Secure Socket Layer(SSL). MergeCOM-3 closes the underlying TCP/IP connection when all association processing has completed and after it calls the **SS_Session_Shutdown** callback.

The **SecureSocketFunctions** structure is declared in mergecom.h as follows:

```
typedef struct MC_Secure_Socket_Functions_Struct {
      SS_STATUS (NOEXP_FUNC *SS_Session_Start)
                (int          SocketToUse,
                CONN_TYPE     ConnectionType,
                void*         ApplicationContext,
                void**        SecurityContext);
      SS_STATUS (NOEXP_FUNC *SS_Read)
                void*         SScontext,
                void*         ApplicationContext,
                char*         Buffer,
                int           BytesToRead,
                void*         BytesRead,
                int           Timeout);
```

```
        SS_STATUS (NOEXP_FUNC *SS_Write)
                  (void*        SScontext,
                  void*         ApplicationContext,
                  char*         Buffer,
                  int           BytesToWrite,
                  int*          BytesWritten,
                  int           Timeout);
        void (NOEXP_FUNC *SS_Session_Shutdown)
                  (void*        SScontext,
                  void*         ApplicationContext);
} SecureSocketFunctions;
```

You must provide valid function pointers for each of the four fields in the **SecureSocketFunctions** structure.

SS_Session_Start

MergeCOM-3 calls the **SS_Session_Start** function just after it has established a TCP/IP connection with the remote host. The *SocketToUse* parameter contains the socket assigned to the connection. Please note that the connection is non-blocking. The *ApplicationContext* parameter is the presented by the *SS_context* parameter of the **MC_Open_Secure_Association** call. *ConnectionType* will be the manifest constant REQUESTER_CONNECTION if the **SS_Session_Start** function is called as a result of a **MC_Open_Secure_Association** call. (It will be ACCEPTOR_CONNECTION if it is called as a result of a **MC_Wait_For_Secure_Association** call.)

The **SS_Session_Start** function is responsible for establishing a secure connection using the socket provided. It is assumed, but not required, that the connection will be a Secure Socket Layer (SSL) or Transport Layer Socket (TLS) connection. A pointer to a context block should be returned at **SecurityContext* if the secure connection is established. MergeCOM-3 will provide this pointer when it calls the other callback functions.

SS_Session_Start must return SS_NORMAL_COMPLETION if the secure connection was successfully established, otherwise SS_ERROR must be returned. If it returns SS_ERROR, the TCP/IP connection will be closed and the **MC_Open_Secure_-Association** call will return a status of **MC_NEGOTIATION_ABORTED**.

SS_Session_Shutdown

MergeCOM-3 calls the **SS_Session_Shutdown** function when the association is aborted or closed. It is the responsibility of the **SS_Session_Shutdown** function to gracefully close the secure network connection. MergeCOM-3 closes the TCP/IP socket connection after calling **SS_Session_Shutdown**. The *ApplicationContext* parameter is that presented by the *SS_context* parameter of the **MC_Open_Secure_Association** call, and the *SScontext* parameter is that previously returned by the **SS_Session_Start** function.

SS_Read

MergeCOM-3 calls the **SS_Read** function whenever it needs association data from the secure connection. The *ApplicationContext* parameter is that presented by the *SS_context* parameter of the **MC_Open_Secure_Association** call, and the *SScontext* parameter is that previously returned by the **SS_Session_Start** function. It is the responsibility of the **SS_Read** function to retrieve into *Buffer* the number of unencrypted data bytes specified by *BytesToRead*. The actual number of bytes placed in the *Buffer* is returned at **BytesRead*. SS_NORMAL_COMPLETION must be returned if the read request was satisfied.

If the **SS_Read** function cannot retrieve *BytesToRead* bytes in *Timeout* seconds, it must return SS_TIMEOUT. Please note that the socket connection passed by MergeCOM-3 to

the **SS_Session_Start** function is non-blocking. (Note that if SS_TIMEOUT is returned, an outstanding MC_Read_Message call will return MC_TIMEOUT.)

If it is determined that the secure socket connection has closed, or that the underlying transport has closed, SS_SESSION_CLOSED must be returned. This should only occur during a DICOM association if the remote host aborted the association. If a fatal error occurs while processing the read request, SS_ERROR must be returned. (Note that if SS_SESSION_CLOSED or SS_ERROR is returned, an outstanding MC_Read_Message call will return MC_NETWORK_SHUT_DOWN.)

SS_Write

MergeCOM-3 calls the **SS_Write** function whenever it needs to send association data over the secure connection. The *ApplicationContext* parameter is that presented by the *SS_context* parameter of the **MC_Open_Secure_Association** call, and the *SScontext* parameter is that previously returned by the **SS_Session_Start** function. It is the responsibility of the **SS_Write** function to send *BytesToWrite* bytes of the data in *Buffer* over the secure connection, returning the number of bytes actually written at **BytesWritten*. SS_NORMAL_COMPLETION must be returned if the write request was satisfied.

If the **SS_ Write** function cannot send *BytesToWrite* bytes in *Timeout* seconds, it must return SS_TIMEOUT. Please note that the socket connection passed by MergeCOM-3 to the **SS_Session_Start** function is non-blocking.

If a fatal error occurs while processing the write request, SS_ERROR must be returned. (Note that if SS_ERROR is returned, an outstanding MC_Send_Request_Message or MC_Send_Response_Message call will return MC_SYSTEM_ERROR.)

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally. *AssociationID* has been set to the association's identification number.
MC_ASSOCIATION_REJECTED	The remote system rejected the open association request.
MC_NEGOTIATION_ABORTED	The association was aborted during negotiation
MC_CONNECTION_FAILED	The remote system could not be connected to at the TCP/IP level. Check that the remote host name and port number have been configured properly.
MC_NULL_POINTER_PARM	*AssociationID*, *SS_functions* or *RemoteApplicationTitle* was NULL.
MC_NULL_VALUE	One or more of the function parameters within *SS_functions* was NULL.
MC_INVALID_APPLICATION_TITLE	*RemoteApplicationTitle* was not 1-16 bytes long.
MC_INVALID_APPLICATION_ID	*ApplicationID* does not identify a valid MergeCOM-3 application.

MC_INVALID_PORT_NUMBER	*RemoteHostPortNumber* was not NULL and pointed at a negative number.
MC_INVALID_SERVICE_LIST_NAME	*ServiceList* was not NULL and pointed at an empty string.
MC_INVALID_HOST_NAME	*RemoteHostTCPIPName* was not NULL and it was not 1-39 bytes long.
MC_TIMEOUT	Timeout attempting the association.
MC_SYSTEM_CALL_INTERRUPTED	The operating system interrupted the network call. Retry the connection.
MC_NO_APPLICATIONS_REGISTERED	An application title for this application has not yet be registered.
MC_LIBRARY_NOT_INITIALIZED	This call was made prior to the initialization of the library
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Open_Association
MC_Register_Application
MC_Wait_For_Association
MC_Wait_For_Secure_Association
MC_Set_Negotiation_Info
MC_Get_Negotiation_Info
MC_Clear_Negotiation_Info
MC_Close_Association
MC_Abort_Association

MC_Parse_Association_Request (UNIX only)

Parses an association request that was received by the internet services daemon (inetd).

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Parse_Association_Request (
      const char* ServiceListName,
      int* ApplicationID,
      int* AssociationID,
)
```

ServiceListName	Name of a service list in the MergeCOM-3 configuration file.
ApplicationID	The identification number of a previously registered application is returned here.
AssociationID	The identification number of an association object is returned here.

Remarks

MC_Parse_Association_Request can be used on UNIX toolkit platforms in conjunction with a UNIX OS's internet services daemon (inetd). In these situations, inetd is configured to handle the TCP/IP connection request on a specific listen port. Once a connection request is received, inetd performs a fork/exec on a pre-configured user program to handle the connection.

Once the user's program is invoked, this function is used in the place of **MC_Wait_For_Association** to complete the association request with one of the previously registered applications.

inetd must be properly configured in order correctly perform the listen on the correct TCP/IP socket and to invoke the user's application. There are some differences between UNIX versions, so the following is only an example configuration. Please consult the appropriate UNIX man pages or documentation for more information.

The server must be specified in the systems services file (typically in /etc/inet/ or in /etc directory). E.g.

dicom_echo 104/tcp

The server must also be specified in the inetd configuration file (typically /etc/inet/inetd.conf or /etc/inetd.conf). E.g.:

dicom_echo stream tcp nowait root /usr/bin/inetd_echo_scp inetd_echo_scp /usr/bin/merge.ini

Where:

"dicom_echo" must be defined in services as servicing the DICOM port (e.g. 104)

"stream" - always

"tcp" - always

"nowait" - always

"root" - from /etc/passwd: typically root

"/usr/bin/inetd_echo_scp" - full path name to this runtime application

"inetd_echo_scp" - argv[0]

"/usr/bin/merge.ini" - argv[1] - The application's arguments, if any.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_TIMEOUT	The *Timeout* period expired.
MC_NEGOTIATION_ABORTED	An incoming association was aborted during negotiation. Normally this situation is handled by retrying the **MC_Parse_Association_Request** call.
MC_SYSTEM_CALL_INTERRUPTED	The system call used to wait for an association request was interrupted by a signal. Normally this situation is handled by retrying the **MC_Parse_Association_Request** call.
MC_NULL_POINTER_PARM	*AssociationID*, *ApplicationID* or *ServiceList* parameter was NULL.
MC_NO_APPLICATIONS_REGISTERED	No applications have been registered using **MC_Register_Application.**
MC_INVALID_SERVICE_LIST_NAME	*ServiceList* points at a null string.
MC_CONFIG_INFO_MISSING	Could not access *ServiceList* configuration parameters.
MC_CONFIG_INFO_ERROR	The *ServiceList* contained too many services.
MC_SYSTEM_ERROR	An unexpected, potentially serious problem was detected in the operating environment. A message describing the error had been written to the MergeCOM-3 log file.

See Also

MC_Wait_For_Association **MC_Wait_For_Secure_Association**

MC_Read_Message

Reads the next message to arrive from the remote application.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Read_Message (
      int AssociationID,
      int Timeout,
      int* MessageID,
      char** ServiceName,
      MC_COMMAND* Command
)
```

AssociationID	The association object's identification number
Timeout	The max time (in seconds) to wait for a message to start arriving. A value of zero(0) means "check one time" and a value of minus one (-1) means "wait forever".
MessageID	The ID of a message object containing the attributes of the message is returned here.
ServiceName	The name of the service associated with *MessageID* is returned here. Note well that this memory is freed when the **MC_Free_Message()** function is called for *MessageID*.
Command	The command which is associated with the message is returned here. The **MC_COMMAND** enumerated values are defined in "mc3msg.h".

Remarks

MC_Read_Message allows the caller to wait for the arrival of a request or response message from the remote application. If a message is not received on the association identified by *AssociationID* within *Timeout* seconds, the function returns a status of MC_TIMEOUT. Note that if *Timeout* is set to less than zero, **MC_Read_Message** will wait forever for a message to arrive. If it is set to zero, it will check one time if a message has arrived and then return.

Once a message begins arriving, the **INACTIVITY_TIMEOUT** configuration value is used. While receiving the message, the association will be aborted if there is no activity on the network for the length of time specified by this configuration value.

If a message arrives, **MessageID* is set to the ID of a message object which has been opened to receive the attributes of the message. The message's attribute values have been set by the DICOM message stream received from the remote application. If **MC_Register_Callback_Function** was called for any message attributes, those values were passed on to the registered callback function by the time this function (**MC_Read_Message**) returns.

ServiceName is set to the name of the service associated with *MessageID* and *Command* is set to the command associated with *MessageID.*

NOTE: It is the responsibility of the caller to release the message object. This is done using the **MC_Free_Message** call.

If **MC_Read_Message** returns either **MC_ASSOCIATION_CLOSED**, **MC_ASSOCIATION_ABORTED**, **MC_NETWORK_SHUT_DOWN, MC_INACTIVITY_TIMEOUT, MC_CONFIG_INFO_ERROR** or

MC_INVALID_MESSAGE_RECEIVED no further calls may be made for the association and no message ID is returned at **MessageID*.

The following commands are supported by the Advanced MergeCOM-3 Tool Kit:

Command	Description
C_STORE_RQ	DICOM Composite Store Service Request
C_STORE_RSP	DICOM Composite Store Service Response
C_ECHO_RQ	DICOM Verification Service Request
C_ECHO_RSP	DICOM Verification Service Response
C_FIND_RQ	DICOM Composite Find Service Request
C_FIND_RSP	DICOM Composite Find Service Response
C_CANCEL_FIND_RQ	Cancel DICOM Composite Find Service Request
C_GET_RQ	DICOM Composite Get Service Request
C_GET_RSP	DICOM Composite Get Service Response
C_CANCEL_GET_RQ	Cancel DICOM Composite Get Service Request
C_MOVE_RQ	DICOM Composite Move Service Request
C_MOVE_RSP	DICOM Composite Move Service Response
C_CANCEL_MOVE_RQ	Cancel DICOM Composite Move Service Request
N_EVENT_REPORT_RQ	DICOM Normalized Report Service Request
N_EVENT_REPORT_RSP	DICOM Normalized Report Service Response
N_GET_RQ	DICOM Normalized Get Service Request
N_GET_RSP	DICOM Normalized Get Service Request
N_SET_RQ	DICOM Normalized Set Service Request
N_SET_RSP	DICOM Normalized Set Service Response
N_ACTION_RQ	DICOM Normalized Action Service Request
N_ACTION_RSP	DICOM Normalized Action Service Response
N_CREATE_RQ	DICOM Normalized Create Service Request
N_CREATE_RSP	DICOM Normalized Create Service Response
N_DELETE_RQ	DICOM Normalized Delete Service Request
N_DELETE_RSP	DICOM Normalized Delete Service Response

Reference the “DICOM V3.0 Standard, Final Text - October 29, 1993” for more information about these commands.

Return Value

One of these enumerated **MC_STATUS** codes defined in **“mcstatus.h”**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_TIMEOUT	The *Timeout* period expired.

MC_MUST_CONTINUE_BEFORE_READING	A previous message was not read completely before making this call.
MC_NULL_POINTER_PARM	The *MessageID*, *ServiceName* or *Command* parameter was NULL.
MC_INVALID_ASSOC_ID	*AssociationID* is not a valid association object ID.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

The association is dropped if any of the following are returned:

MC_NETWORK_SHUT_DOWN	The network connect unexpectedly dropped.
MC_ASSOCIATION_ABORTED	The association has been aborted. A message describing the error has been written to the MergeCOM-3 log file.
MC_ASSOCIATION_CLOSED	The association has been closed.
MC_INACTIVITY_TIMEOUT	A timeout occurred in the middle of receiving a message.
MC_CONFIG_INFO_ERROR	The message information file describing the message's service/command pair could not be accessed.
MC_INVALID_MESSAGE_RECEIVED	An improperly formatted DICOM message was received. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Read_Message
MC_Continue_Read_Message_To_Tag
MC_Continue_Read_Message
C_Register_Callback_Function
MC_Free_Message

MC_Read_Message_To_Tag

Reads the next message to arrive from the remote applicate up until a specified tag.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Read_Message_To_Tag (
      int AssociationID,
      int Timeout,
      unsigned long StopTag,
      int* MessageID,
      char** ServiceName,
      MC_COMMAND* Command
)
```

AssociationID	The association object's identification number
Timeout	The max time (in seconds) to wait for a message to start arriving. A value of zero(0) means "check one time" and a value of minus one (-1) means "wait forever".
StopTag	Read up to and include this tag when reading from the remote application. Must be outside the command section (greater than or equal to (0001,0000))
MessageID	The ID of a message object containing the attributes of the message is returned here.
ServiceName	The name of the service associated with *MessageID* is returned here. Note well that this memory is freed when the **MC_Free_Message()** function is called for *MessageID*.
Command	The command which is associated with the message is returned here. The **MC_COMMAND** enumerated values are defined in "mc3msg.h".

Remarks

MC_Read_Message_To_Tag allows the caller to wait for the arrival of a request or response message from the remote application. If a message is not received on the association identified by *AssociationID* within *Timeout* seconds, the function returns a status of MC_TIMEOUT. Note that if *Timeout* is set to less than zero, **MC_Read_Message_To_Tag** will wait forever for a message to arrive. If it is set to zero, it will check one time if a message has arrived and then return.

Once a message begins arriving, the **INACTIVITY_TIMEOUT** configuration value is used. While receiving the message, the association will be aborted if there is no activity on the network for the length of time specified by this configuration value.

If a message arrives, **MessageID* is set to the ID of a message object which has been opened to receive the attributes of the message, up to, and including, the specified *StopTag*. The message's attribute values have been set by the DICOM message stream received from the remote application. If **MC_Register_Callback_Function** was called for any message attributes, those values were passed on to the registered callback function by the time this function (**MC_Read_Message_To_Tag**) returns.

The message prior to this call must have been read with **MC_Read_Message**, or read up to the last tag (FFFF,FFFF) either explicitly with **MC_Continue_Read_Message_To_Tag**, or by using **MC_Continue_Read_Message**.

The message returned must be read until the last tag (FFFF,FFFF) either explicitly with **MC_Continue_Read_Message_To_Tag**, or by calling **MC_Continue_Read_Message** before using **MC_Read_Message** or **MC_Read_Message_To_Tag** again.

ServiceName is set to the name of the service associated with *MessageID* and *Command* is set to the command associated with *MessageID.*

NOTE: It is the responsibility of the caller to release the message object. This is done using the **MC_Free_Message** call.

If **MC_Read_Message_To_Tag** returns either **MC_ASSOCIATION_CLOSED**, **MC_ASSOCIATION_ABORTED**, **MC_NETWORK_SHUT_DOWN, MC_INACTIVITY_TIMEOUT, MC_CONFIG_INFO_ERROR** or **MC_INVALID_MESSAGE_RECEIVED** no further calls may be made for the association and no message ID is returned at **MessageID*.

The following commands are supported by the Advanced MergeCOM-3 Tool Kit:

Command	Description
C_STORE_RQ	DICOM Composite Store Service Request
C_STORE_RSP	DICOM Composite Store Service Response
C_ECHO_RQ	DICOM Verification Service Request
C_ECHO_RSP	DICOM Verification Service Response
C_FIND_RQ	DICOM Composite Find Service Request
C_FIND_RSP	DICOM Composite Find Service Response
C_CANCEL_FIND_RQ	Cancel DICOM Composite Find Service Request
C_GET_RQ	DICOM Composite Get Service Request
C_GET_RSP	DICOM Composite Get Service Response
C_CANCEL_GET_RQ	Cancel DICOM Composite Get Service Request
C_MOVE_RQ	DICOM Composite Move Service Request
C_MOVE_RSP	DICOM Composite Move Service Response
C_CANCEL_MOVE_RQ	Cancel DICOM Composite Move Service Request
N_EVENT_REPORT_RQ	DICOM Normalized Report Service Request
N_EVENT_REPORT_RSP	DICOM Normalized Report Service Response
N_GET_RQ	DICOM Normalized Get Service Request
N_GET_RSP	DICOM Normalized Get Service Request
N_SET_RQ	DICOM Normalized Set Service Request
N_SET_RSP	DICOM Normalized Set Service Response
N_ACTION_RQ	DICOM Normalized Action Service Request
N_ACTION_RSP	DICOM Normalized Action Service Response
N_CREATE_RQ	DICOM Normalized Create Service Request
N_CREATE_RSP	DICOM Normalized Create Service Response
N_DELETE_RQ	DICOM Normalized Delete Service Request
N_DELETE_RSP	DICOM Normalized Delete Service Response

Reference the "DICOM V3.0 Standard, Final Text - October 29, 1993" for more information about these commands.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_MUST_CONTINUE_BEFORE_READING	A previous message was not read completely before making this call.
MC_TIMEOUT	The *Timeout* period expired.
MC_NULL_POINTER_PARM	The *MessageID*, *ServiceName* or *Command* parameter was NULL.
MC_INVALID_ASSOC_ID	*AssociationID* is not a valid association object ID.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.
MC_INVALID_TAG	*StopTag* was within the command set (less then (0001,0000)).
The association is dropped if any of the following are returned:	
MC_NETWORK_SHUT_DOWN	The network connect unexpectedly dropped.
MC_ASSOCIATION_ABORTED	The association has been aborted. A message describing the error has been written to the MergeCOM-3 log file.
MC_ASSOCIATION_CLOSED	The association has been closed.
MC_INACTIVITY_TIMEOUT	A timeout occurred in the middle of receiving a message.
MC_CONFIG_INFO_ERROR	The message information file describing the message's service/command pair could not be accessed.
MC_INVALID_MESSAGE_RECEIVED	An improperly formatted DICOM message was received. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Read_Message
MC_Continue_Read_Message_To_Tag
MC_Continue_Read_Message
MC_Register_Callback_Function
MC_Free_Message

MC_Register_Application

Registers a MergeCOM-3 application

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Register_Application (
      int* ApplicationID,
      char* AEtitle
)
```

ApplicationID	A unique application identification number will be returned here
AEtitle	The DICOM application title used by this application..

Remarks

MC_Register_Application registers an application with MergeCOM-3. This function provides MergeCOM-3 with the information it needs to identify this DICOM application and to maintain information specific to this application.

AEtitle is the DICOM name of the application. *AEtitle* must be made known to any other DICOM application which wishes to communicate with this one.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	*ApplicationID* or *AEtitle* was NULL.
MC_NO_LICENSE	A license number was not specified in the MergeCOM-3 configuration file.
MC_INVALID_LICENSE	The license number specified in the MergeCOM-3 configuration file was invalid.
MC_INVALID_LENGTH_FOR_TITLE	*AEtitle* must be 1 to 16 bytes long.
MC_ALREADY_REGISTERED	An application title with this name has already been registered.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Release_Application
MC_Register_Callback_Function
MC_Set_Message_Callbacks
MC_Set_Negotiation_Info

MC_Register_Callback_Function
MC_Register_pCallback_Function

Registers a callback function to provide or retrieve a standard or private attribute's values as they are transmitted or received on a DICOM association or read or written to a DICOM file

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Register_Callback_Function (
      int ApplicationID,
      unsigned long Tag,
      void* UserInfo,
      MC_STATUS (*CallbackFunction)()
)

MC_STATUS MC_Register_pCallback_Function (
      int ApplicationID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      void* UserInfo,
      MC_STATUS (*pCallbackFunction)
)
```

ApplicationID	The identifier assigned to this application by the **MC_Register_Application** function.
Tag	DICOM tag which identifies the attribute to which this callback function applies. *Tag* must specify an attribute with a value representation of OB, OW, OL, or OF.
PrivateCode	The code string which dictates which block in the private *Group* "owns" the attribute.
Group	The number identifying the private group containing the private attribute. It must be an odd number.
ElementByte	The number identifying the private attribute within the private *Group* for this *PrivateCode*.
UserInfo	Address of data which will be passed on to *CallbackFunction* each time it is called. This may be NULL.
CallbackFunction or *pCallbackFunction*	Name of a function which will be called repeatedly to provide blocks of data or to request blocks of data for the attribute's value.

The functions must be prototyped as follows:

```
MC_STATUS CallbackFunction (
      int CBmessageID,
      unsigned long CBtag,
      void* CBuserInfo,
      CALLBACK_TYPE CBtype,
      unsigned long* CBdataSizePtr,
      void** CBdataBufferPtr,
      int CBisFirst,
      int* CBisLastPtr
)
```

```
MC_STATUS pCallbackFunction (
      int CBmessageID,
      char* CBprivateCode,
      unsigned short CBgroup,
      unsigned char CBelementByte,
      void* CBuserInfo,
      CALLBACK_TYPE CBtype,
      unsigned long* CBdataSizePtr,
      void CBdataBufferPtr,
      int CBisFirst,
      int* CBisLastPtr
)
```

CBmessageID	MergeCOM-3 sets this to the identifier assigned to the message, item or file object by the **MC_Open_Message** function.
CBtag	MergeCOM-3 sets this to the DICOM tag which identifies the attribute.
CBprivateCode	The code string which dictates which block in the private *CBgroup* "owns" the attribute.
Cbgroup	The number identifying the private group containing the private attribute. It must be an odd number.
CbelementByte	The number identifying the private attribute within the private *CBgroup* for this *CBprivateCode*.
CbuserInfo	MergeCOM-3 sets this to the address of data which is being passed from the **MC_Register_Callback_Function** function. This may be NULL.
Cbtype	When the ***CallbackFunction*** is called, this is set to one of the following enumerated CALLBACK_TYPE codes as defined in "mergecom.h": REQUEST_FOR_DATA REQUEST_FOR_DATA_LENGTH PROVIDING_DATA PROVIDING_DATA_LENGTH PROVIDING_MEDIA_DATA_LENGTH PROVIDING_OFFSET_TABLE
CbdataSizePtr	If *CBtype* is REQUEST_FOR_DATA, then you must set **CbdataSizePtr* to the number of bytes of data you are providing at **CBdataBufferPtr*. This must be an even number and its value must be no larger than INT_MAX. If *CBtype* is PROVIDING_DATA, then MergeCOM-3 has set **CBdataSizePtr* to the number of bytes of data it is providing at **CBdataBufferPtr*. If *CBtype* is REQUEST_FOR_DATA_LENGTH then you must set **CBdataSizePtr* to the total size (in bytes) of the attribute's value. This must be an even number. If *CBtype* is PROVIDING_DATA_LENGTH then MergeCOM-3 has set **CBdataSizePtr* to the total size (in bytes) of the attribute's value. MergeCOM-3 will call the function with *CBtype* of PROVIDING_DATA_LENGTH before it calls the function with *CBtype* of PROVIDING_DATA. If *CBtype* is PROVIDING_MEDIA_DATA_LENGTH then

	MergeCOM-3 has set **CBdataSizePtr* to the total size (in bytes) of the attribute's value. If *CBtype* is PROVIDING_OFFSET_TABLE then MergeCOM-3 has set **CBdataSizePtr* to the number of bytes of data it is providing at **CbdataBufferPtr*.
CbdataBufferPtr	If *CBtype* is REQUEST_FOR_DATA or PROVIDING_DATA this points at the address of the buffer containing a portion of the attribute's value. The value for this field must be even because of alignment problems on some platforms. If *CBtype* is REQUEST_FOR_DATA_LENGTH or *CBtype* is PROVIDING_DATA_LENGTH this parameter is not used. If *CBtype* is PROVIDING_MEDIA_DATA_LENGTH, **CbdataBufferPtr* is a pointer to long that contains the offset of the attribute's value from the beginning of the DICOM file.
CbisFirst	If *CBtype* is REQUEST_FOR_DATA, then MergeCOM-3 will set *CBisFirst* to TRUE (not zero) the first time it is requesting data for this attribute's value (i.e. when you are to provide the first block of data). If *CBtype* is PROVIDING_DATA , then MergeCOM-3 will set *CBisFirst* to TRUE (not zero) the first time it is providing data for this attribute's value (i.e. when it is providing the first block of data). If *CBtype* is PROVIDING_OFFSET_TABLE, then MergeCOM-3 will set *CBisFirst* to TRUE (not zero) the first time it is providing data for the basic offset table. If *CBtype* is REQUEST_FOR_DATA_LENGTH, *CBtype* is PROVIDING_DATA_LENGTH, or *CBtype* is PROVIDING_MEDIA_DATA_LENGTH this parameter is not used.
CbisLastPtr	If *CBtype* is REQUEST_FOR_DATA, then you must set **CBisLastPtr* to TRUE (not zero) the last time you are providing data for this attribute's value (i.e. when you are providing the last block of data). If *CBtype* is PROVIDING_DATA, then MergeCOM-3 has set **CBisLastPtr* to TRUE (not zero) because this is the last time it will be providing data for this attribute's value (i.e. when it is providing the last block of data). If *CBtype* is REQUEST_FOR_DATA_LENGTH, *CBtype* is PROVIDING_DATA_LENGTH or *CBtype* is PROVIDING_MEDIA_DATA_LENGTH this parameter is not used.

Remarks

MC_Register_Callback_Function registers *CallbackFunction* to provide or retrieve an attribute's values as they are transmitted or received on a DICOM association or read or written to a DICOM file. The callback will also be called if the user calls **MC_Get_Value_To_Function** or **MC_Set_Value_From_Function** although this should not happen because the user already is storing the OB, OW, or OF data. **MC_Register_Callback_Function** may only be used for attributes with a value representation of OB, OW, or OF.

MC_Register_pCallback_Function registers *pCallbackFunction* to provide or retrieve a private attribute's values as they are transmitted or received on a DICOM association or read or written to a DICOM file. The callback will also be called if the user calls **MC_Get_pValue_To_Function** or **MC_Set_pValue_From_Function** although this should not happen because the user already is storing the OB, OW, or OF data. **MC_Register_Callback_Function** may only be used for attributes with a value representation of OB, OW, or OF. Subsequent descriptions in this section are applicable for both standard and private registered callback functions.

NOTE: If a value has already been set for *Tag*, *CallbackFunction* will not be called to set or get its value.

NOTE: The **CALLBACK_MIN_DATA_SIZE** configuration value specifies the minimum sized value for which a callback function is used. This configuration option can be used so that only messages or files with very large attribute values use a callback function.

NOTE: Some message definitions contain OB, OW, or OF data but a value does not have to be set for these attributes (such as C-FIND-RQ and C-FIND-RSP messages). To have the registered callback function ignored, the attribute should be deleted from the message with the **MC_Delete_Attribute** function.

If a callback function was already registered, *CallbackFunction* replaces the function previously registered. A callback function may be "de-registered" by calling **MC_Release_Callback_Function** for the same attribute.

The **MC_Register_Callback_Function** caller may provide the *CallbackFunction* with a pointer to agreed upon data in its *UserInfo* parameter. This is optional and *UserInfo* may be NULL.

Network Communications:

When the MergeCOM-3 communications software receives a message on a DICOM association, it uses **MC_Open_Message** to acquire a message compatible with the message being received. It then populates the message with the values it receives over the association. When it receives attributes of type OB, OW, or OF (or any other large values) the message library stores the values in a configurable location (normally in temporary files). Then, the receiving application could use any of the "Get" functions (including **MC_Get_Value_To_Function**) to retrieve attribute values from the message object.

Some applications may wish to bypass the storing of these large values in the message object. To do so, the **MC_Register_Callback_Function** is used to register a function which will be called as the attribute's value arrives on the association; blocks of the value are then passed to this "*CallbackFunction*" instead of storing the value in the message.

Similarly, a client application may use MC_Register_Callback_Function to register a callback function to provide MergeCOM-3 with an attribute's value as MergeCOM-3 needs the value for transmission on the network. *"CallbackFunction"* will be called when *Tag* is encountered in any message being transferred between this *ApplicationID* and a remote application.

Media Applications:

When the MergeCOM-3 media software reads a file with **MC_Open_File**, it populates the file object with the values it receives from media. When it receives attributes of type OB, OW, or OF (or any other large values) the file library stores the values in a configurable location (normally in temporary files). Then, the reading application could

use any of the "Get" functions (including **MC_Get_Value_To_Function**) to retrieve attribute values from the file object.

Some applications may wish to bypass the storing of these large values in the file object. To do so, the **MC_Register_Callback_Function** is used to register a function which will be called as the attribute's value is read from media. The user has two options when the attribute's value is read from media. If the file is being read with the function **MC_Open_File** or **MC_Open_File_Upto_Tag** it behaves in the same manner as with associations. The OB, OW, or OF data is passed to *CallbackFunction* instead of storing it in the file object. However, when **MC_Open_File_Bypass_OBOW** is used, the OB, OW, or OF data is left on media. The location and length of the data within the file is passed to *CallbackFunction*. The actual data is not passed. Whenever *Tag* is encountered in any file being read within *ApplicationID, CallbackFunction* will be called.

In some cases, *CallbackFunction* must also supply MergeCOM-3 with the OB, OW, or OF data. If the file object is converted to a message object with **MC_File_To_Message** and sent over the network, MergeCOM-3 will request the attribute's value from *CallbackFunction* when it needs the value for transmission. Similarly, if the file object is re-written to media, *CallbackFunction* will also be requested for the attribute's value.

The *CallbackFunction:*

MergeCOM-3 passes the *CallbackFunction* both *CBmessageID* and *CBtag* to identify which attribute is involved. **MC_Get_Message_Service** may be called to identify which service and command this *CBmessageID* relates to. The *CBuserInfo* parameter is optional data from the **MC_Register_Callback_Function** caller.

CallbackFunction must return a status of **MC_NORMAL_COMPLETION** if it could accommodate the callback request, or **MC_CANNOT_COMPLY**, if not. These status codes are defined in "mcstatus.h."

CBtype is set to REQUEST_FOR_DATA_LENGTH if MergeCOM-3 is requesting the size (in bytes) of the attribute's value. In this case *CallbackFunction* must set **CBdataSizePtr* to the attribute's value size. . MergeCOM-3 will call the *CallbackFunction* with *CBtype* set to REQUEST_FOR_DATA_LENGTH before it begins calling *CallbackFunction* with *CBtype* set to REQUEST_FOR_DATA.

CBtype is set to PROVIDING_DATA_LENGTH if MergeCOM-3 is providing *CallbackFunction* with the total size of the attribute's value. The value size (in bytes) is placed at **CBdataSizePtr*. MergeCOM-3 will call the *CallbackFunction* with *CBtype* set to PROVIDING_DATA_LENGTH before it begins calling *CallbackFunction* with *CBtype* set to PROVIDING_DATA.

CBtype is set to REQUEST_FOR_DATA if MergeCOM-3 is requesting data from the *CallbackFunction*. In this case *CallbackFunction* sets **CBdataBufferPtr* to the address of a block of the attribute's value, and **CBdataSizePtr* to the number of bytes at **CBdataBufferPtr*. If this is the first time MergeCOM-3 is calling *CallbackFunction* for this attribute's data *CBisFirst* will be TRUE(non-zero). If *CallbackFunction* is returning the last block of data to be provided for this attribute, it must set **CBisLastPtr* to TRUE(non-zero). MergeCOM-3 will repeatedly call *CallbackFunction* until **CBisLastPtr* is set to TRUE or *CallbackFunction* returns with a status other than **MC_NORMAL_COMPLETION**.

CBtype is set to PROVIDING_DATA if MergeCOM-3 is providing data to the *CallbackFunction*. In this case MergeCOM-3 has set **CBdataBufferPtr* to the address of a block of the attribute's value, and **CBdataSizePtr* to the number of bytes at **CBdataBufferPtr*. If this is the first time MergeCOM-3 is calling *CallbackFunction* for this attribute's data *CBisFirst* will be TRUE(non-zero). If this is the last block of data to

be provided for this attribute, MergeCOM-3 has set **CBisLastPtr* to TRUE(non-zero). MergeCOM-3 will repeatedly call *CallbackFunction* until all of the value has been provided. (At that time it sets **CBisLastPtr* to TRUE.).

CBtype is set to PROVIDING_OFFSET_TABLE if MergeCOM-3 is providing basic offset table to the *CallbackFunction*. In this case MergeCOM-3 has set **CBdataBufferPtr* to the address of the offset table, and **CBdataSizePtr* to the number of bytes at **CBdataBufferPtr*. If this is the first time MergeCOM-3 is calling *CallbackFunction* for this offset table's data *CBisFirst* will be TRUE(non-zero).

NOTE: **CBdataBufferPtr* may be NULL when **CBisLastPtr* is returned TRUE.

NOTE: The *CBtype* PROVIDING_OFFSET_TABLE is given to the callback function when the pixel data is encapsulated.

NOTE: *CallbackFunction* is not called with this value when a file is opened with **MC_Open_File_Bypass_OBOW** because the OB, OW, or OF data is left on media.

CBtype is set to PROVIDING_MEDIA_DATA_LENGTH if MergeCOM-3 is reading a file from media by use of the **MC_Open_File_Bypass_OBOW** function. *CallbackFunction* is provided with the total size of the attribute's value and the byte offset from the beginning of the media file of the attribute's value. The value size (in bytes) is placed at **CBdataSizePtr*. ***CBdataBufferPtr* is a long int that contains the byte offset of the attribute's value from the beginning of the media file. MergeCOM-3 will not call the *CallbackFunction* with *CBtype* set to PROVIDING_DATA when it calls *CallbackFunction* with *CBtype* set to PROVIDING_MEDIA_DATA_LENGTH.

If *CallbackFunction* returns with a status other than MC_NORMAL_COMPLETION, MergeCOM-3 will stop calling it.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	*CallbackFunction*, *pCallbackFunction* or *PrivateCode* were NULL.
MC_INVALID_APPLICATION_ID	ApplicationID does not identify a valid application object.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Set_Value_From_Function
MC_Set_pValue_From_Function
MC_Get_Message_Service
MC_Open_File_Bypass_OBOW
MC_Release_Callback_Function
MC_Set_Message_Callbacks
MC_Get_Value_To_Function
MC_Get_pValue_To_Function
MC_Open_File
MC_Open_File_Upto_Tag
MC_Register_Application

MC_Register_Compression_Callbacks

Registers compression/decompression callback functions to compress or decompress a standard or private attribute's values. The callbacks are automatically used when **MC_Set_Encapsulated_Value, MC_Set_Next_Encapsulated_Value, MC_Get_Encapsulated_Value, MC_Get_Next_Encapsulated_Value,** and **MC_Duplicate_Message** are called.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Register_Compression_Callbacks (
      int MsgFileItemID,
      MC_STATUS (*CompressionCallbackFunction)()
      MC_STATUS (*DecompressionCallbackFunction)()
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
CompressionCallback Function	Name of a function which will be called repeatedly with blocks of uncompressed data, and will return compressed data.
DecompressionCallb ackFunction	Name of a function which will be called repeatedly with compressed blocks of data, and will return decompressed data

The functions must be prototyped as follows:

```
MC_STATUS YourDestMsg(De)CompressionCallback (
      int CBmsgID,
      void** CBcontext,
      unsigned long CBdataLength,
      void* CBdataValue,
      unsigned long* CBoutdataLength,
      void** CBoutdataValue,
      int CBisFirst,
      int CBisLast
      int CBrelease
)
```

CBmsgID	The identifier assigned to this message object by the **MC_Open_Message**, **MC_Open_Empty_Message** functions.
CBcontext	A data structure that contains user data and has to be preserved between compression calls. The first call to the callback function should initialize this structure.
CBdataLength	The length of the incoming data pointed to by CBdataValue.
CBdataValue	Pointer to incoming data.
CBoutdataLength	The length of the outgoing data pointed to by CBoutdataValue.
CBoutdataValue	Pointer to the outgoing data.
CBisFirst	This is TRUE (not zero) the first time ***YourDestMsg(De)CompressionCallback*** is being called.
CBisLast	This is TRUE (not zero) the last time ***YourDestMsg(De)CompressionCallback*** is being called.
CBrelease	This is TRUE (not zero) if the callback should release all the context memory and return.

Remarks

MC_Register_Compression_Callbacks may only be used for attributes with a value representation of OB or OW. The built-in callbacks can be used, or the user may implement their own (Please see Users Manual, "Using Compression/Decompression Callback Functions")

The toolkit has a built in compressor/decompressor, **MC_Standard_Compressor, MC_Standard_Decompressor** which are capable of:

Grayscale:
JPEG_BASELINE (8 bit)
JPEG_EXTENDED_2_4 (8,10,12 bit)
JPEG_LOSSLESS_HIER_14 (8,10,12,16 bit)
JPEG_2000 (8,10,12,16 bit)
JPEG_2000_LOSSLESS_ONLY (8,10,12,16 bit)

Other possible inputs/outputs:
8 bits/pixel, 3 samples/pixel RGB
8 bits/pixel, 3 samples/pixel YBR_FULL_422

If a callback function was already registered, the passed in callback replaces the function previously registered. A callback function may be "de-registered" by calling **MC_Register_Compression_Callbacks** with NULL as the parameter.

The **MC_Register_Callback_Function** caller may provide the *CallbackFunction* with a pointer to agreed upon data in its *UserInfo* parameter. This is optional and *UserInfo* may be NULL.

NOTE: **CBdataBufferPtr* may be NULL when **CBisLastPtr* is returned TRUE.

ALSO NOTE FOR MC_Standard_(De)compressor:
JPEG_2000/JPEG_2000_LOSSLESS_ONLY will cause a irreversible, or reversible color transformation when compressing RGB data. The Photometric Interpretation MUST be changed from RGB to:

1) YBR_ICT if JPEG_2000 is used with COMPRESSION_WHEN_J2K_USE_LOSSY = Yes (Lossy color transform for lossy compression)

2) YBR_RCT if JPEG_2000_LOSSLESS_ONLY, or JPEG_2000 with COMPRESSION_WHEN_J2K_USE_LOSSY = No (Lossless color transform for lossless compression).

Similarly, on the decompression end, the Photometric Interpretation should be changed back to RGB, but the Lossy Image Compression attribute should indicate if it had been compressed lossy.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_TRANSFER_SYNTAX	MsgFileItemID's transfer syntax is not one of the encapsulated transfer syntaxs.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment.

A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Set_Encapsulated_Value_From_Function
MC_Set_Next_Encapsulated_Value_From_Function
MC_Get_Encapsulated_Value_From_Function
MC_Get_Next_Encapsulated_Value_From_Function
MC_Duplicate_Message

MC_Register_Enhanced_MemoryLog_Function

Registers a callback function that will be notified as each log message is written

Synopsis

```
#include "mc3msg.h"

void MC_Register_Enhanced_MemoryLog_Function (
                                    void (*MemLogFunction)()
)
```

MemLogFunction The name of a function that will be called each time MergeCOM-3 emits a log message.

MemLogFunction must be prototyped as follows:

```
void (*MemLogFunction)( LogInfo* logInfo),
```

logInfo A structure containing information about the message that is being logged.

The **LogInfo** structure is defined as follows:

```
typedef struct LogInfo_struct
{
    MsgLogType  typeCode;
    LogTime     timeValues;
    char*       type;
    char*       prefix;
    char*       processID;
    char*       timeStamp;
    char*       function;
    char*       message;
} LogInfo;
```

typeCode The type of message being logged. It will be one of the following constants defined by the **MsgLogType** enumeration:

MC_ERROR_MESSAGE
MC_WARNING_MESSAGE
MC_INFO_MESSAGE
MC_T1_MESSAGE
MC_T2_MESSAGE
MC_T3_MESSAGE
MC_T4_MESSAGE
MC_T5_MESSAGE
MC_T6_MESSAGE
MC_T7_MESSAGE
MC_T8_MESSAGE
MC_T9_MESSAGE

timeValues A **LogTime** structure containing the exact time the message was being logged. (See structure definition below.)

type The string representation of the message type. For example, "E", "W", "I", "T1" etc.

prefix	An option prefix string. Normally this will be an empty string. The **MC_Set_Log_Prefix** function is used to provide a string that will be prefixed to all log messages.
processID	A string that represents the ID of the process that generated the message. In some operating environments this may be an empty string.
timeStamp	A string representation of the time the message was logged. This string may or may not be included in the actual log *message*.
function	A string representation of the MergeCOM-3 function that generated the message. This may be an empty string.
message	The log message. This is the string normally logged to screen or to file.

The **LogTime** structure that is included in the returned *logInfo* is defined as follows:

```
typedef struct LogTime_struct {
        int hour;
        int min;
        int sec;
        int hun;
        int day;
        int mon;
        int year;
} LogTime;
```

hour	hours since midnight - [0,23]
min	minutes after the hour - [0,59]
sec	seconds after the minute - [0,59]
hun	hundredths of a sec after the sec [0,99]
day	day of the month - [1,31]
mon	months since January - [0,11]
year	years since 1900

Remarks

This function registers an enhanced message log handler to process system log messages. (See **MC_Register_MemoryLog_Function** for information about registering a standard log handler.)

Once registered, all system log messages are passed to the *MemLogFunction*. Information about the logged message is passed by the *logInfo* parameter. Refer to the above description of the **LogInfo** structure for detailed information about the data that is made available.

Only one enhanced message log handler may exist at any one time. If this function is called again, any existing enhanced log handler will no longer be called. The *MemLogFunction* parameter may be null to de-register the last enhanced handler.

Log messages can be directed to one or more of four destinations by settings in the merge.ini file, or by setting the destinations at runtime using the **MC_Set_Log_-Destination** function. Different types of log messages may be directed to the screen, to a file, to memory, or to the "bit bucket". Messages directed to the "bit bucket" are those that have no destination set for them.

Note that all messages with a screen or file destination are also passed on to the *MemLogFunction*. If you wish to capture log messages without sending messages to a file or screen destination, set the memory destination in the merge.ini file.

See Also

MC_Set_Log_Destination
MC_Set_Log_Prefix
MC_Register_MemoryLog_Function

MC_Register_MemoryLog_Function

Registers a callback function that will be notified as each log message is written

Synopsis

```
#include "mc3msg.h"

void MC_Register_MemoryLog_Function(
                                    void (*MemLogFunction)()
)
```

MemLogFunction The name of a function that will be called each time MergeCOM-3 emits a log message.

MemLogFunction must be prototyped as follows:

```
void (*MemLogFunction)(MsgLogType MsgType,
                       char*      Msg)
```

MsgType The type of message being logged. It will be one of the following constants defined by the **MsgLogType** enumeration:

- MC_ERROR_MESSAGE
- MC_WARNING_MESSAGE
- MC_INFO_MESSAGE
- MC_TRACE_MESSAGE

Msg The message string that is being logged

Remarks

This function registers a standard message log handler to process system log messages. (See **MC_Register_Enhanced_MemoryLog_Function** for information about registering an enhanced log handler.)

Once registered, all system log messages are passed to the *MemLogFunction*. The type of message being logged is provided by the *MsgType* parameter and the message string itself is passed by the *Msg* parameter.

Only one standard message log handler may exist at any one time. If this function is called again, any existing standard log handler will no longer be called. The *MemLogFunction* parameter may be null to de-register the last standard handler.

Log messages can be directed to one or more of four destinations by settings in the merge.ini file, or by setting the destinations at runtime using the **MC_Set_Log_-Destination** function. Different types of log messages may be directed to the screen, to a file, to memory, or to the "bit bucket". Messages directed to the "bit bucket" are those that have no destination set for them.

Note that all messages with a screen or file destination are also passed on to the *MemLogFunction*. If you wish to capture log messages without sending messages to a file or screen destination, set the memory destination in the merge.ini file.

See Also

MC_Set_Log_Destination
MC_Register_Enhanced_MemoryLog_Function

MC_Register_Network_Capture_Callbacks

Registers a set of callback functions that will be used to capture network data, overriding the default network capture handler.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Register_Network_Capture_Callbacks (
      MC_NetworkCaptureInfo *AcallbackInfo
)
```

AcallbackInfo — Pointer to an MC_NetworkCaptureInfo structure that provides an optional application context, plus various callback functions. If NULL, the default MergeCOM-3 handler functions will be used.

AcallbackInfo is a pointer to an MC_NetworkCaptureInfo type prototyped as follows:

```
typedef struct MC_NetworkCaptureInfo_Struct {
    void* CAP_applicationContext;

    int (NOEXP_FUNC *CAP_initialization) (
                void*          AppContext,
                long           MaxFileSize,
                char*          FileName,
                int            NumFiles,
                int            RewriteFiles);

    void (NOEXP_FUNC *CAP_shutdown) (
                void*          AppContext);

    void* (NOEXP_FUNC *CAP_getConnectionContext) (
                void*          AppContext,
                unsigned long  IPaddr1,
                unsigned long  IPaddr2,
                unsigned short Port1,
                unsigned short Port2);

    void (NOEXP_FUNC *CAP_freeConnectionContext) (
                void*          AppContext,
                void*          ConnContext);

    int (NOEXP_FUNC *CAP_conectionOpened) (
                void*          AppContext,
                void*          ConnContext,
                unsigned long  SourceIPaddr,
                unsigned long  DestIPaddr,
                unsigned short SourcePort,
                unsigned short DestPort);

    int (NOEXP_FUNC *CAP_conectionClosed) (
                void*          AppContext,
                void*          ConnContext,
                unsigned long  SourceIPaddr,
                unsigned long  DestIPaddr,
                unsigned short SourcePort,
```

```
                    unsigned short DestPort);

    int (NOEXP_FUNC *CAP_dataSent) (
                    void*          AppContext,
                    void*          ConnContext,
                    unsigned long  SourceIPaddr,
                    unsigned long  DestIPaddr,
                    unsigned short SourcePort,
                    unsigned short DestPort,
                    char*          Data,
                    long           DataLength);
} MC_NetworkCaptureInfo;
```

The functions provided in the MC_NetworkCaptureInfo structure will override the default network capture handler provided by MergeCOM-3. (Please refer to the *MergeCOM-3 Users Manual* for more information about the default handler.)

CAP_applicationContext - An optional pointer to application context data. MergeCOM-3 passes this pointer when it calls the callback functions. It may be NULL.

CAP_initialization - MergeCOM-3 calls this function at startup time (if the NETWORK_CAPTURE configuration parameter is Yes), whenever a network capture configuration value is changed at runtime, and whenever new capture functions are registered. Five configuration parameters may be set in the [TRANSPORT_PARMS] section of the mergecom.pro file:

NETWORK_CAPTURE = Yes/No - Determines whether or not MergeCOM-3 should call the capture handler functions.

CAPTURE_FILE_SIZE = n - Sets the maximum size in kilobytes of capture files, with zero meaning unlimited size.

CAPTURE_FILE = <filename> - Provides the base name to use for capture files.

NUMBER_OF_CAP_FILES = n **-** Provides the maximum number of capture files to create.

REWRITE_CAPTURE_FILES = Yes/No - Determines whether capture files will be reused or not.

Each of the above values may also be set at runtime using the appropriate MC_Set_XXX_Config_Value function.

CAP_initialization is called with these parameters:

AppContext	The application context provided by **CAP_applicationContext.**
MaxFileSize	the current value of the **CAPTURE_FILE_SIZE** configuration parameter
FileName	the current value of the **CAPTURE_FILE** configuration parameter

NumFiles	the current value of the **NUMBER_OF_CAP_FILES** configuration parameter
RewriteFiles	the current value of the **REWRITE_CAPTURE_FILES** configuration parameter

CAP_shutdown - MergeCOM-3 calls this function when the library is released, or when NETWORK_CAPTURE is turned off after it was on. The function is called with one parameter:

AppContext	The application context provided by CAP_applicationContext.

CAP_getConnectionContext - MergeCOM-3 calls this function when it has opened a new association connection. The function is expected to return a pointer to a context area that will be used to process the connection's capture information. MergeCOM-3 will pass this pointer (as the *ConnContext* parameter) to other callbacks (see below). MergeCOM-3 provides the TCP/IP addresses used on the connection. The IP addresses are presented in network order (i.e. big endian).

AppContext	The application context provided by CAP_applicationContext.
IPaddr1	The first Internet Protocol address used on the connection.
Ipaddr2	The second Internet Protocol address used on the connection..
Port1	The TCP port number used with *IPaddr1*.
Port2	The TCP port number used with *Ipaddr2*.

CAP_freeConnectionContext - MergeCOM-3 calls this function when it closes a network connection. MergeCOM-3 will never use the *ConnContext* pointer again.

AppContext	The application context provided by CAP_applicationContext.
ConnContext	The connection context provided by the CAP_getConnectionContext callback.

CAP_conectionOpened - MergeCOM-3 calls this function when it has opened a new association connection after calling CAP_getConnectionContext. The source and destination addresses are provided. The function is called with these parameters:

AppContext	The application context provided by CAP_applicationContext.
ConnContext	The connection context provided by the CAP_getConnectionContext callback.
SourceIPaddr	The Internet Protocol address of the node that opened the connection.
DestIPaddr	The Internet Protocol address of the node that accepted the connection.
SourcePort	The port number of the node that opened the connection.
DestPort	The port number of the node that accepted the connection.

CAP_conectionClosed - MergeCOM-3 calls this function

AppContext	The application context provided by CAP_applicationContext.
ConnContext	The connection context provided by the CAP_getConnectionContext callback.
SourceIPaddr	The Internet Protocol address of the node that opened the connection.
DestIPaddr	The Internet Protocol address of the node that accepted the connection.
SourcePort	The port number of the node that opened the connection.
DestPort	The port number of the node that accepted the connection.

CAP_dataSent - MergeCOM-3 calls this function

AppContext	The application context provided by CAP_applicationContext.
ConnContext	The connection context provided by the CAP_getConnectionContext callback.
SourceIPaddr	The Internet Protocol address of the node that sent the data.
DestIPaddr	The Internet Protocol address of the node that received the data.
SourcePort	The port number of the node that sent the data.
DestPort	The port number of the node that received the data.

Remarks

MergeCOM-3 provides a facility to capture raw data that is transmitted over a network connection. The data is captured in files that can be read by the MergeDPM utility to analyze the network activity. The network capture facility is controlled by the configuration parameters discussed in the above section.

In rare cases you may wish to replace the default MergeCOM-3 network capture facility with one of your own. The **MC_Register_Network_Capture_Callbacks** function is used to register the functions that you will provide to capture network data, overriding the default handler functions.

Refer to the above description of the individual callback functions for details of the network capture interface.

Note that received network data is provided by MergeCOM-3 after it has been optionally decrypted by any secure socket functions (see **MC_Open_Secure_Association** and **MC_Wait_For_Secure_Association**). Likewise, the data is provided to the callbacks before it sends the data to secure socket functions for possible encryption.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	One or more of the functions provided in the MC_NetworkCaptureInfo structure was NULL.

See Also

MC_Open_Secure_Association
MC_Wait_For_Secure_Association

MC_Reject_Association

Reject a remote application's request for a DICOM association

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Reject_Association (
      int AssociationID,
      REJECT_REASON Reason
)
```

AssociationID The association object's identification number

Reason **The reason the DICOM association is being rejected. Use one of the enumerated REJECT_REASON types defined in "mergecom.h":**

PERMANENT_NO_REASON_GIVEN,
TRANSIENT_NO_REASON_GIVEN,
PERMANENT_CALLING_AE_TITLE_NOT_RECOGNIZED,
TRANSIENT_TEMPORARY_CONGESTION,
TRANSIENT_LOCAL_LIMIT_EXCEEDED

Remarks

If a **MC_Wait_For_Association** or **MC_Wait_For_Secure_Association** function completes normally, one of two functions must be called. **MC_Accept_Association** informs the remote application that the association can proceed. Use **MC_Reject_Association** to reject the association request.

Upon successful return from the **MC_Reject_Association** call, no further calls may be made for the association.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NO_REQUEST_PENDING	There is no pending association request for this *AssociationID*.
MC_INVALID_ASSOC_ID	*AssociationID* is not a valid association object ID.

See Also

MC_Wait_For_Association **MC_Wait_For_Secure_Association**
MC_Accept_Association

MC_Release_Application

De-register an application.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Release_Application (
      int* ApplicationID
)
```

ApplicationID Address of the application object's identification number

Remarks

MC_Release_Application releases the system resources used by the application. If the application has one or more associations open, they are aborted. Applications should always close or abort associations before calling **MC_Release_Application**.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	The *ApplicationID* parameter was NULL.
MC_INVALID_APPLICATION_ID	***ApplicationID* is not a valid application ID.

See Also

MC_Register_Application

MC_Release_Callback_Function

De-register a callback function for a given attribute.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Release_Callback_Function (
      int ApplicationID,
      unsigned long Tag
)
```

ApplicationID	The application object's identification number
Tag	The identifying tag of an attribute which had a callback function registered for it.

Remarks

MC_Release_Callback_Function releases the callback function which was registered for the attribute identified by *Tag*. The callback function will no longer be called when the attribute's value is being received or when the attribute's value is required.

The callback function will, however, still be used for messages that were opened before this function call was made.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_APPLICATION_ID	*ApplicationID* does not identify a valid MergeCOM-3 application.
MC_INVALID_TAG	There was no callback function registered for *Tag*.

See Also

MC_Register_Callback_Function

MC_Release_Parent_Association

Releases parent's copy of an association object.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Release_Parent_Association (
      int AssociationID
)
```

AssociationID The ID of an association object returned by a **MC_Wait_For_Association** or **MC_Wait_For_Secure_Association** function.

Remarks

MC_Release_Parent_Association is used by a parent process to free its copy of an association object after spawning a child process (and passing the association object to the child). This is often done in UNIX environments in which a server process loops, calling the **MC_Wait_For_Association** or **MC_Wait_For_Secure_Association** function. As each association request arrives, the process spawns a child process to handle the association, then uses this function to free its copy of the association object. IT IS VERY IMPORTANT that **MC_Release_Parent_Association** be called in this situation to avoid memory leaks.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_ASSOC_ID	*AssociationID* does not identify a valid MergeCOM-3 association.

See Also

MC_Wait_For_Association **MC_Wait_For_Secure_Association**

MC_Report_Memory

Reports information on the memory used by the toolkit's internal memory management.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Report_Memory (
      unsigned long* bytesAllocated,
      unsigned long* bytesInUse
)
```

bytesAllocated	The number of bytes that the toolkit has allocated.
bytesInUse	The number of bytes of non-freeable memory.

Remarks

This function steps through the toolkits internal memory structures to determine how much memory has been allocated, and how much of that is actually in use. This function is helpful in determining if a MC_Cleanup_Memory call should be made.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_LIBRARY_NOT_INITIALIZED	The library has not been properly initialized.
MC_NULL_POINTER_PARM	One or both of the parameters were non-initialized pointers.

See Also

MC_Cleanup_Memory

MC_Reset_Filename

Resets the filename associated with a file object

Synopsis

```
#include "mc3media.h"

MC_STATUS MC_Reset_Filename (
      int FileID,
      char* NewFilename
)
```

FileID	The identifier assigned to this object by the **MC_Create_Empty_File** or **MC_Create_File** function.
NewFilename	A pointer to a string containing the filename to be associated with the new file object.

Remarks

The **MC_Reset_Filename** function changes the filename associated with a file object.

The filename is passed to the user's callback function when the **MC_Write_File** function is called.

This function is useful when reusing file objects or saving a copy of a file.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_FILE_ID	The *FileID* value is not a valid file object ID.
MC_NULL_POINTER_PARM	*NewFilename* was NULL.

See Also

MC_Write_File **MC_Empty_File**

MC_Send_Request_Message

Sends a request message to the remote application.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Send_Request_Message (
      int AssociationID,
      int MessageID
)
```

AssociationID	The association object's identification number
MessageID	A message object's identification number

Remarks

MC_Send_Request_Message sends a request message to the remote application. The message is identified by *MessageID* and must have been created using the **MC_Open_Message** function. The values of the message attributes must have been set with the "Set..." message functions and/or the **MC_Stream_To_Message** function. The message is sent across an open DICOM association identified by *AssociationID*.

NOTE: If the message being sent was opened with **MC_Open_Empty_Message** and the pixel data is being supplied to the message with a registered callback function, the pixel data attribute should be explicitly added to the message with the function **MC_Add_Standard_Attribute** before **MC_Send_Request_Message** is called.

If **MC_Register_Callback_Function** was called to register a callback function for any attributes in the message, **MC_Send_Request_Message** calls that function to retrieve the attribute values as it sends the message.

NOTE: Some DICOM services require that values for certain "command level" message attributes (i.e. group 0 attributes) be set. (MergeCOM-3 automatically adds command level attributes to the message when **MC_Open_Message** is called. With the exceptions listed below, the caller must set any command level attribute values before sending the message. **MC_Send_Request_Message** does, however, set the following attribute values for you - **if you have not set them**:

- The group length attribute (0000,0000) value is always set by MergeCOM-3.
- If the message command requires it, the Affected SOP Class UID attribute (0000,0002) value is set to the service's abstract syntax UID.
- The command attribute (0000,0100) value is always set by MergeCOM-3.
- The Message ID attribute (0000,0110) value is set to a unique number for this association.
- If the message command requires it, the Priority attribute (0000,0700) value is set "medium" priority (i.e. zero).
- The Data Set Type attribute (0000,0800) value is always set by MergeCOM-3.

If **MC_Send_Request_Message** returns a status code of **MC_ASSOCIATION_ABORTED**, no further calls may be made for the association.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_ASSOC_ID	*AssociationID* is not a valid association object ID.
MC_INVALID_MESSAGE_ID	The *MessageID* value is not a valid message object ID.
MC_MAX_OPERATIONS_EXCEEDED	Sending of the message would cause the Maximum Number of Operations Invoked that was negotiated to be exceeded. A response message must be read before this message can be sent.
MC_REQUIRED_ATTRIBUTE_MISSING	Some DICOM services require certain command level attributes (group 0) to have values set. This status indicates that a required attribute value for the service related to *MessageID* has not been set.
MC_UNACCEPTABLE_SERVICE	The service related to *MessageID* has not been successfully negotiated for this association.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.
The association is dropped if any of the following are returned:	
MC_ASSOCIATION_ABORTED	The association has been aborted. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Register_Callback_Function
MC_Send_Request_Message_For_Service
MC_Send_Response_Message
MC_Open_Message
MC_Read_Message

MC_Send_Request_Message_For_Service

Sends a request message to the remote application.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Send_Request_Message_For_Service (
      int AssociationID,
      int MessageID,
      char *AServiceName
)
```

AssociationID	The association object's identification number
MessageID	A message object's identification number
AserviceName	Name of the service negotiated to utilize for the presentation context .

Remarks

MC_Send_Request_Message_For_Service sends a request message to the remote application. The presentation context used in the transfer is looked up based on the service specified in the call. This call is used to select the proper presentation context when the object is common to multiple services negotiated. An example is the FilmBox when both BASIC_GRAYSCALE_PRINT_MANAGEMENT and BASIC_COLOR_PRINT_MANAGEMENT are negotiated. The message is identified by *MessageID* and must have been created using the **MC_Open_Message** function. The values of the message attributes must have been set with the "Set..." message functions and/or the **MC_Stream_To_Message** function. The message is sent across an open DICOM association identified by *AssociationID*.

NOTE: If the message being sent was opened with **MC_Open_Empty_Message** and the pixel data is being supplied to the message with a registered callback function, the pixel data attribute should be explicitly added to the message with the function **MC_Add_Standard_Attribute** before **MC_Send_Request_Message** is called.

If **MC_Register_Callback_Function** was called to register a callback function for any attributes in the message, **MC_Send_Request_Message** calls that function to retrieve the attribute values as it sends the message.

NOTE: Some DICOM services require that values for certain "command level" message attributes (i.e. group 0 attributes) be set. (MergeCOM-3 automatically adds command level attributes to the message when **MC_Open_Message** is called. With the exceptions listed below, the caller must set any command level attribute values before sending the message. **MC_Send_Request_Message** does, however, set the following attribute values for you **- if you have not set them**:

- The group length attribute (0000,0000) value is always set by MergeCOM-3.
- If the message command requires it, the Affected SOP Class UID attribute (0000,0002) value is set to the service's abstract syntax UID.
- The command attribute (0000,0100) value is always set by MergeCOM-3.
- The Message ID attribute (0000,0110) value is set to a unique number for this association.
- If the message command requires it, the Priority attribute (0000,0700) value is set "medium" priority (i.e. zero).

- The Data Set Type attribute (0000,0800) value is always set by MergeCOM-3.

If **MC_Send_Request_Message** returns a status code of **MC_ASSOCIATION_ABORTED**, no further calls may be made for the association.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_ASSOC_ID	*AssociationID* is not a valid association object ID.
MC_INVALID_MESSAGE_ID	The *MessageID* value is not a valid message object ID.
MC_MAX_OPERATIONS_EXCEEDED	Sending of the message would cause the Maximum Number of Operations Invoked that was negotiated to be exceeded. A response message must be read before this message can be sent.
MC_REQUIRED_ATTRIBUTE_MISSING	Some DICOM services require certain command level attributes (group 0) to have values set. This status indicates that a required attribute value for the service related to *MessageID* has not been set.
MC_UNACCEPTABLE_SERVICE	The service related to *MessageID* has not been successfully negotiated for this association.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.
The association is dropped if any of the following are returned:	
MC_ASSOCIATION_ABORTED	The association has been aborted. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Register_Callback_Function
MC_Send_Request_Message
MC_Send_Response_Message
MC_Open_Message
MC_Read_Message
MC_Get_Meta_ServiceName

MC_Send_Response_Message

Sends a response to a message received from the remote application.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Send_Response_Message (
      int AssociationID,
      RESP_STATUS ResponseStatus,
      int ResponseMessageID
)
```

AssociationID	The association object's identification number
ResponseStatus	This is the status code to be returned. It must be one of the enumerated **RESP_STATUS** codes defined in "mergecom.h".
ResponseMessageID	This is the ID of a message object containing the response message.

Remarks

MC_Send_Response_Message allows the caller to respond to a message received from the remote application. **MC_Send_Response_Message** is called after receiving a request message using **MC_Read_Message**.

ResponseStatus must be set to a valid response code for the service involved. Response codes for standard DICOM services are defined in the "mergecom.h" file.

ResponseMessageID must be set to an identifier returned by **MC_Open_Message**. Many DICOM services do not require a response of more than just a status. Others, however, (e.g. C_FIND_RSP) require the setting of several message attributes.

NOTE: Some DICOM services require that values for certain "command level" message attributes (i.e. group 0 attributes) be set. (MergeCOM-3 automatically adds command level attributes to the message when **MC_Open_Message** is called. With the exceptions listed below, the caller must set any command level attribute values before sending the message. **MC_Send_Response_Message** does, however, set the following attribute values for you - **if you have not set them**:

- The group length attribute (0000,0000) value is always set by MergeCOM-3.
- If the message command requires it, the Affected SOP Class UID attribute (0000,0002) value is set to the service's abstract syntax UID.
- The command attribute (0000,0100) value is always set by MergeCOM-3.
- The Response Message ID attribute (0000,0120) value is set to the Message ID value of the last received Request message.
- The Data Set Type attribute (0000,0800) value is always set by MergeCOM-3.

If **MC_Send_Response_Message** returns **MC_ASSOCIATION_ABORTED**, no further calls may be made for that association.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning

MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_ASSOC_ID	*AssociationID* is not a valid association object ID.
MC_INVALID_MESSAGE_ID	The *MessageID* value is not a valid message object ID.
MC_REQUIRED_ATTRIBUTE_MISSING	Some DICOM services require certain command level attributes (group 0) to have values set. This status indicates that a required attribute value for the service related to *MessageID* has not been set.
MC_UNACCEPTABLE_SERVICE	The service related to *MessageID* has not been successfully negotiated for this association.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

The association is dropped if any of the following are returned:

MC_ASSOCIATION_ABORTED	The association has been aborted. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Send_Request_Message **MC_Read_Message**

MC_Set_Bool_Config_Value

Used to set the value of a boolean tool kit configuration parameter at runtime.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Set_Bool_Config_Value (
      BoolParm Aparm, int Avalue
)
```

Aparm An enumerated constant identifying the boolean configuration parameter to be (re)set. Aparm can have any of the following values:

LOG_FILE_BACKUP,
ACCEPT_ANY_CONTEXT_NAME,
ACCEPT_ANY_APPLICATION_TITLE,
ACCEPT_ANY_PROTOCOL_VERSION,
ACCEPT_DIFFERENT_IC_UID,
ACCEPT_DIFFERENT_VERSION,
AUTO_ECHO_SUPPORT,
SEND_SOP_CLASS_UID,
SEND_SOP_INSTANCE_UID,
SEND_LENGTH_TO_END,
SEND_RECOGNITION_CODE,
SEND_MSG_ID_RESPONSE,
SEND_ECHO_PRIORITY,
SEND_RESPONSE_PRIORITY,
INSURE_EVEN_UID_LENGTH,
BLANK_FILL_LOG_FILE,
ELIMINATE_ITEM_REFERENCES,
HARD_CLOSE_TCP_IP_CONNECTION,
FORCE_DICM_IN_PREFIX,
ACCEPT_ANY_PRESENTATION_CONTEXT,
ACCEPT_ANY_HOSTNAME,
ALLOW_OUT_OF_ORDER_TAGS,
EMPTY_PRIVATE_CREATOR_CODES,
FORCE_OPEN_EMPTY_ITEM,
FORCE_JAVA_BIG_ENDIAN,
ACCEPT_MULTIPLE_PRES_CONTEXTS,
ALLOW_INVALID_PRIVATE_CREATOR_CODES,
REMOVE_PADDING_CHARS,
PRIVATE_SYNTAX_1_LITTLE_ENDIAN,
PRIVATE_SYNTAX_1_EXPLICIT_VR,
PRIVATE_SYNTAX_1_ENCAPSULATED,
PRIVATE_SYNTAX_2_LITTLE_ENDIAN,
PRIVATE_SYNTAX_2_EXPLICIT_VR,
PRIVATE_SYNTAX_2_ENCAPSULATED,
EXPORT_UN_VR_TO_MEDIA,
EXPORT_UN_VR_TO_NETWORK,
EXPORT_PRIVATE_ATTRIBUTES_TO_MEDIA,
EXPORT_PRIVATE_ATTRIBUTES_TO_NETWORK,
ALLOW_INVALID_PRIVATE_ATTRIBUTES,
RETURN_COMMA_IN_DS_FL_FD_STRINGS,
ALLOW_COMMA_IN_DS_FL_FD_STRINGS,
DEFLATE_ALLOW_FLUSH,
COMPRESSION_ALLOW_FRAGS,

NETWORK_CAPTURE,
DUPLICATE_ENCAPSULATED_ICON,
COMPRESSION_WHEN_J2K_USE_LOSSY,
COMPRESSION_J2K_LOSSY_USE_QUALITY,
TCPIP_DISABLE_NAGLE,
COMBINE_DATA_WITH_HEADER,
REWRITE_CAPTURE_FILES,
MSG_FILE_ITEM_OBJ_TRACE,
EXPORT_UNDEFINED_LENGTH_SQ,
EXPORT_UNDEFINED_LENGTH_SQ_IN_DICOMDIR,
EXPORT_GROUP_LENGTHS_TO_NETWORK,
EXPORT_GROUP_LENGTHS_TO_MEDIA

These names are the same as those given to the parameters in the tool kit configuration files.

Avalue The boolean value to which Aparm is to be set.

Remarks

The MergeCOM-3 Advanced Library accesses several configuration files at startup. This call allows your application to (re)set boolean configurable parameters specified in these files at runtime. This call should be made immediately after calling **MC_Library_Initialization** to avoid using these parameters before they are set. Please see the detailed description of tool kit configuration elsewhere in this manual.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_PARAMETER_NAME	Parameter specified is not one of the above listed parameters.
MC_LIBRARY_NOT_INITIALIZED	The library has not been properly initialized.

See Also

MC_Get_Int_Config_Value
MC_Get_Log_Destination
MC_Get_Long_Config_Value
MC_Get_String_Config_Value
MC_Set_Int_Config_Value
MC_Set_Log_Destination
MC_Set_Long_Config_Value
MC_Set_String_Config_Value

MC_Set_Encapsulated_Value_From_Function

Encapsulates a single frame and stores the encapsulated value into the message object for the given attribute. The value representation of the attribute must be OB or OW. If a compressor is registered, the data will be compressed in the message.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Set_Encapsulated_Value_From_Function (
      int MsgFileItemID,
      unsigned long Tag,
      void* UserInfo,
      MC_STATUS (*YourSetFunction)()
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
Tag	DICOM tag which identifies the attribute.
UserInfo	Address of user data which will be passed on to *YourSetFunction* each time it is called. This may be NULL.
YourSetFunction	Name of a function which will be called repeatedly for blocks of data for the attribute's value.

The function must be prototyped as follows:

```
MC_STATUS YourSetFunction (
      int CBMsgFileItemID,
      unsigned long CBtag,
      int CBisFirst,
      void* CBuserInfo,
      int* CBdataSizePtr,
      void** CBdataBufferPtr,
      int* CBisLastPtr
)
```

CBMsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
Cbtag	DICOM tag which identifies the attribute.
CBisFirst	This is TRUE (non-zero) the first time MergeCOM-3 calls *YourSetFunction* to request data blocks.
CBuserInfo	Address of data which is being passed from the **MC_Set_Value_From_Function** function. This may be NULL.
CBdataSizePtr	Set **CBdataSizePtr* to the number of bytes you are providing.
CBdataBufferPtr	Set * *CBdataBufferPtr* to the address of the data you are providing.
CBisLastPtr	Set **CBisLastPtr* to TRUE (not zero) when you are returning with the last block of OBOW data.

Remarks

The **MC_Set_Encapsulated_Value_From_Function** function is used to encapsulate, and then set the value of an attribute which has a value representation of OB or OW. If a

compressor is registered with **MC_Register_Compression_Callbacks**, the data will be compressed using the registered compressor. Such attributes tend to have values of great length. To accommodate this, one uses the **MC_Set_Encapsulated_Value_From_Function** function to specify the name of a function (***YourSetFunction***) which MergeCOM-3, in turn, calls repeatedly requesting blocks of the attribute's data value.

MC_Close_Encapsulated_Value must be called when the user is through encapsulating data within this message. If more frames will be encapsulated, use **MC_Set_Next_Encapsulated_Value_From_Function** until all frames have been encapsulated, followed by **MC_Close_Encapsulated_Value**.

An optional *UserInfo* parameter may be used to pass information between the **MC_Set_Encapsulated_Value_From_Function** caller and ***YourSetFunction*** which receives the data in its *CBuserInfo* parameter. If no such exchange of data is necessary, you may use NULL for *UserInfo*.

YourSetFunction

YourSetFunction must set **CBdataBufferPtr* to the address of a block of data which is a portion of the attribute's value. It also must set **CBdataSizePtr* to the number of bytes in the block.

MergeCOM-3 sets *CBisFirst* to TRUE (non-zero) if it is requesting the first block of the attribute's value. ***YourSetFunction*** must set **CBisLastPtr* to TRUE (non-zero) if it is supplying the last block of the attribute's value.

YourSetFunction must return **MC_NORMAL_COMPLETION** if all went well. If not, it must return **MC_CANNOT_COMPLY**. Both of these are defined in "mc3msg.h".

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_TRANSFER_SYNTAX	The message's transfer syntax is non-encapsulated.
MC_NULL_POINTER_PARM	One or more of the pointer-type parameters was NULL.
MC_INVALID_TAG	The message does not contain an attribute with an ID of *Tag*. The attribute can be added with **MC_Add_Standard_Attribute**. NOTE: This status will not be returned if setting the value of a message object which was opened using **MC_Open_Empty_Message** or **MC_Create_Empty_File**. In that case, the attribute is automatically added to the object before setting the value.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.
MC_INCOMPATIBLE_VR	The attribute's value representation was not OB or OW.
MC_CALLBACK_DATA_SIZE_UNEVEN	The value set by ***YourSetFunction*** in its

	**CBdataSizePtr* parameter was not an even number.
MC_CALLBACK_PARM_ERROR	The value set by ***YourSetFunction*** in its **CBdataSizePtr* parameter was an odd number
MC_CALLBACK_CANNOT_COMPLY	***YourSetFunction*** returned with **MC_CANNOT_COMPLY.**

See Also

MC_Set_Next_Encapsulated_Value_From_Function
MC_Close_Encapsulated_Value
MC_Register_Compression_Callbacks

MC_Set_File_Preamble

Sets the preamble for a file object

Synopsis

```
#include "mc3media.h"

MC_STATUS MC_Set_File_Preamble(
      int FileID,
      char* Preamble
)
```

FileID	The identifier assigned to this object by the **MC_Create_Empty_File** or **MC_Create_File** function.
Preamble	A pointer to the 128 byte preamble to be associated with the file object *FileID*.

Remarks

MC_Set_File_Preamble changes the preamble associated with a file object. The function copies the 128 byte DICOM file preamble pointed to by *Preamble* into the file object *FileID*. This preamble is written to media when the **MC_Write_File** function is called.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_FILE_ID	The *FileID* value is not a valid file object ID.
MC_NULL_POINTER_PARM	*Preamble* was NULL.

See Also

MC_Get_File_Preamble

MC_Set_Int_Config_Value

Used to set the value of a integer tool kit configuration parameter at runtime.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Set_Int_Config_Value (
      IntParm Aparm,
      int Avalue
)
```

Aparm An enumerated constant identifying the integer configuration parameter to be (re)set. Aparm can have any of the following values:

LOG_FILE_SIZE,
LOG_MEMORY_SIZE,
ARTIM_TIMEOUT,
ASSOC_REPLY_TIMEOUT,
RELEASE_TIMEOUT,
WRITE_TIMEOUT,
CONNECT_TIMEOUT,
INACTIVITY_TIMEOUT,
LARGE_DATA_SIZE,
DESIRED_LAST_PDU_SIZE,
OBOW_BUFFER_SIZE,
FLATE_GROW_OUTPUT_BUF_SIZE,
DEFLATE_COMPRESSION_LEVEL,
COMPRESSION_LUM_FACTOR,
COMPRESSION_CHROM_FACTOR,
COMPRESSION_J2K_LOSSY_RATIO,
COMPRESSION_J2K_LOSSY_QUALITY,
WORK_BUFFER_SIZE,
TCPIP_LISTEN_PORT,
MAX_PENDING_CONNECTIONS,
TCPIP_SEND_BUFFER_SIZE,
TCPIP_RECEIVE_BUFFER_SIZE,
NUM_HISTORICAL_LOG_FILES,
LOG_FILE_LINE_LENGTH,
NUMBER_OF_CAP_FILES,
IGNORE_JPEG_BAD_SUFFIX

These names are the same as those given to the parameters in the tool kit configuration files. A description of the options can be found in Appendix B.

Avalue The integer value to which Aparm is to be set.

Remarks

The MergeCOM-3 Advanced Tool Kit Library accesses several configuration files at startup. This call allows your application to (re)set integer configurable parameters specified in these files at runtime. This call should be made immediately after calling **MC_Library_Initialization** to avoid using these parameters before they are set. Please see the detailed description of tool kit configuration elsewhere in this manual.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_PARAMETER_NAME	Parameter specified is not one of the above listed parameters.
MC_MUST_BE_POSITIVE	Parameter value cannot be negative.
MC_LIBRARY_NOT_INITIALIZED	The library has not been properly initialized.

See Also

MC_Get_Int_Config_Value
MC_Get_Log_Destination
MC_Get_Long_Config_Value
MC_Get_String_Config_Value
MC_Set_Log_Destination
MC_Set_Long_Config_Value
MC_Set_String_Config_Value

MC_Set_Log_Destination

Determines where log messages will be written.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Set_Log_Destination (
      LogParm Aparm,
      int Avalue
)
```

Aparm — An enumerated constant identifying the class of logging that is to be redirected. Aparm can have any of the following values:

ERROR_DESTINATIONS,
WARNING_DESTINATIONS,
INFO_DESTINATIONS,
T1_DESTINATIONS,
T2_DESTINATIONS,
T3_DESTINATIONS,
T4_DESTINATIONS,
T5_DESTINATIONS,
T6_DESTINATIONS,
T7_DESTINATIONS,
T8_DESTINATIONS
T9_DESTINATIONS

Avalue — An defined term identifying where the logging is to be directed. Avalue can have any of the following values:

File_Destination,
Memory_Destination,
Screen_Destination,
Bitbucket_Destination

These values can also be OR'ed together to indicate multiple destinations.

Remarks

This call allows you to redirect the logging of error, warning, and info messages at runtime. The [DEFAULT_LIBRARY] section of the MergeCOM-3 initialization file contains the setting used at startup. This call should be made immediately after calling **MC_Library_Initialization** to avoid using these parameters before they are set. Please see the detailed description of tool kit configuration elsewhere in this manual.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_PARAMETER_NAME	Parameter specified is not one of the above listed parameters.
MC_LIBRARY_NOT_INITIALIZED	The library has not been properly initialized.

See Also

MC_Get_Int_Config_Value
MC_Set_Int_Config_Value
MC_Get_Log_Destination
MC_Get_Long_Config_Value
MC_Set_Long_Config_Value
MC_Get_String_Config_Value
MC_Set_String_Config_Value

MC_Set_Log_Prefix

Used to set a string to be placed at the beginning of log statements.

Synopsis

```
#include “mc3msg.h”

void MC_Set_Log_Prefix (
      char* Prefix
)
```

Prefix A prefix to place in front of all logs

Remarks

This call allows you to place a string in front of all log statements generated by the tool kit.

Return Value

None

See Also

MC_Get_Log_Destination **MC_Set_Log_Destination**

MC_Set_Long_Config_Value

Used to set the value of a long integer tool kit configuration parameter at runtime.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Set_Long_Config_Value (
      LongParm Aparm,
      long int Avalue
)
```

Aparm — An enumerated constant identifying the long integer configuration parameter to be (re)set. Aparm can have only the following values:

PDU_MAXIMUM_LENGTH,
CALLBACK_MIN_DATA_SIZE,
CAPTURE_FILE_SIZE

This name is the same as those given to the parameters in the tool kit configuration files.

Avalue — The long integer value to which Aparm is to be set.

Remarks

The MergeCOM-3 Advanced Tool Kit Library accesses several configuration files at startup. This call allows your application to (re)set long integer configurable parameters specified in these files at runtime. This call should be made immediately after calling **MC_Library_Initialization** to avoid using this parameter before is set. Please see the detailed description of tool kit configuration elsewhere in this manual.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_PARAMETER_NAME	Parameter specified is not one of the above listed parameters.
MC_MUST_BE_POSITIVE	Parameter value cannot be negative.
MC_LIBRARY_NOT_INITIALIZED	The library has not been properly initialized.

See Also

MC_Get_Int_Config_Value
MC_Set_Int_Config_Value
MC_Get_Log_Destination
MC_Get_Long_Config_Value
MC_Set_Long_Config_Value
MC_Get_String_Config_Value
MC_Set_String_Config_Value

MC_Set_MergeINI

Set the path of the merge.ini file so that the MERGE_INI environment variable is not used.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Set_MergeINI (
      char* Filename
)
```

Filename An absolute or relative filename containing the location of the merge.ini file.

Remarks

The MergeCOM-3 Advanced Tool Kit Library accesses several configuration files at startup. The `MERGE_INI` environment variable is used to point to the main configuration file (`merge.ini`). **MC_Set_MergeINI** can be used to override the use of this environment variable. *Filename* contains an absolute or relative filename where the merge.ini file is located. This call should be made immediately before calling **MC_Library_Initialization**.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.
MC_LIBRARY_ALREADY_INITIALIZED	The library has already been initialized.

See Also

MC_Library_Initialization
MC_Library_Release
MC_Library_Reset

MC_Set_Message_Callbacks

Associates registered callback functions with a message or file

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Set_Message_Callbacks (
      int ApplicationID,
      int MessageFileID,
)
```

ApplicationID	The identifier assigned to this application by the **MC_Register_Application** function.
MessageFileID	The identifier assigned to this object by the **MC_Open_Message** or **MC_Create_File** functions.

Remarks

MC_Set_Message_Callbacks associates **the callback functions registered for an application with a message or file object. When this function is not used, the callback function is first associated with a message or file when it is transmitted or received on a DICOM association or read or written to a DICOM file.**

When **MC_Set_Message_Callbacks** is used, subsequent calls to **MC_Get_Value_To_Function** or **MC_Set_Value_From_Function** will also call the callback function to store an attribute's values.

See MC_Register_Callback_Function for further details on using callback functions to manage OB, OW, or OF data.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_APPLICATION_ID	*ApplicationID* does not identify a valid application object.
MC_INVALID_MESSAGE_ID	*MessageFileID* is not a valid message or file object ID.
MC_LIBRARY_NOT_INITIALIZED	The library has not been properly initialized.

See Also

MC_Set_Value_From_Function
MC_Get_Value_To_Function
MC_Register_Callback_Function
MC_Release_Callback_Function

MC_Set_Message_Transfer_Syntax

Sets the transfer syntax over which a message will be sent on the network.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Set_Message_Transfer_Syntax (
      int MessageID,
      TRANSFER_SYNTAX TransferSyntax
)
```

MessageID	The identifier assigned to this object by the **MC_Open_Message** or **MC_Open_Empty_Message** function.
TransferSyntax	The name of the transfer syntax to associate with the message.

Remarks

MC_Set_Message_Transfer_Syntax associates a DICOM transfer syntax with a message. This function is only of use for SCU applications that negotiate more than one transfer syntax for a service. This call is used in conjunction with **MC_Send_Request_Message** to specify the transfer syntax to use when sending the message.

Reference the *MergeCOM-3 Advanced Tool Kit User's Manual* for a discussion on negotiating multiple transfer syntaxes for a service.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_MESSAGE_ID	*MessageID* is not a valid message object ID.

See Also

MC_Send_Request_Message **MC_Get_Message_Transfer_Syntax**

MC_Set_Negotiation_Info

Registers extended negotiation information.

NOTE: Use of this call is deprecated. **MC_Set_Negotiation_Info_For_Association** and setting of negotiation information through service lists should be used in place of this call and **MC_Clear_Negotiation_Info**.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Set_Negotiation_Info (
      int ApplicationID,
      char* ServiceName,
      void* ExtInfoBuffer,
      int ExtInfoLength
)
```

ApplicationID	The identification number for the registered application.
ServiceName	The name given to a valid DICOM service.
ExtInfoBuffer	A pointer to the buffer containing extended negotiation information.
ExtInfoLength	The number of bytes contained in the *ExtInfoBuffer*. This must be an even number.

Remarks

The DICOM standard allows application entities to exchange "extended negotiation information" when establishing an association. The contents of the negotiation information must be known to both the association requestor application and the association acceptor application. Such extended negotiation is not often used for DICOM services, but some services may require it.

MC_Set_Negotiation_Info allows the caller to supply MergeCOM-3 with extended negotiation information which it will use when establishing associations.

When a **MC_Open_Association** or **MC_Open_Secure_Association** call is made, MergeCOM-3 sends any registered extended negotiation information to the association acceptor. The acceptor normally returns an updated version of the negotiation information which can be accessed using the **MC_Get_Negotiation_Info** call.

When a **MC_Wait_For_Association** or **MC_Wait_For_Secure_Association** call is made, MergeCOM-3 stores any received extended negotiation information. This information may be accessed using the **MC_Get_Negotiation_Info** call. **MC_Set_Negotiation_Info** may be used to "update" the extended negotiation information before calling **MC_Accept_Association** to accept the association. MergeCOM-3 will return any registered negotiation information to the remote application.

Use **MC_Clear_Negotiation_Info** to remove extended information registered for a service.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.

MC_INVALID_APPLICATION_ID	*ApplicationID* is not a valid application identifier.
MC_NULL_POINTER_PARM	Either *ServiceName* or *ExtInfoBuffer* was NULL.
MC_UNKNOWN_SERVICE	*ServiceName* was not registered in the MergeCOM-3 configuration files.
MC_LIBRARY_NOT_INITIALIZED	The library has not been properly initialized.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Get_Negotiation_Info
MC_Clear_Negotiation_Info
MC_Set_Negotiation_Info_For_Association

MC_Set_Negotiation_Info_For_Association

Sets extended negotiation information before acceptance of an association.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Set_Negotiation_Info_For_Association (
      int AssociationID,
      char* ServiceName,
      void* ExtInfoBuffer,
      int ExtInfoLength
)
```

AssociationID	An association identification number retruend by an **MC_Wait_For_Association** or **MC_Wait_For_Secure_Association** call.
ServiceName	The name given to a valid DICOM service.
ExtInfoBuffer	A pointer to the buffer containing extended negotiation information.
ExtInfoLength	The number of bytes contained in the *ExtInfoBuffer*.

Remarks

The DICOM standard allows application entities to exchange "extended negotiation information" when establishing an association. The contents of the negotiation information must be known to both the association requestor application and the association acceptor application. Such extended negotiation is not often used for DICOM services, but some services may require it.

MC_Set_Negotiation_Info_For_Association allows the caller to supply MergeCOM-3 with extended negotiation information which it will use when establishing associations. When a **MC_Wait_For_Association** or **MC_Wait_For_Secure_Association** call is made, MergeCOM-3 stores any received extended negotiation information. This information may be accessed using the **MC_Get_Negotiation_Info** call. **MC_Set_Negotiation_Info_For_Association** may then be used to set extended negotiation information in the association response. After setting the extended negotiation information to return, **MC_Accept_Association** would be called to accept the association. If **MC_Set_Negotiation_Info_For_Association** is not used, no extended negotiation information will be returned by **MC_Accept_Association** in the response.

Note that if no negotiation information was included in the association request, **MC_Set_Negotiation_Info_For_Association** will not set the negotiation information in the association response.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_ASSOC_ID	*AssociationID* is not a valid association identifier.
MC_NO_REQUEST_PENDING	*AssociationID* references to an association that has already been accepted.
MC_NULL_POINTER_PARM	Either *ServiceName* or *ExtInfoBuffer* was NULL.

MC_UNKNOWN_SERVICE	*ServiceName* was not registered in the MergeCOM-3 configuration files or was not negotiated for the association.
MC_LIBRARY_NOT_INITIALIZED	The library has not been properly initialized.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Get_Negotiation_Info
MC_Clear_Negotiation_Info

MC_Set_Next_Encapsulated_Value_From_Function

Encapsulates the next frame and stores the encapsulated value into the message object for the given attribute. The value representation of the attribute must be OB or OW. If a compressor is registered, the data will be compressed in the message.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Set_Next_Encapsulated_Value_From_Function (
      int MsgFileItemID,
      unsigned long Tag,
      void* UserInfo,
      MC_STATUS (*YourSetFunction)()
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
Tag	DICOM tag which identifies the attribute.
UserInfo	Address of user data which will be passed on to *YourSetFunction* each time it is called. This may be NULL.
YourSetFunction	Name of a function which will be called repeatedly for blocks of data for the attribute's value.

The function must be prototyped as follows:

```
MC_STATUS YourSetFunction (
      int CBMsgFileItemID,
      unsigned long CBtag,
      int CBisFirst,
      void* CBuserInfo,
      int* CBdataSizePtr,
      void** CBdataBufferPtr,
      int* CBisLastPtr
)
```

CBMsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
Cbtag	DICOM tag which identifies the attribute.
CBisFirst	This is TRUE (non-zero) the first time MergeCOM-3 calls *YourSetFunction* to request data blocks.
CBuserInfo	Address of data which is being passed from the **MC_Set_Value_From_Function** function. This may be NULL.
CBdataSizePtr	Set **CBdataSizePtr* to the number of bytes you are providing.
CBdataBufferPtr	Set * *CBdataBufferPtr* to the address of the data you are providing.
CBisLastPtr	Set **CBisLastPtr* to TRUE (not zero) when you are returning with the last block of OBOW data.

Remarks

The **MC_Set_Next_Encapsulated_Value_From_Function** function is used to encapsulate the next frame, and then set the value of an attribute which has a value representation of OB or OW. If a compressor is registered with **MC_Register_Compression_Callbacks**, the data will be compressed using the registered compressor. Such attributes tend to have values of great length. To accommodate this, one uses the **MC_Set_Next_Encapsulated_Value_From_Function** function to specify the name of a function (***YourSetFunction***) which MergeCOM-3, in turn, calls repeatedly requesting blocks of the attribute's data value.

MC_Close_Encapsulated_Value must be called when the user is through encapsulating data within this message. If more frames will be encapsulated, use **MC_Set_Next_Encapsulated_Value_From_Function** until all frames have been encapsulated, followed by **MC_Close_Encapsulated_Value**.

An optional *UserInfo* parameter may be used to pass information between the **MC_Set_Next_Encapsulated_Value_From_Function** caller and ***YourSetFunction*** which receives the data in its *CBuserInfo* parameter. If no such exchange of data is necessary, you may use NULL for *UserInfo*.

YourSetFunction

YourSetFunction must set **CBdataBufferPtr* to the address of a block of data which is a portion of the attribute's value. It also must set **CBdataSizePtr* to the number of bytes in the block.

MergeCOM-3 sets *CBisFirst* to TRUE (non-zero) if it is requesting the first block of the attribute's value. ***YourSetFunction*** must set **CBisLastPtr* to TRUE (non-zero) if it is supplying the last block of the attribute's value.

YourSetFunction must return **MC_NORMAL_COMPLETION** if all went well. If not, it must return **MC_CANNOT_COMPLY**. Both of these are defined in "mc3msg.h".

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_TRANSFER_SYNTAX	The message's transfer syntax is non-encapsulated.
MC_NULL_POINTER_PARM	One or more of the pointer-type parameters was NULL.
MC_INVALID_TAG	The message does not contain an attribute with an ID of *Tag*. The attribute can be added with **MC_Add_Standard_Attribute**. NOTE: This status will not be returned if setting the value of a message object which was opened using **MC_Open_Empty_Message** or **MC_Create_Empty_File**. In that case, the attribute is automatically added to the object before setting the value.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.

MC_INCOMPATIBLE_VR	The attribute's value representation was not OB or OW.
MC_CALLBACK_DATA_SIZE_UNEVEN	The value set by ***YourSetFunction*** in its **CBdataSizePtr* parameter was not an even number.
MC_CALLBACK_PARM_ERROR	The value set by ***YourSetFunction*** in its **CBdataSizePtr* parameter was an odd number
MC_CALLBACK_CANNOT_COMPLY	***YourSetFunction*** returned with **MC_CANNOT_COMPLY.**

See Also

MC_Set_Encapsulated_Value_From_Function
MC_Close_Encapsulated_Value
MC_Register_Compression_Callbacks

MC_Set_Next_pValue... Functions

Appends another value to a private attribute in a message object.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Set_Next_pValue (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      MC_DT DataType, void* Value
)
MC_STATUS MC_Set_Next_pValue_From_Double (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      double Value
)
MC_STATUS MC_Set_Next_pValue_From_Float (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      float Value
)
MC_STATUS MC_Set_Next_pValue_From_Int (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      int Value
)
MC_STATUS MC_Set_Next_pValue_From_ShortInt (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      short int Value
)
MC_STATUS MC_Set_Next_pValue_From_LongInt (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      long int Value
)
MC_STATUS MC_Set_Next_pValue_From_String (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      char* Value
)
MC_STATUS MC_Set_Next_pValue_From_UInt (
      int MsgFileItemID,
```

```
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      unsigned int Value
)
MC_STATUS MC_Set_Next_pValue_From_UShortInt (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      unsigned short Value
)
MC_STATUS MC_Set_Next_pValue_From_ULongInt (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      unsigned long Value
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
PrivateCode	The code string which identifies which block in the private Group "owns" the attribute.
Group	The number identifying the private group containing the private attribute. It must be an odd number.
ElementByte	The number identifying the private attribute within the private Group for this PrivateCode.
DataType	One of the enumerated codes identifying the data type of the value in Value. The **MC_DT** enumerated type is defined in "mc3msg.h". They are:

String_Type	Null-terminated character string
Int_Type	Binary integer number
UInt_Type	Binary unsigned integer number
ShortInt_Type	Binary short integer number
UShortInt_Type	Binary unsigned short integer number
LongInt_Type	Binary long integer number
ULongInt_Type	Binary unsigned long integer number
Float_Type	Binary Floating point number

Value	The attribute should be set to this value.

Remarks

These functions add another value for an attribute identified by *ElementByte* for the *PrivateCode* within the given *Group*. If no values exist yet for the attribute, *Value* is stored as the attribute's first value. If **MC_Set_Next_pValue** is used, the data type of the *Value* is specified by the *DataType* parameter. The other function names imply the data type. For example **MC_Set_Next_pValue_From_Int** is the same as calling **MC_Set_Next_pValue** with *DataType* specified as **Int_Type**. Each function will assign the value at *Value*, which must be prototyped as the appropriate type.

NOTE: string values are NULL-terminated character arrays.

Any reasonable conversion will be made from *Value*'s data type to the attribute's value representation. However, some conversions are illogical (see the table below). An appropriate error code will be returned if the conversion attempt failed.

NOTE: attributes with a value representation of **SQ** (sequence of items) *Value* must represent an integer which is the *ItemID* of an item object opened previously using **MC_Open_Item**.

	Function may be used to set values for attributes with these Value Representations
MC_Set_Next_pValue_From_Int	**DS, FD, FL, IS, SL, SS, UL, US, SQ**
MC_Set_Next_pValue_From_UInt	**DS, FD, FL, IS, SL, SS, UL, US, SQ**
MC_Set_Next_pValue_From_ShortInt	**DS, FD, FL, IS, SL, SS, UL, US, SQ**
MC_Set_Next_pValue_From_UShortInt	**DS, FD, FL, IS, SL, SS, UL, US, SQ**
MC_Set_Next_pValue_From_LongInt	**AT, DS, FD, FL, IS, SL, SS, UL, US, SQ**
MC_Set_Next_pValue_From_ULongInt	**AT, DS, FD, FL, IS, SL, SS, UL, US, SQ**
MC_Set_Next_pValue_From_Float	**DS, FD, FL, IS, SL, SS, UL, US, SQ**
MC_Set_Next_pValue_From_Double	**DS, FD, FL, IS, SL, SS, UL, US, SQ**
MC_Set_Next_pValue_From_String	**AS, AT, CS, DA, DS, DT, FD, FL, IS, LO, LT, PN, SH, SL, SS, ST, TM, UI, UL, US, SQ**

The same rules apply to the **MC_Set_Next_pValue** function, based on the value used in the *DataType* parameter.

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

	Value	Meaning
INFORMATIONAL MESSAGES:	**MC_NORMAL_COMPLETION**	The function completed normally.
WARNINGS: (VALUE WAS STILL ENCODED)	**MC_INVALID_CHARS_IN_VALUE**	A string-type *Value* had invalid characters for the value representation of *Tag*.
	MC_INVALID_VALUE_FOR_VR	*Value* does not conform to the requirements of its value representation.
ERRORS:	**MC_NULL_POINTER_PARM**	One or more of the pointer-type parameters was NULL.
	MC_INVALID_PRIVATE_CODE	The message does not contain an attributes in *Group* for *PrivateCode*.
	MC_INVALID_GROUP	*Group* was not an odd number.
	MC_INVALID_TAG	The message does not contain an attribute identified by *ElementByte* in *Group* for *PrivateCode*.
	MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.
	MC_INCOMPATIBLE_VR	The attribute's value representation cannot be derived from *Value*. See the table above.

MC_VALUE_OUT_OF_RANGE	A numeric *Value* was larger than could be accommodated by the attribute. (E.g. setting an attribute with an unsigned int VR to the value 123.45, or setting it to 128000, or setting it to -1).
MC_TEMP_FILE_ERROR	If the attribute's value is large, it may be stored in a temporary file (if so configured). If a file I/O error occurs, this status is returned.
MC_INVALID_DATA_TYPE	*DataType* is not valid.
MC_TOO_MANY_VALUES	This attribute contains the most values that a standard attribute can contain (65535).
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Set_Value... Functions
MC_Set_Value_From_Function
MC_Set_Value_To_NULL
MC_Set_Value_To_Empty
MC_Set_pValue... Functions
MC_Set_pValue_From_Function
MC_Set_pValue_To_NULL
MC_Set_pValue_To_Empty
MC_Set_Next_Value... Functions
MC_Stream_To_Message

MC_Set_Next_pValue_To_NULL

Sets the next value of a private attribute in a message object to NULL.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Set_pValue_To_NULL (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message** function or the **MC_Open_Item** function.
PrivateCode	The code string which identifies which block in the private *Group* "owns" the attribute.
Group	The number identifying the private group containing the private attribute. It must be an odd number.
ElementByte	The number identifying the private attribute within the private *Group* for this *PrivateCode*.

Remarks

MC_Set_Next_pValue_To_NULL adds a NULL to the next value of a private multi-valued text type attribute identified by *PrivateCode*, *Group* and *ElementByte* in the message identified by *MsgFileItemID*. Any existing values remain in the attribute. Note that the function can only be used on attributes which 1) allow the backslash (\) character as a field delimiter and 2) are of a text type. It can not be used on numerical attributes.

If a callback function was registered for the attribute, it is "de-registered."

NOTE: if the attribute has a value representation of SQ (sequence of items), setting the value to null does NOT free the item object identified by the value.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INCOMPATIBLE_VR	The attribute identified by *Tag* can not be set to NULL.
MC_INVALID_PRIVATE_CODE	The message does not contain an attributes in *Group* for *PrivateCode.*
MC_INVALID_TAG	The message does not contain an attribute identified by *Tag* . NOTE: This status will not be returned if setting the value of a message object which was opened using **MC_Open_Empty_Message** or **MC_Create_Empty_File**. In that case, the attribute is automatically added to the object before setting the value to NULL.

MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.
MC_LIBRARY_NOT_INITIALIZED	The library has not been properly initialized.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Set_Next_Value_To_NULL

MC_Set_Next_Value... Functions

Appends a value to an attribute in a message object.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Set_Next_Value (
      int MsgFileItemID,
      unsigned long Tag,
      MC_DT DataType,
      void* Value
)
MC_STATUS MC_Set_Next_Value_From_Double (
      int MsgFileItemID,
      unsigned long Tag,
      double Value
)
MC_STATUS MC_Set_Next_Value_From_Float (
      int MsgFileItemID,
      unsigned long Tag,
      float Value
)
MC_STATUS MC_Set_Next_Value_From_Int (
      int MsgFileItemID,
      unsigned long Tag,
      int Value
)
MC_STATUS MC_Set_Next_Value_From_ShortInt (
      int MsgFileItemID,
      unsigned long Tag,
      short int Value
)
MC_STATUS MC_Set_Next_Value_From_LongInt (
      int MsgFileItemID,
      unsigned long Tag,
      long int Value
)
MC_STATUS MC_Set_Next_Value_From_String (
      int MsgFileItemID,
      unsigned long Tag,
      char* Value
)
MC_STATUS MC_Set_Next_Value_From_UInt (
      int MsgFileItemID,
      unsigned long Tag,
      unsigned int Value
)
MC_STATUS MC_Set_Next_Value_From_UShortInt (
      int MsgFileItemID,
      unsigned long Tag,
      unsigned short Value
)
MC_STATUS MC_Set_Next_Value_From_ULongInt (
      int MsgFileItemID,
      unsigned long Tag,
      unsigned long Value
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
Tag	DICOM tag which identifies the attribute.
DataType	One of the enumerated codes identifying the data type of the value in *Value*. The MC_DT enumerated type is defined in "mc3msg.h". They are:

String_Type	Null-terminated character string
Int_Type	Binary integer number
UInt_Type	Binary unsigned integer number
ShortInt_Type	Binary short integer number
UShortInt_Type	Binary unsigned short integer number
LongInt_Type	Binary long integer number
ULongInt_Type	Binary unsigned long integer number
Float_Type	Binary Floating point number

Value	The attribute will be set to this value.

Remarks

These functions add another value for the attribute with the given *Tag*. If no values exist yet for the attribute, *Value* is stored as the attribute's first value. If **MC_Set_Next_Value** is used, the data type of the *Value* is specified by the *DataType* parameter. The other function names imply the data type. For example **MC_Set_Next_Value_From_Int** is the same as calling **MC_Set_Next_Value** with *DataType* specified as **Int_Type**. Each function will set the value at *Value*, which must be prototyped as the appropriate type. Note that string values are NULL-terminated character arrays.

Any reasonable conversion will be made from *Value's* data type to the attribute's value representation. However, some conversions are illogical (see the table below). An appropriate error code will be returned if the conversion attempt failed.

NOTE: for attributes with a value representation of **SQ** (sequence of items) *Value* must represent an integer which is the *ItemID* of an item object opened previously using **MC_Open_Item**.

Functions	**Value Representations** to use for setting values for attributes
MC_Set_Next_Value_From_Int	DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Set_Next_Value_From_UInt	DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Set_Next_Value_From_ShortInt	DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Set_Next_Value_From_UShortInt	DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Set_Next_Value_From_LongInt	AT, DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Set_Next_Value_From_ULongInt	AT, DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Set_Next_Value_From_Float	DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Set_Next_Value_From_Double	DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Set_Next_Value_From_String	AS, AT, CS, DA, DS, DT, FD, FL, IS, LO, LT, PN, SH, SL, SS, ST, TM, UI, UL, US, SQ

The same rules apply to the **MC_Set_Next_Value** function, based on the value used in the *DataType* parameter.

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

	Value	Meaning
INFORMATIONAL MESSAGES:	**MC_NORMAL_COMPLETION**	The function completed normally.
WARNINGS: (VALUE WAS STILL ENCODED)	**MC_INVALID_CHARS_IN_VALUE**	A string-type *Value* had invalid characters for the value representation of *Tag*.
	MC_INVALID_VALUE_FOR_VR	*Value* does not conform to the requirements of its value representation.
ERRORS:	**MC_NULL_POINTER_PARM**	One or more of the pointer-type parameters was NULL.
	MC_INVALID_TAG	The message does not contain an attribute with an ID of *Tag*. The attribute can be added with **MC_Add_Standard_Attribute**. NOTE: This status will not be returned if setting the value of a message object which was opened using **MC_Open_Empty_Message** or **MC_Create_Empty_File**. In that case, the attribute is automatically added to the object before setting the value.
	MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.
	MC_INCOMPATIBLE_VR	The attribute's value representation cannot be derived from *Value*. See the table below.
	MC_VALUE_OUT_OF_RANGE	A numeric *Value* was larger than could be accommodated by the attribute. (E.g. setting an attribute with an unsigned int VR to the value 123.45, or setting it to 128000, or setting it to -1).
	MC_TEMP_FILE_ERROR	If the attribute's value is large, it may be stored in a temporary file (if so configured). If a file I/O error occurs, this status is returned.
	MC_INVALID_DATA_TYPE	*DataType* is not valid.
	MC_TOO_MANY_VALUES	This attribute contains the most values that a standard attribute can contain (65535).
	MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Set_Value... Functions
MC_Set_Value_From_Function
MC_Set_Value_To_NULL
MC_Set_Value_To_Empty
MC_Set_pValue... Functions
MC_Set_pValue_From_Function
MC_Set_pValue_To_NULL
MC_Set_pValue_To_Empty
MC_Set_Next_pValue... Functions
MC_Stream_To_Message

MC_Set_Next_Value_To_NULL

Sets the next value of an attribute in a message object to NULL.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Set_Next_Value_To_NULL (
      int MsgFileItemID,
      unsigned long Tag
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
Tag	DICOM tag which identifies the attribute.

Remarks

This function adds a NULL to the next value of a multi-valued text type attribute with the given *Tag*. Any existing values remain in the attribute. Note that the function can only be used on attributes which 1) allow the backslash (\) character as a field delimiter and 2) are of a text type. It can not be used on numerical attributes.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INCOMPATIBLE_VR	The attribute identified by *Tag* can not be set to NULL.
MC_INVALID_TAG	The message does not contain an attribute identified by *Tag* . The attribute can be added with **MC_Add_Standard_Attribute**. NOTE: This status will not be returned if setting the value of a message object which was opened using **MC_Open_Empty_Message** or **MC_Create_Empty_File**. In that case, the attribute is automatically added to the object before setting the value to NULL.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.
MC_LIBRARY_NOT_INITIALIZED	The library has not been properly initialized.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Set_Next_pValue_To_NULL

MC_Set_pValue... Functions

Sets the value of a private attribute in a message object.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Set_pValue (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      MC_DT DataType,
      void* Value
)
MC_STATUS MC_Set_pValue_From_Double (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      double Value
)
MC_STATUS MC_Set_pValue_From_Float (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      float Value
)
MC_STATUS MC_Set_pValue_From_Int (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      int Value
)
MC_STATUS MC_Set_pValue_From_ShortInt (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      short int Value
)
MC_STATUS MC_Set_pValue_From_LongInt (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      long int Value
)
MC_STATUS MC_Set_pValue_From_String (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      char* Value
)
MC_STATUS MC_Set_pValue_From_UInt (
```

```
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      unsigned int Value
)
MC_STATUS MC_Set_pValue_From_UShortInt (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      unsigned short Value
)
MC_STATUS MC_Set_pValue_From_ULongInt (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      unsigned long Value
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
PrivateCode	The code string which identifies which block in the private *Group* "owns" the attribute.
Group	The number identifying the private group containing the private attribute. It must be an odd number.
ElementByte	The number identifying the private attribute within the private *Group* for this *PrivateCode*.
DataType	One of the enumerated codes identifying the data type of the value in *Value*. The **MC_DT** enumerated type is defined in "mc3msg.h". They are:

String_Type	Null-terminated character string
Int_Type	Binary integer number
UInt_Type	Binary unsigned integer number
ShortInt_Type	Binary short integer number
UShortInt_Type	Binary unsigned short integer number
LongInt_Type	Binary long integer number
ULongInt_Type	Binary unsigned long integer number
Float_Type	Binary Floating point number

Value	The attribute should be set to this value.

Remarks

These functions store a new value for a private attribute identified by *ElementByte* for the *PrivateCode* within the given *Group*. If one or more values already exist for the attribute they are first removed. Use **MC_Set_Next_pValue** to append attribute values. If **MC_Set_pValue** is used, the data type of the *Value* is specified by the *DataType* parameter. The other function names imply the data type.. For example **MC_Set_pValue_From_Int** is the same as calling **MC_Set_pValue** with *DataType* specified as **Int_Type**. Each function will assign the value at *Value*, which must be prototyped as the appropriate type.

NOTE: string values are NULL-terminated character arrays.

Any reasonable conversion will be made from *Value's* data type to the attribute's value representation. However, some conversions are illogical (see the table below). An appropriate error code will be returned if the conversion attempt failed.

NOTE: for attributes with a value representation of **SQ** (sequence of items) *Value* must represent an integer which is the *ItemID* of an item object opened previously using **MC_Open_Item**.

Function	**Value Representations** to use for setting values of attributes
MC_Set_pValue_From_Int	DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Set_pValue_From_UInt	DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Set_pValue_From_ShortInt	DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Set_pValue_From_UShortInt	DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Set_pValue_From_LongInt	AT, DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Set_pValue_From_ULongInt	AT, DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Set_pValue_From_Float	DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Set_pValue_From_Double	DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Set_pValue_From_String	AS, AT, CS, DA, DS, DT, FD, FL, IS, LO, LT, PN, SH, SL, SS, ST, TM, UI, UL, US, SQ
MC_Set_pValue_From_Function	OB, OW, OF, SL, SS, UL, US, AT, FL, FD

The same rules apply to the **MC_Set_pValue** function, based on the value used in the *DataType* parameter. The **MC_Set_pValue_From_Function** call is described in its own section.

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

	Value	**Meaning**
INFORMATIONAL MESSAGES:	**MC_NORMAL_COMPLETION**	The function completed normally.
WARNINGS: (VALUE WAS STILL ENCODED)	**MC_INVALID_CHARS_IN_VALUE**	A string-type *Value* had invalid characters for the value representation of *Tag*.
	MC_INVALID_VALUE_FOR_VR	*Value* does not conform to the requirements of its value representation.
ERRORS:	**MC_NULL_POINTER_PARM**	One or more of the pointer-type parameters was NULL.
	MC_INVALID_TAG	The message does not contain an attribute identified by *ElementByte* in *Group* for *PrivateCode*. NOTE: This status will not be returned if setting the value of a message object which was opened using **MC_Open_Empty_Message**. In that case, the attribute is automatically added to the object before setting the value.
	MC_INVALID_PRIVATE_CODE	The message does not contain an attributes in *Group* for *PrivateCode.*
	MC_INVALID_GROUP	*Group* was not an odd number.

MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.
MC_INCOMPATIBLE_VR	The attribute's value representation cannot be derived from *Value*. See the table above.
MC_VALUE_OUT_OF_RANGE	A numeric *Value* was larger than could be accommodated by the attribute. (E.g. setting an attribute with an unsigned int VR to the value 123.45, or setting it to 128000, or setting it to -1).
MC_TEMP_FILE_ERROR	If the attribute's value is large, it may be stored in a temporary file (if so configured). If a file I/O error occurs, this status is returned.
MC_INVALID_DATA_TYPE	*DataType* is not valid.
MC_TOO_MANY_VALUES	This attribute contains the most values that a standard attribute can contain (65535).
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Set_Value... Functions
MC_Set_Value_From_Function
MC_Set_Value_To_NULL
MC_Set_Value_To_Empty
MC_Set_pValue_From_Function
MC_Set_pValue_To_NULL
MC_Set_pValue_To_Empty
MC_Set_Next_Value... Functions
MC_Set_Next_pValue... Functions
MC_Stream_To_Message

MC_Set_pValue_From_Function

Sets the value of a message object private attribute which has the value representation of OB, OW, or OF.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Set_pValue_From_Function (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      void* UserInfo,
      MC_STATUS (*YourSetFunction)()
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
PrivateCode	The code string which identifies which block in the private *Group* "owns" the attribute.
Group	The number identifying the private group containing the private attribute. It must be an odd number.
ElementByte	The number identifying the private attribute within the private *Group* for this *PrivateCode*.
UserInfo	Address of data which will be passed on to *YourSetFunction* each time it is called. This may be NULL.
YourSetFunction	Name of a function which will be called repeatedly to request blocks of data for the attribute's value.

The function must be prototyped as follows:

```
MC_STATUS YourSetFunction (
      int CBMsgFileItemID,
      unsigned long Cbtag,
      int CBisFirst,
      void* CBuserInfo,
      int* CBdataSizePtr,
      void** CBdataBufferPtr,
      int* CBisLastPtr
)
```

CBMsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
Cbtag	DICOM tag which identifies the attribute.
CBisFirst	This is TRUE (non-zero) the first time MergeCOM-3 calls *YourSetFunction* to request data blocks.
CBuserInfo	Address of data which is being passed from the **MC_Set_pValue_From_Function** function. This may be NULL.
CBdataSizePtr	Set **CBdataSizePtr* to the number of bytes you are providing.

CBdataBufferPtr	Set * *CBdataBufferPtr* to the address of the data you are providing.
CBisLastPtr	Set **CBisLastPtr* to TRUE (not zero) when you are returning with the last block of OB, OW, or OF data.

Remarks

The **MC_Set_pValue_From_Function** function is used to set the value of a private attribute which has a value representation of OB, OW, or OF. Such attributes tend to have values of great length. To accommodate this, one uses **MC_Set_pValue_From_Function** to specify the name of a function (***YourSetFunction***) which MergeCOM-3, in turn, calls. This "callback" function is called repeatedly to request blocks of the attribute's data value.

An optional *UserInfo* parameter may be used to pass information between the **MC_Set_pValue_From_Function** caller and ***YourSetFunction*** which receives the data in its *CBuserInfo* parameter. If no such exchange of data is necessary, you may use NULL for *UserInfo*.

YourSetFunction

YourSetFunction must set **CBdataBufferPtr* to the address of a block of data which is a portion of the attribute's value. It also must set **CBdataSizePtr* to the number of bytes in the block.

MergeCOM-3 sets *CBisFirst* to TRUE (non-zero) if it is requesting the first block of the attribute's value. ***YourSetFunction*** must set **CBisLastPtr* to TRUE (non-zero) if it is supplying the last block of the attribute's value.

YourSetFunction must return **MC_NORMAL_COMPLETION** if all went well. If not, it must return **MC_CANNOT_COMPLY**. Both of these are defined in **"mcstatus.h"**.

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	*PrivateCode* or *YourSetFunction* was NULL.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.
MC_INVALID_TAG	The message does not contain an attribute identified by *ElementByte* in *Group* for *PrivateCode.*
MC_INVALID_GROUP	*Group* was not an odd number.
MC_INVALID_PRIVATE_CODE	The message does not contain an attributes in *Group* for *PrivateCode.*
MC_INCOMPATIBLE_VR	The attribute's value representation was not OB, OW, or OF.
MC_CALLBACK_REGISTERED	This function is not allowed when the **MC_Register_Callback_Function** function was issued for the same attribute.

MC_CALLBACK_PARM_ERROR	The value set by ***YourSetFunction*** in its **CBdataBufferPtr* parameter was NULL and **CBisLastPtr* was not set to true(non-zero).
MC_CALLBACK_DATA_SIZE_NEGATIVE	The value set by ***YourSetFunction*** in its **CBdataSizePtr* parameter was a negative number
MC_CALLBACK_DATA_SIZE_UNEVEN	The value set by ***YourSetFunction*** in its **CBdataSizePtr* parameter was an odd number
MC_CALLBACK_CANNOT_COMPLY	***YourSetFunction*** returned a status other than **MC_NORMAL_COMPLETION.**
MC_VALUE_TOO_LARGE	The value set by ***YourSetFunction*** int **CBdataSizePtr* was too large. It may not be larger than the largest unsigned int on your machine.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Set_pValue... Functions
MC_Set_pValue_To_Empty
MC_Set_Value_To_NULL
MC_Set_Next_Value... Functions
MC_Set_pValue_To_NULL
MC_Set_Value... Functions
MC_Set_Value_To_Empty
MC_Set_Next_pValue... Functions

MC_Set_pValue_Representation

Sets a private attribute's value representation code from Unknown_VR to a valid code.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Set_pValue_Representation (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte,
      MC_VR ValueRep
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
PrivateCode	The code string which identifies which block in the private *Group* "owns" the attribute.
Group	The number identifying the private group containing the private attribute. It must be an odd number.
ElementByte	The number identifying the private attribute within the private *Group* for this *PrivateCode*.
ValueRep	A valid value representation code as defined by the **MC_VR** type in "mc3msg.h": **AE, AS, CS, DA, DS, DT, IS, LO, LT, PN, SH, ST, TM, UI, SS, US, AT, SL, UL, FL, FD, OB, OW, OF, SQ**

Remarks

MC_Set_pValue_Representation is used to change a private attribute's value representation from "UNKNOWN_VR" to a valid code. The attribute is identified by *ElementByte* for the *PrivateCode* within the given *Group*. A message attribute may have an unknown value representation if it is obtained from a message stream (**MC_Stream_To_Message**) and the attribute is unknown. An error is returned if the attribute's VR is already valid.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.
MC_INVALID_PRIVATE_CODE	The message does not contain an attributes in *Group* for *PrivateCode.*
MC_INVALID_GROUP	*Group* was not an odd number.
MC_INVALID_TAG	The message does not contain an attribute identified by *ElementByte* in *Group* for *PrivateCode*.
MC_INVALID_VR_CODE	*ValueRep* is invalid.

MC_ATTRIBUTE_HAS_VALUES	When *ValueRep* is SQ, the attribute may not already have a value.
MC_VR_ALREADY_VALID	The attribute already has a valid Value Representation.

See Also

MC_Set_Value_Representation

MC_Set_pValue_To_Empty

Removes all values from a private attribute in a message object.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Set_pValue_To_Empty (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
PrivateCode	The code string which identifies which block in the private *Group* "owns" the attribute.
Group	The number identifying the private group containing the private attribute. It must be an odd number.
ElementByte	The number identifying the private attribute within the private *Group* for this *PrivateCode*.

Remarks

MC_Set_pValue_To_Empty removes any existing values for the attribute identified by *PrivateCode*, *Group* and *ElementByte* in the message identified by *MsgFileItemID*. If a callback function was registered for the attribute, it is "de-registered."

NOTE: this is not the same as setting a NULL value. Use **MC_Set_pValue_To_NULL** to give the attribute a value length of zero (i.e. a NULL value).

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_PRIVATE_CODE	The message does not contain an attributes in *Group* for *PrivateCode*.
MC_INVALID_GROUP	*Group* was not an odd number.
MC_INVALID_TAG	The message does not contain an attribute identified by *ElementByte* in *Group* for *PrivateCode*.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.

See Also

MC_Set_Value_To_Empty
MC_Set_Value_To_NULL
MC_Set_pValue_To_NULL

MC_Set_pValue_To_NULL

Sets the value of a private attribute in a message object to NULL.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Set_pValue_To_NULL (
      int MsgFileItemID,
      char* PrivateCode,
      unsigned short Group,
      unsigned char ElementByte
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
PrivateCode	The code string which identifies which block in the private *Group* "owns" the attribute.
Group	The number identifying the private group containing the private attribute. It must be an odd number.
ElementByte	The number identifying the private attribute within the private *Group* for this *PrivateCode*.

Remarks

MC_Set_pValue_To_NULL sets the value of the attribute identified by *PrivateCode*, *Group* and *ElementByte* in the message identified by *MsgFileItemID* to NULL. That is, it sets the attribute's value length to zero. NULL is a valid value in DICOM. Use **MC_Set_pValue_To_Empty** if the intent is to remove the attribute's value altogether. If a callback function was registered for the attribute, it is "de-registered."

NOTE: If the attribute has a value representation of SQ (sequence of items), setting the value to null does NOT free the item object identified by the value.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_PRIVATE_CODE	The message does not contain an attributes in *Group* for *PrivateCode*.
MC_INVALID_GROUP	*Group* was not an odd number.
MC_INVALID_TAG	The message does not contain an attribute identified by *ElementByte* in *Group* for *PrivateCode*.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.

See Also

MC_Set_pValue_To_Empty
MC_Set_Value_To_Empty
MC_Set_Value_To_NULL

MC_Set_Service_Command

Associates a MergeCOM-3 message or file object with a given DICOM service/command pair.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Set_Service_Command (
      int MessageID,
      char* ServiceName,
      MC_COMMAND Command
)
```

MessageID	The identifier assigned to the message object by the **MC_Open_Empty_Message**, **MC_Open_Message**, **MC_Create_Empty_File**, or **MC_Create_File** function.
ServiceName	String name of a DICOM service to be associated with this message object.
Command	The command which is to be associated with this message. The **MC_COMMAND** enumerated values are defined in "mc3msg.h".

Remarks

The **MC_Set_Service_Command** function associates a message object with a given DICOM service/command pair. This is necessary if the message object was opened using **MC_Open_Empty_Message** and the message object is to be sent to a network partner, or is to be validated using the **MC_Validate_Message** function.

The *ServiceName* must be a valid service name defined in the MergeCOM service profile file, and *Command* must be a valid DICOM command.

If the service "DICOMDIR" and command "C_STORE_RQ" is specified and the object identified by *MessageID* is a file object, the object will be converted to a DICOMDIR object. The object can then be manipulated with the **MC_Dir_...** functions. **MC_Set_Service_Command** should only be used on empty file objects, because the **MC_Dir_...** functions will not work correctly if any directory records are already contained in the object.

The following commands are supported by the Advanced MergeCOM-3 Tool Kit:

Command	Description
C_STORE_RQ	DICOM Composite Store Service Request
C_STORE_RSP	DICOM Composite Store Service Response
C_ECHO_RQ	DICOM Verification Service Request
C_ECHO_RSP	DICOM Verification Service Response
C_FIND_RQ	DICOM Composite Find Service Request
C_FIND_RSP	DICOM Composite Find Service Response
C_CANCEL_FIND_RQ	Cancel DICOM Composite Find Service Request
C_GET_RQ	DICOM Composite Get Service Request
C_GET_RSP	DICOM Composite Get Service Response

C_CANCEL_GET_RQ	Cancel DICOM Composite Get Service Request
C_MOVE_RQ	DICOM Composite Move Service Request
C_MOVE_RSP	DICOM Composite Move Service Response
C_CANCEL_MOVE_RQ	Cancel DICOM Composite Move Service Request
N_EVENT_REPORT_RQ	DICOM Normalized Report Service Request
N_EVENT_REPORT_RSP	DICOM Normalized Report Service Response
N_GET_RQ	DICOM Normalized Get Service Request
N_GET_RSP	DICOM Normalized Get Service Request
N_SET_RQ	DICOM Normalized Set Service Request
N_SET_RSP	DICOM Normalized Set Service Response
N_ACTION_RQ	DICOM Normalized Action Service Request
N_ACTION_RSP	DICOM Normalized Action Service Response
N_CREATE_RQ	DICOM Normalized Create Service Request
N_CREATE_RSP	DICOM Normalized Create Service Response
N_DELETE_RQ	DICOM Normalized Delete Service Request
N_DELETE_RSP	DICOM Normalized Delete Service Response

Reference the “DICOM V3.0 Standard, Final Text - October 29, 1993” for more information about these commands.

Return Value

One of these enumerated **MC_STATUS** codes defined in **“mcstatus.h”**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	*ServiceName* was NULL.
MC_INVALID_MESSAGE_ID	*MessageID* is not a valid message or file object ID.
MC_INVALID_COMMAND	*Command* is not a supported command.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Create_Empty_File
MC_Create_File
MC_Open_Empty_Message
MC_Open_Message
MC_Get_MergeCOM_Service
MC_Get_UID_From_MergeCOM_Service

MC_Set_String_Config_Value

Used to set the value of a character string tool kit configuration parameter at runtime.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Set_String_Config_Value (
      StringParm Aparm,
      char* Avalue
)
```

Aparm An enumerated constant identifying the character string configuration parameter to be (re)set. Aparm can have any of the following values:

LOG_FILE,
LICENSE,
IMPLEMENTATION_CLASS_UID,
IMPLEMENTATION_VERSION,
LOCAL_APPL_CONTEXT_NAME,
IMPLICIT_LITTLE_ENDIAN_SYNTAX,
IMPLICIT_BIG_ENDIAN_SYNTAX,
EXPLICIT_LITTLE_ENDIAN_SYNTAX,
EXPLICIT_BIG_ENDIAN_SYNTAX,
DEFLATED_EXPLICIT_LITTLE_ENDIAN_SYNTAX,
RLE_SYNTAX,
JPEG_BASELINE_SYNTAX,
JPEG_EXTENDED_2_4_SYNTAX,
JPEG_EXTENDED_3_5_SYNTAX,
JPEG_SPEC_NON_HIER_6_8_SYNTAX,
JPEG_SPEC_NON_HIER_7_9_SYNTAX,
JPEG_FULL_PROG_NON_HIER_10_12_SYNTAX,
JPEG_FULL_PROG_NON_HIER_11_13_SYNTAX,
JPEG_LOSSLESS_NON_HIER_14_SYNTAX,
JPEG_LOSSLESS_NON_HIER_15_SYNTAX,
JPEG_EXTENDED_HIER_16_18_SYNTAX,
JPEG_EXTENDED_HIER_17_19_SYNTAX,
JPEG_SPEC_HIER_20_22_SYNTAX,
JPEG_SPEC_HIER_21_23_SYNTAX,
JPEG_FULL_PROG_HIER_24_26_SYNTAX,
JPEG_FULL_PROG_HIER_25_27_SYNTAX,
JPEG_LOSSLESS_HIER_28_SYNTAX,
JPEG_LOSSLESS_HIER_29_SYNTAX,
JPEG_LOSSLESS_HIER_14_SYNTAX,
JPEG_2000_LOSSLESS_ONLY_SYNTAX,
JPEG_2000_SYNTAX,
INITIATOR_NAME,
RECEIVER_NAME,
DICTIONARY_FILE,
MSG_INFO_FILE,
TEMP_FILE_DIRECTORY,
UNKNOWN_VR_CODE,
PRIVATE_SYNTAX_1_SYNTAX,
PRIVATE_SYNTAX_2_SYNTAX,
CAPTURE_FILE,
PEGASUS_OP_LIP3PLUS_NAME,

PEGASUS_OP_LIE3PLUS_NAME,
PEGASUS_OP_D2SEPLUS_NAME
PEGASUS_OP_SE2DPLUS_NAME,
PEGASUS_OP_J2KP_NAME,
PEGASUS_OP_J2KE_NAME,
PEGASUS_OP_LIP3PLUS_REGISTRATION,
PEGASUS_OP_LIE3PLUS_REGISTRATION,
PEGASUS_OP_D2SEPLUS_REGISTRATION,
PEGASUS_OP_SE2DPLUS_REGISTRATION,
PEGASUS_OP_J2KP_REGISTRATION,
PEGASUS_OP_J2KE_REGISTRATION,
LARGE_DATA_STORE,
DICTIONARY_ACCESS,
NULL_TYPE3_VALIDATION

These names are the same as those given to the parameters in the tool kit configuration files.

Avalue The character string value to which Aparm is to be set.

Remarks

The MergeCOM-3 Advanced Tool Kit Library accesses several configuration files at startup. This call allows your application to (re)set character string configurable parameters specified in these files at runtime. This call should be made immediately after calling **MC_Library_Initialization** to avoid using these parameters before they are set. Please see the detailed description of tool kit configuration elsewhere in this manual.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_PARAMETER_NAME	Parameter specified is not one of the above listed parameters.
MC_NULL_POINTER_PARM	Avalue was a null pointer.
MC_SYSTEM_ERROR	Out of memory condition occurred.
MC_LIBRARY_NOT_INITIALIZED	The library has not been properly initialized.

See Also

MC_Get_Int_Config_Value
MC_Get_Log_Destination
MC_Get_Long_Config_Value
MC_Get_String_Config_Value
MC_Set_Int_Config_Value
MC_Set_Log_Destination
MC_Set_Long_Config_Value
MC_Set_String_Config_Value

MC_Set_Value... Functions

Sets the value of an attribute in a message object.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Set_Value (
      int MsgFileItemID,
      unsigned long Tag,
      MC_DT DataType,
      void* Value
)
MC_STATUS MC_Set_Value_From_Double (
      int MsgFileItemID,
      unsigned long Tag,
      double Value
)
MC_STATUS MC_Set_Value_From_Float (
      int MsgFileItemID,
      unsigned long Tag,
      float Value
)
MC_STATUS MC_Set_Value_From_Int (
      int MsgFileItemID,
      unsigned long Tag,
      int Value
)
MC_STATUS MC_Set_Value_From_ShortInt
      (int MsgFileItemID,
      unsigned long Tag,
      short int Value
)
MC_STATUS MC_Set_Value_From_LongInt (
      int MsgFileItemID,
      unsigned long Tag,
      long int Value
)
MC_STATUS MC_Set_Value_From_String (
      int MsgFileItemID,
      unsigned long Tag,
      char* Value
)
MC_STATUS MC_Set_Value_From_UInt (
      int MsgFileItemID,
      unsigned long Tag,
      unsigned int Value
)
MC_STATUS MC_Set_Value_From_UShortInt (
      int MsgFileItemID,
      unsigned long Tag,
      unsigned short Value
)
MC_STATUS MC_Set_Value_From_ULongInt (
      int MsgFileItemID,
      unsigned long Tag,
      unsigned long Value
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
Tag	DICOM tag which identifies the attribute.
DataType	One of the enumerated codes identifying the data type of the value in *Value*. The MC_DT enumerated type is defined in "mc3msg.h". They are:

String_Type	Null-terminated character string
Int_Type	Binary integer number
UInt_Type	Binary unsigned integer number
ShortInt_Type	Binary short integer number
UShortInt_Type	Binary unsigned short integer number
LongInt_Type	Binary long integer number
ULongInt_Type	Binary unsigned long integer number
Float_Type	Binary Floating point number

Value	The value to be used as the attribute's value.

Remarks

These functions store a new value for the attribute with the given *Tag*. If one or more values already exist for the attribute they are first removed. Use **MC_Set_Next_Value** to append attribute values. If **MC_Set_Value** is used, the data type of the *Value* is specified by the *DataType* parameter. The other function names imply the data type.. For example **MC_Set_Value_From_Int** is the same as calling **MC_Set_Value** with *DataType* specified as **Int_Type**. Each function will set the value at *Value*, which must be prototyped as the appropriate type. Note that string values are NULL-terminated character arrays.

Any reasonable conversion will be made from *Value's* data type to the attribute's value representation. However, some conversions are illogical (see the table below). An appropriate error code will be returned if the conversion attempt failed.

NOTE: for attributes with a value representation of **SQ** (sequence of items) *Value* must represent an integer which is the *ItemID* of an item object opened previously using **MC_Open_Item**.

Function	**Value Representations** to use for setting values of attributes
MC_Set_Value_From_Int	DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Set_Value_From_UInt	DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Set_Value_From_ShortInt	DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Set_Value_From_UShortInt	DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Set_Value_From_LongInt	AT, DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Set_Value_From_ULongInt	AT, DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Set_Value_From_Float	DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Set_Value_From_Double	DS, FD, FL, IS, SL, SS, UL, US, SQ
MC_Set_Value_From_String	AS, AT, CS, DA, DS, DT, FD, FL, IS, LO, LT, PN, SH, SL, SS, ST, TM, UI, UL, US, SQ
MC_Set_Value_From_Function	OB, OW, OL, OF, SL, SS, UL, US, AT, FL, FD

The same rules apply to the **MC_Set_Value** function, based on the value used in the *DataType* parameter. The MC_**Set_Value_From_Function** call is described in its own section.

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

	Value	Meaning
INFORMATIONAL MESSAGES:	**MC_NORMAL_COMPLETION**	The function completed normally.
WARNINGS: (VALUE WAS STILL ENCODED)	**MC_INVALID_CHARS_IN_VALUE**	A string-type *Value* had invalid characters for the value representation of *Tag*.
	MC_INVALID_VALUE_FOR_VR	*Value* does not conform to the requirements of its value representation.
ERRORS:	**MC_NULL_POINTER_PARM**	One or more of the pointer-type parameters was NULL.
	MC_INVALID_TAG	The message does not contain an attribute with an ID of *Tag*. The attribute can be added with **MC_Add_Standard_Attribute**. NOTE: This status will not be returned if setting the value of a message object which was opened using **MC_Open_Empty_Message** or **MC_Create_Empty_File**. In that case, the attribute is automatically added to the object before setting the value.
	MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.
	MC_INCOMPATIBLE_VR	The attribute's value representation cannot be derived from *Value*. See the table below.
	MC_VALUE_OUT_OF_RANGE	A numeric *Value* was larger than could be accommodated by the attribute. (E.g. setting an attribute with an unsigned int VR to the value 123.45, or setting it to 128000, or setting it to -1).
	MC_TEMP_FILE_ERROR	If the attribute's value is large, it may be stored in a temporary file (if so configured). If a file I/O error occurs, this status is returned.
	MC_INVALID_DATA_TYPE	*DataType* is not valid.
	MC_TOO_MANY_VALUES	This attribute contains the most values that a standard attribute can contain (65535).
	MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Set_Value_From_Function
MC_Set_Value_To_Empty
MC_Set_pValue_From_Function
MC_Set_Value_To_NULL
MC_Set_pValue... Functions
MC_Set_pValue_To_NULL

MC_Set_pValue_To_Empty **MC_Set_Next_Value... Functions**
MC_Set_Next_pValue... Functions

MC_Set_Value_From_Function

Sets the value of a message object attribute which has the value representation of OB, OW, or OF.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Set_Value_From_Function (
      int MsgFileItemID,
      unsigned long Tag,
      void* UserInfo,
      MC_STATUS (*YourSetFunction)()
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
Tag	DICOM tag which identifies the attribute.
UserInfo	Address of data which will be passed on to *YourSetFunction* each time it is called. This may be NULL.
YourSetFunction	Name of a function which will be called repeatedly to request blocks of data for the attribute's value.

The function must be prototyped as follows:

```
MC_STATUS YourSetFunction (
      int CBMsgFileItemID,
      unsigned long CBtag,
      int CBisFirst,
      void* CBuserInfo,
      int* CBdataSizePtr,
      void** CBdataBufferPtr,
      int* CBisLastPtr
)
```

CBMsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
Cbtag	DICOM tag which identifies the attribute.
CBisFirst	This is TRUE (non-zero) the first time MergeCOM-3 calls *YourSetFunction* to request data blocks.
CBuserInfo	Address of data which is being passed from the **MC_Set_Value_From_Function** function. This may be NULL.
CBdataSizePtr	Set **CBdataSizePtr* to the number of bytes you are providing.
CBdataBufferPtr	Set * *CBdataBufferPtr* to the address of the data you are providing.
CBisLastPtr	Set **CBisLastPtr* to TRUE (not zero) when you are returning with the last block of OB, OW, or OF data.

Remarks

The **MC_Set_Value_From_Function** function is used to set the value of an attribute which has a value representation of OB, OW, or OF. Such attributes tend to have values of great length. To accommodate this, one uses the **MC_Set_Value_From_Function**

function to specify the name of a function (***YourSetFunction***) which MergeCOM-3, in turn, calls. This "callback" function is called repeatedly to request blocks of the attribute's data value.

An optional *UserInfo* parameter may be used to pass information between the **MC_Set_Value_From_Function** caller and ***YourSetFunction*** which receives the data in its *CBuserInfo* parameter. If no such exchange of data is necessary, you may use NULL for *UserInfo*.

YourSetFunction

YourSetFunction must set **CBdataBufferPtr* to the address of a block of data which is a portion of the attribute's value. It also must set **CBdataSizePtr* to the number of bytes in the block.

MergeCOM-3 sets *CBisFirst* to TRUE (non-zero) if it is requesting the first block of the attribute's value. ***YourSetFunction*** must set **CBisLastPtr* to TRUE (non-zero) if it is supplying the last block of the attribute's value.

YourSetFunction must return **MC_NORMAL_COMPLETION** if all went well. If not, it must return **MC_CANNOT_COMPLY**. Both of these are defined in "mc3msg.h".

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	One or more of the pointer-type parameters was NULL.
MC_INVALID_TAG	The message does not contain an attribute with an ID of *Tag*. The attribute can be added with **MC_Add_Standard_Attribute**. NOTE: This status will not be returned if setting the value of a message object which was opened using **MC_Open_Empty_Message** or **MC_Create_Empty_File**. In that case, the attribute is automatically added to the object before setting the value.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.
MC_INCOMPATIBLE_VR	The attribute's value representation was not OB, OW, or OF.
MC_CALLBACK_REGISTERED	This function is not allowed when the **MC_Register_Callback_Function** function was issued for the same attribute.
MC_CALLBACK_DATA_SIZE_UNEVEN	The value set by ***YourSetFunction*** in its **CBdataSizePtr* parameter was not an even number.
MC_CALLBACK_PARM_ERROR	The value set by ***YourSetFunction*** in its **CBdataSizePtr* parameter was an odd number

MC_CALLBACK_CANNOT_COMPLY

YourSetFunction returned with **MC_CANNOT_COMPLY.**

See Also

MC_Set_Value... Functions
MC_Set_Next_Value... Functions

MC_Set_Value_Representation

Sets an attribute's value representation code from Unknown_VR to a valid code.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Set_Value_Representation (
      int MsgFileItemID,
      unsigned long Tag,
      MC_VR ValueRep
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
Tag	A tag identifying the attribute.
ValueRep	A valid value representation code as defined by the **MC_VR** type in "mc3msg.h": **AE, AS, CS, DA, DS, DT, IS, LO, LT, PN, SH, ST, TM, UI, SS, US, AT, SL, UL, FL, FD, OB, OW, OF, SQ**

Remarks

MC_Set_Value_Representation is used to change an attribute's value representation from "UNKNOWN_VR" to a valid code. A message attribute may have an unknown value representation if it is obtained from a message stream (**MC_Stream_To_Message**) and the attribute is unknown. An error is returned if the attribute's VR is already valid.

This function can also be used to change US to OW or SS, or SS to OW or US to accomodate attributes that have more then one possible VR.

OB, OW, or OF may also be changed to the appropriate VR using this function.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.
MC_INVALID_TAG	The message does not contain an attribute with an ID of *Tag*.
MC_INVALID_VR_CODE	*ValueRep* is invalid.
MC_ATTRIBUTE_HAS_VALUES	When *ValueRep* is SQ, the attribute may not already have a value.
MC_VR_ALREADY_VALID	The attribute already has a valid Value Representation.

See Also

MC_Set_pValue_Representation

MC_Set_Value_To_Empty

Removes all values from an attribute in a message object.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Set_Value_To_Empty (
      int MsgFileItemID,
      unsigned long Tag
)
```

MsgItemItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
Tag	DICOM tag which identifies the attribute.

Remarks

MC_Set_Value_To_Empty removes any existing values for the attribute identified by *Tag* in the message identified by *MsgFileItemID*. If a callback function was registered for the attribute, it is "de-registered."

NOTE: this is not the same as setting a NULL value. Use **MC_Set_Value_To_NULL** to give the attribute a value length of zero (i.e. a NULL value).

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_TAG	The message does not contain an attribute with an ID of *Tag*.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.

See Also

MC_Set_pValue_To_Empty
MC_Set_Value_To_NULL
MC_Set_pValue_To_NULL

MC_Set_Value_To_NULL

Sets the value of an attribute in a message object to NULL.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Set_Value_To_NULL (
      int MsgFileItemID,
      unsigned long Tag
)
```

MsgFileItemID	The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_File**, **MC_Create_Empty_File**, or **MC_Open_Item** functions.
Tag	DICOM tag which identifies the attribute.

Remarks

MC_Set_Value_To_NULL sets the value of the attribute identified by *Tag* in the message identified by *MsgFileItemID* to NULL. That is, it sets the attribute's value length to zero. NULL is a valid value in DICOM. Use **MC_Set_Value_To_Empty** if the intent is to remove the attribute's value altogether.

If a callback function was registered for the attribute, it is "de-registered."

NOTE: if the attribute has a value representation of SQ (sequence of items), setting the value to null does NOT free the item object identified by the value.

Return Value

One of these enumerated **MC_STATUS** codes define in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_TAG	The message does not contain an attribute with an ID of *Tag*. The attribute can be added with **MC_Add_Standard_Attribute**. NOTE: This status will not be returned if setting the value of a message object which was opened using **MC_Open_Empty_Message** or **MC_Create_Empty_File**. In that case, the attribute is automatically added to the object before setting the value to NULL.
MC_INVALID_MESSAGE_ID	*MsgFileItemID* is not a valid message object ID, file object ID or item object ID.

See Also

MC_Set_Value_To_Empty **MC_Set_pValue_To_Empty**
MC_Set_pValue_To_NULL

MC_SR_Add_Child

Adds a lower level SR node to a specified SR tree management.

Synopsis

```
#include "mc3sr.h"

MC_STATUS MC_SR_Add_Child (
      int SRID,
      int ItemID
)
```

SRID	The identifier assigned to the given SR object to which the child is to be added.
ItemID	The identifier assigned to the child SR object which is to be added to the parent.

Remarks

MC_SR_Add_Child adds a new lower level SR tree management node referenced by the item object *SRID*, and places it in the SR tree management object identified by *ItemID*.

The child SR tree management record node is placed at the end of *SRID*'s SR record list and all internal links to the new entity are adjusted by MergeCOM-3.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_SR_ID	The *SRID* value is not a valid SR tree management object ID.
MC_INVALID_SR_ITEM_ID	The *ItemID* value is not a valid SR tree management record object ID.
MC_NULL_POINTER_PARM	The *NewEntityID*, *NewItemID*, or *NewItemName* parameter was NULL.
MC_MSGFILE_ERROR	An error occurred attempting to access the configuration data file for the directory record.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_SR_Add_Child
MC_SR_Get_First_Child
MC_SR_Get_Next_Child
MC_SR_Get_Root
MC_Delete_Child

MC_SR_Add_Root

Adds the root node to the SR tree management structure.

Synopsis

```
#include "mc3sr.h"

MC_STATUS MC_SR_Add_Root (
      int MsgID,
      int* SRID
)
```

MsgID	The identifier assigned to the message.
SRID	Upon successful completion, the root SR item object identifier is returned here.

Remarks

MC_SR_Add_Root adds an node as the root node of an SR tree management object. A valid identifier of a message object must be passed into this function. This returned item identifier can then be used to set and retrieve SR values.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_SR_ID	The *SRID* value is not a valid SR tree management object ID.
MC_MSGFILE_ERROR	An error occurred attempting to access the configuration data file for the directory record.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_SR_Add_Child
MC_SR_Get_First_Child
MC_SR_Get_Next_Child
MC_SR_Get_Root
MC_SR_Delete_Child

MC_SR_Delete_Child

Deletes a given SR child node.

Synopsis

```
#include "mc3sr.h"

MC_STATUS MC_SR_Delete_Child (
      int SRID
)
```

SRID The identifier of the SR item which is to be deleted is passed into this function here.

Remarks

MC_SR_Delete_Child deletes the given child SR object and all associated child objects from the SR tree management structures.

Note well that while this function does not manage the changes to any references to deleted objects from elsewhere in the SR message, **MC_SR_Delete_Child** does update all of the location values for affected SR objects within the SR tree.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_SR_ID	The *SRID* value is not a valid SR tree management object ID.
MC_MSGFILE_ERROR	An error occurred attempting to access the configuration data file for the directory record.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_SR_Add_Child
MC_SR_Get_First_Child
MC_SR_Get_Next_Child
MC_SR_Get_Root
MC_SR_Delete_Child

MC_SR_Get_First_Child

Returns a pointer to the first child node at a given parent node of the SR tree management structure.

Synopsis

```
#include "mc3sr.h"

MC_STATUS MC_SR_First_Child (
      int SRID,
      int* ItemID,
      int* IsLast
)
```

SRID	The identifier assigned to this SR tree management node.
ItemID	Upon successful completion, the item object identifier of the first SR tree management child record of the given *SRID* object is returned here.
IsLast	Upon successful completion, this parameter is set to true (non-zero) if the first record is also the last record in the SR tree management entity (i.e., it is the only element)

Remarks

MC_SR_Get_First_Child retrieves the identifier of the first SR tree management child record of the SR tree management item *ItemID*. If *ItemID* is an empty root SR tree management entity, **IsLast* is set to non-zero, and *ItemID* is set to -1.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_SR_ID	The *SRID* value is not a valid SR tree management object ID.
MC_INVALID_SR_ITEM_ID	The *ItemID* value is not a valid SR tree management record object ID.
MC_NULL_POINTER_PARM	The *NewEntityID*, *NewItemID*, or *NewItemName* parameter was NULL.
MC_MSGFILE_ERROR	An error occurred attempting to access the configuration data file for the directory record.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_SR_Add_Child
MC_SR_Get_First_Child
MC_SR_Get_Next_Child
MC_SR_Get_Root
MC_SR_Delete_Child

MC_SR_Get_Location

Returns the location within the current SR tree management structure.

Synopsis

```
#include "mc3sr.h"

MC_STATUS MC_SR_Get_Location (
      int SRID,
      int sizeOfLocation,
      char* Location
)
```

SRID	The identifier assigned to this SR tree management object.
sizeOfLocation	The size of the character array that the toolkit is to place the value for *Location* into.
Location	String name of the location is returned here upon successful completion.

Remarks

MC_SR_Get_Location is used to obtain a string value which represents the location of a given item within the SR tree management structure. The string that is returned will contain a dot-separated list of numbers. These numbers represent the numerical "position" of the item within the tree structure. For example, if the number returned by this function is "1.2.1.3", then the item referenced by *SRID* is the third item, under the first item, under the second item, under the root entity.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_SR_ID	The *SRID* value is not a valid SR tree management object ID.
MC_INVALID_SR_ITEM_ID	The *ItemID* value is not a valid SR tree management record object ID.
MC_NULL_POINTER_PARM	The *NewEntityID*, *NewItemID*, or *NewItemName* parameter was NULL.
MC_MSGFILE_ERROR	An error occurred attempting to access the configuration data file for the directory record.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_SR_Add_Child
MC_SR_First_Child
MC_SR_Next_Child
MC_SR_Root
MC_Delete_Child

MC_SR_Get_Next_Child

Retrieves a pointer to the next child SR tree management node linked to a specified SR tree management record

Synopsis

```
#include "mc3sr.h"

MC_STATUS MC_SR_Get_Next_Child (
      int SRID,
      int* NextChildID,
      int* IsLast
)
```

SRID	The identifier assigned to this SR tree management object.
NextChildID	Upon successful completion, the item object identifier of the next SR tree management node in the SR tree management entity *NextChildID* is returned here. At a given parent *SRID* a pointer to the next record is maintained.
IsLast	Upon successful completion, this parameter is set to true (non-zero) if the next record in the SR tree management entity *NextChildID* is also the last record (i.e., it is the only element)

Remarks

MC_SR_Get_Next_Child returns *NextChildID*, the entity object identifier of the next child SR tree management node linked to *SRID*. The value *IsLast* is set if the given *NextChildID* is the last in the SR tree management structure.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_SR_ID	The *SRID* value is not a valid SR tree management object ID.
MC_INVALID_SR_ITEM_ID	The *ItemID* value is not a valid SR tree management record object ID.
MC_NULL_POINTER_PARM	The *NewEntityID*, *NewItemID*, or *NewItemName* parameter was NULL.
MC_MSGFILE_ERROR	An error occurred attempting to access the configuration data file for the directory record.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_SR_Add_Child
MC_SR_Get_First_Child
MC_SR_Get_Next_Child
MC_SR_Get_Root
MC_Delete_Child

MC_SR_Get_Root

Returns a pointer to the root node of an SR tree management structure.

Synopsis

```
#include "mc3media.h"

MC_STATUS MC_SR_Root (
      int MsgID,
      int* SRID
)
```

MsgID	The identifier assigned to this SR tree management object by the **MC_Message_To_SR** function.
SRID	Upon successful completion, the root SR item object identifier is returned here.

Remarks

MC_SR_Get_Root returns an identifier to the root node of an SR tree management object. A valid identifier of a message containing an SR tree object must be passed into this function. This item identifier can then be used to set and retrieve SR values.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_SR_ID	The *SRID* value is not a valid SR tree management object ID.
MC_MSGFILE_ERROR	An error occurred attempting to access the configuration data file for the directory record.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_SR_Add_Child
MC_SR_Get_First_Child
MC_SR_Get_Next_Child
MC_SR_Get_Root
MC_SR_Delete_Child

MC_SR_To_Message

Converts a toolkit managed SR tree object into a toolkit managed message object.

Synopsis

```
#include "mc3media.h"

MC_STATUS MC_SR_To_Message (
      int SRID
)
```

SRID The identifier assigned to an existing toolkit message that has an SR tree management structure associated with it.

Remarks

MC_SR_To_Message converts an SR tree management object into a normal toolkit message object by deleting the SR tree management structures from the toolkit's memory

Since the SR API calls are used to alter a toolkit DICOM message, this function is used to remove the SR tree management structures from the toolkit's memory. This call can be used as a last step, after modifications are completed to the normal toolkit message.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_INVALID_SR_ID	The *SRID* value is not a valid SR tree management object ID.
MC_MSGFILE_ERROR	An error occurred attempting to access the configuration data file for the directory record.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_SR_Add_Child
MC_SR_Get_First_Child
MC_SR_Get_Next_Child
MC_SR_Get_Root
MC_SR_Delete_Child

MC_Stream_To_Message
MC_Stream_To_Message_With_Offset

Request that the values of a message object be retrieved from a DICOM stream.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Stream_To_Message (
      int MessageID,
      unsigned long StartTag,
      unsigned long StopTag,
      TRANSFER_SYNTAX SyntaxType,
      unsigned long* ErrorTagPtr,
      void* UserInfo,
      MC_STATUS (*YourStreamToFunction)()
)

MC_STATUS MC_Stream_To_Message_With_Offset (
      int MessageID,
      unsigned long StartTag,
      unsigned long StopTag,
      TRANSFER_SYNTAX SyntaxType,
      unsigned long* ErrorTagPtr,
      unsigned long* Offset,
      void* UserInfo,
      MC_STATUS (*YourStreamToFunction)()
)
```

MessageID — The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, or the **MC_Open_Item** functions.

StartTag — The DICOM tag identifying the first attribute which should be added to the message.

StopTag — The DICOM tag identifying the last attribute which should be added to the message.

SyntaxType — Specify which DICOM transfer syntax was used to encode the stream data you will be providing. Use one of the enumerated TRANSFER_SYNTAX types defined in "mc3msg.h":

IMPLICIT_LITTLE_ENDIAN
IMPLICIT_BIG_ENDIAN
EXPLICIT_LITTLE_ENDIAN
EXPLICIT_BIG_ENDIAN
DEFLATED_EXPLICIT_LITTLE_ENDIAN
RLE
JPEG_BASELINE
JPEG_EXTENDED_2_4
JPEG_EXTENDED_3_5
JPEG_SPEC_NON_HIER_6_8
JPEG_SPEC_NON_HIER_7_9
JPEG_FULL_PROG_NON_HIER_10_12
JPEG_FULL_PROG_NON_HIER_11_13
JPEG_LOSSLESS_NON_HIER_14
JPEG_LOSSLESS_NON_HIER_15
JPEG_EXTENDED_HIER_16_18
JPEG_EXTENDED_HIER_17_19
JPEG_SPEC_HIER_20_22

	JPEG_SPEC_HIER_21_23 JPEG_FULL_PROG_HIER_24_26 JPEG_FULL_PROG_HIER_25_27 JPEG_LOSSLESS_HIER_28 JPEG_LOSSLESS_HIER_29 JPEG_LOSSLESS_HIER_14 JPEG_2000_LOSSLESS_ONLY JPEG_2000 PRIVATE_SYNTAX_1 or PRIVATE_SYNTAX_2
ErrorTagPtr	If an error occurs, **ErrorTagPtr* will be set to the tag which caused the error.
Offset	The offset from the beginning of the stream to the attribute after *StopTag*.
UserInfo	Address of data which will be passed on to *YourStreamToFunction* each time it is called. This may be NULL.
YourStreamToFunction	Name of a function which will be called repeatedly to request blocks of streamed DICOM message data.

The function must be prototyped as follows:

```
MC_STATUS YourStreamToFunction (
      int CBmessageID,
      void* CBuserInfo,
      int CBFirstCall,
      int* CBdataSizePtr,
      void** CBdataBuffer,
      int* CBisLastPtr
)
```

CBmessageID	The identifier assigned to the message object by the **MC_Open_Message** function.
CBuserInfo	Address of data which is being passed from the **MC_Stream_To_Message** function. This may be NULL.
CBFirstCall	This is TRUE (non-zero) the first time MergeCOM-3 calls *YourStreamToFunction* to request data blocks.
CBdataSizePtr	Set **CBdataSizePtr* to the number of bytes you are providing. This must be an even number.
CBdataBuffer	Set ** CBdataBuffer* to the address of the data you are providing.
CBisLastPtr	Set **CBisLastPtr* to TRUE (not zero) when you are returning with the last block of stream data.

Remarks

MC_Stream_To_Message requests that the contents of a "streamed message" (i.e. a message in the form defined by the DICOM standard) be converted and placed in the message object or item object identified by *MessageID*. The streamed message is passed to MergeCOM-3 by ***YourStreamToFunction***. MergeCOM-3 repeatedly calls ***YourStreamToFunction*** until all of the streamed message has been processed.

MC_Stream_To_Message can pass data to ***YourStreamToFunction*** by specifying the data's address in *UserInfo*. MergeCOM-3 passes the address to ***YourStreamToFunction*** in *CBuserInfo* each time it is called. *UserInfo* may be NULL.

If an error occurs while processing the stream, MergeCOM-3 will put the tag of the attribute in error in **ErrorTagPtr*.

MC_Stream_To_Message_With_Offset is identical to **MC_Stream_To_Message** except that it returns the byte offset from the beginning of the stream to the next attribute after *StopTag*. If there is not a tag after *StopTag*, the length of the file will be returned.

StartTag and *StopTag* specify which attributes in the stream are to be placed in the message object. Any attributes in the stream with tags less than *StartTag* or greater than *StopTag* will be ignored. Neither *StartTag* nor *StopTag* need be in the stream.

SyntaxType must be set to **one of the values listed above.** The transfer syntax specifies the byte order used in the streamed message, whether or not each attribute's value representation is explicitly encoded in the stream, and how the pixel data is encoded in the message.

If the stream contains any attributes with a value representation of SQ (i.e. the stream contains one or more sequence of items), an item object is automatically opened for each item in the stream. The *ItemID* associated with each opened item object is used as the value for each item in the sequence attribute. Later, the **MC_Get_Value...** functions may be used to retrieve the *ItemID*'s from the SQ attribute. Then, again using the **MC_Get_Value...** functions, the attributes of the *ItemID* object may be retrieved.

NOTE: A runtime configuration parameter determines what happens if the input stream contains an attribute which is not in the message. The default is to ignore such attributes (with a warning message logged). If requested, however, such attributes and their values are added to the message. If the Value Representation of the attribute being added cannot be determined, the attribute is given a pseudo Value Representation of "**Unknown_VR**". Only the **MC_Get_Value_To_Buffer** function retrieves the value of such attributes. To change the value of such attributes, **MC_Set_Value_Representation** or **MC_Set_pValue_Representation** must first be called to assign a valid Value Representation to the attribute. If **MC_Message_To_Stream** is used, attributes with unknown VR's are simply copied (memcpy) to the stream with no consideration given to byte ordering.

YourStreamToFunction

It is the responsibility of ***YourStreamToFunction*** to pass blocks of data back to MergeCOM-3 each time it is called. As a convenience, MergeCOM-3 sets *CBFirstCall* to TRUE (non-zero) the first time it calls ***YourStreamToFunction*** for this attribute.

**CBdataBufferPtr* must be set to the address of the buffer containing the stream data block, and **CBdataSizePtr* must be set to the number of bytes placed at **CBdataBufferPtr*.

YourStreamToFunction must set **CBisLast* to TRUE(non-zero) when it is providing the last block of the streamed message.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	*YourStreamToFunction* or *ErrorTagPtr* was NULL.

MC_INVALID_MESSAGE_ID	The *MessageID* value is not a valid message object ID.
MC_INVALID_TRANSFER_SYNTAX	An invalid code was used for the *SyntaxType* parameter.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.
MC_CALLBACK_CANNOT_COMPLY	***YourStreamToFunction*** returned with a status other than MC_NORMAL_COMPLETION.
MC_CALLBACK_DATA_SIZE_UNEVEN	The *CBdataSizePtr* parameter returned by the *YourStreamToFunction* was an uneven number.
MC_CALLBACK_PARM_ERROR	A callback function registered by your application returned an empty (NULL) data buffer when the buffers length was specified as being non-zero. See **MC_Register_Callback_Function**.
MC_OUT_OF_ORDER_TAG	A tag was found in the file that was not in ascending order. A message describing the error has been written to the MergeCOM-3 log file.
MC_INVALID_TAG	The message contains an invalid tag. The tag is placed at **ErrorTagPtr*.
MC_VALUE_TOO_LARGE	An attribute in the stream message (other than those with value representations of OB, OW, or OF) was larger than that acceptable for its value representation.
MC_UNEXPECTED_EOD	***YourStreamToFunction*** stopped without passing the entire value for an attribute.
MC_INVALID_LENGTH_FOR_VR	The value(s) for one of the stream attributes was not legal for its value representation.
MC_CALLBACK_REGISTERED	This function may not be called when a callback has been registered (using **MC_Register_Callback_Function**).

Also any of the status codes which may be returned by the **MC_Set_Value** call may also be returned.

See Also

MC_Register_Callback_Function
MC_Message_To_Stream

MC_Validate_Attribute

Insures that an attribute meets DICOM validation criteria.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Validate_Attribute (
      int MessageFileID,
      unsigned long Tag
      VAL_ERR** ErrorInfo,
      VAL_LEVEL ErrorLevel
)
```

MessageFileID The identifier assigned to this object by the **MC_Open_Message**, **MC_Open_Empty_Message**, **MC_Create_Empty_File** or **MC_Create_File** function.

Tag Tag value of the attribute to validate.

ErrorInfo The address of a validation error block describing the first violation encountered is returned here if a validation violation occurs. Otherwise NULL is returned here. The block is the **VAL_ERR** type defined in "mc3msg.h":

```
typedef struct ValErr_struct
{
       unsigned long Tag;     /* Tag of attribute with validation
                                 violation */
       int MsgItemID;         /* ID of message or item object
                                 containing the attribute */
       int ValueNumber;       /* Value number involved - zero if no
                                 value involved */
       MC_STATUS      Status; /*     Validation violation  status code */

} VAL_ERR;
```

ErrorLevel The level of validation checking is specified here. Use one of the enumerated **VAL_LEVEL** types defined in "mc3msg.h":

Validation_Level1:	Report only Errors.
Validation_Level2:	Report Errors and Warnings.
Validation_Level3:	Report Errors, Warnings and Info messages.

Remarks

The **MC_Validate_Attribute** function validates an attribute contained in the object identified by *MessageFileID* to determine if its values are valid. The attribute is validated to insure that it meets the requirements of the service and command specified when the object was opened by **MC_Open_Message** or **MC_Create_File**, or when **MC_Set_Service_Command** was called for the object.

Performance Tuning: **MC_Validate_Attribute** has a lower overhead than the **MC_Validate_Message** and **MC_Validate_File** functions. When validation is only required for a select group of attributes, this function should be called.

While the validation is quite comprehensive there are limitations in validation:

- If the attribute was specified by DICOM Part 3 as being in either a user optional module, or a conditional module, it will always be treated as a Type 3 attribute (optional attribute) by **MC_Validate_Attribute**.

- The Ultrasound Image Object used in storage services has mutually exclusive Image, Overlay, and Curve information entities. The attributes defined in these entities are all treated as Type 3 (optional attributes) by **MC_Validate_Attribute**.

In any case, **MC_Validate_Attribute** is a powerful tool that can catch most all problems with an attribute. The DICOM Standard should always be the final word on whether or not an attribute satisfies the DICOM protocol.

ErrorLevel specifies the type of validation violation which will cause the **MC_Validate_Attribute** function to return a status of **MC_DOES_NOT_VALIDATE**.

NOTE: : All validation violations encountered will be logged to the MergeCOM-3 log file, regardless of the level specified in *ErrorLevel*.

A list of validation violations is created and the validation error block describing the first violation is returned at **ErrorInfo*. Subsequent violations may be accessed by using the **MC_Get_Next_Validate_Error** function. If the file validates, the status of **MC_MESSAGE_VALIDATES** is returned and NULL is returned at **ErrorInfo* . Also, the results of the validation are logged into the message log (usually merge.log) at the `T5_MESSAGE` level.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_MESSAGE_VALIDATES	The attribute contained no validation violations of the type specified in *ErrorLevel*.
MC_DOES_NOT_VALIDATE	The attribute contained at least one validation violation of the type specified in *ErrorLevel*.
MC_INVALID_MESSAGE_ID	*MessageFileID* does not identify a valid message or file object.
MC_NULL_POINTER_PARM	One or more of the pointer-type parameters was NULL.
MC_INVALID_TAG	*Tag* was not in the message.
MC_MSGFILE_ERROR	An error occurred involving the message info file for the file object. The exact error is logged in the MergeCOM-3 log file.
MC_INVALID_ITEM_ID	The message or file object contains a sequence of items attribute with an invalid item ID value.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

Validation Violations

If a status of **MC_DOES_NOT_VALIDATE** is returned, the status of the first validation violation is returned at `ErrorInfo->Status`. The violation status codes are defined in **"mcstatus.h"**. They are arranged by violation type below:

INFO MESSAGES:

MC_NO_CONDITION_FUNCTION	The attribute's DICOM type is "1C" or "2C" and no function was available to check the condition. A user-defined condition function was not correctly linked with the program.
MC_UNABLE_TO_CHECK_CONDITION	Not enough information is available to check whether or not a DICOM type "1C" or "2C" attribute is required.

WARNINGS:

MC_NOT_ONE_OF_DEFINED_TERMS	A value set for this attribute is not one of the valid defined terms assigned to this attribute.
MC_NON_SERVICE_ATTRIBUTE	The attribute is not one of those defined for the file's service. Note that private attributes will not cause this violation

ERRORS:

MC_INVALID_VR_CODE	The attribute's value representation is unknown.
MC_REQUIRED_ATTRIBUTE_MISSING	An attribute which is required for the message service has been deleted from the file object.
MC_REQUIRED_VALUE_MISSING	The attribute is required to have a value and does not.
MC_VALUE_MAY_NOT_BE_NULL	The attribute is DICOM type "1" or type "1C" and it has been encoded with a NULL value.
MC_VALUE_NOT_ALLOWED	This DICOM type "1C" or type "2C" attribute may not have a value under current conditions.
MC_TOO_FEW_VALUES	The attribute is required to have more values set.
MC_TOO_MANY_VALUES	The attribute has more values set than are allowed.
MC_INVALID_ITEM_ID	This sequence of items (SQ) attribute has an invalid value assigned to it.
MC_NOT_ONE_OF_ENUMERATED_VALUES	A value set for this attribute is not one of the valid enumerated values assigned to this attribute.
MC_INVALID_VALUE_FOR_VR	A value for this attribute does not conform to the requirements of its value representation.
MC_INVALID_CHARS_IN_VALUE	A value for this attribute does not contain valid characters for its value representation.

See Also

MC_Get_Next_Validate_Error
MC_Validate_Message
MC_Validate_File

MC_Validate_File

Insures that a file meets DICOM validation criteria.

Synopsis

```
#include "mc3media.h"

MC_STATUS MC_Validate_File (
      int FileID,
      VAL_ERR** ErrorInfo,
      VAL_LEVEL ErrorLevel
)
```

FileID — The identifier assigned to this object by the **MC_Create_Empty_File** or **MC_Create_File** function.

ErrorInfo — The address of a validation error block describing the first violation encountered is returned here if a validation violation occurs. Otherwise NULL is returned here. The block is the **VAL_ERR** type defined in "mc3msg.h":

```
typedef struct ValErr_struct
{
       unsigned long Tag;     /* Tag of attribute with validation
                                 violation */
       int MsgItemID;         /* ID of message or item object
                                 containing the attribute */
       int ValueNumber;       /* Value number involved - zero if no
                                 value involved */
       MC_STATUS      Status; /*     Validation violation  status code */

} VAL_ERR;
```

ErrorLevel — The level of validation checking is specified here. Use one of the enumerated **VAL_LEVEL** types defined in "mc3msg.h":

Validation_Level1:	Report only Errors.
Validation_Level2:	Report Errors and Warnings.
Validation_Level3:	Report Errors, Warnings and Info messages.

Remarks

The **MC_Validate_File** function validates the file identified by *FileID* to determine that required attributes are present and that all attribute values are valid. The file is validated to insure that it meets the requirements of the service and command specified when the message was opened with **MC_Create_File** or when **MC_Set_Service_Command** was called for the file object. If the file was opened with **MC_Create_Empty_File**, the **MC_Set_Service_Command** function must be called to specify the service and command to validate against.

While the validation is quite comprehensive there are limitations in validating files:

- If the attribute was specified by DICOM Part 3 as being in either a user optional module, or a conditional module, it will always be treated as a Type 3 attribute (optional attribute) by **MC_Validate_File**.

- The Ultrasound Image Object used in storage services has mutually exclusive Image, Overlay, and Curve information entities. The attributes defined in these entities are all treated as Type 3 (optional attributes) by **MC_Validate_File**.

- **MC_Validate_File** does not validate properly for application profiles that modify the standard content of a message.

In any case, **MC_Validate_File** is a powerful tool that can catch most all problems with a file. The DICOM Standard should always be the final word on whether or not a file satisfies the DICOM protocol.

ErrorLevel specifies the type of validation violation which will cause the **MC_Validate_File** function to return a status of **MC_DOES_NOT_VALIDATE**.

NOTE: : All validation violations encountered will be logged to the MergeCOM-3 log file, regardless of the level specified in *ErrorLevel*.

A list of validation violations is created and the validation error block describing the first violation is returned at **ErrorInfo*. Subsequent violations may be accessed by using the **MC_Get_Next_Validate_Error** function. If the file validates, the status of **MC_MESSAGE_VALIDATES** is returned and NULL is returned at **ErrorInfo* . Also, the results of the validation are logged into the message log (usually merge.log) at the `T5_MESSAGE` level.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_MESSAGE_VALIDATES	The file object contained no validation violations of the type specified in *ErrorLevel*.
MC_DOES_NOT_VALIDATE	The file object contained at least one validation violation of the type specified in *ErrorLevel*.
MC_INVALID_FILE_ID	*FileID* does not identify a valid file object.
MC_NULL_POINTER_PARM	One or more of the pointer-type parameters was NULL.
MC_MSGFILE_ERROR	An error occurred involving the message info file for the file object. The exact error is logged in the MergeCOM-3 log file.
MC_INVALID_ITEM_ID	The file object contains a sequence of items attribute with an invalid item ID value.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

Validation Violations

If a status of **MC_DOES_NOT_VALIDATE** is returned, the status of the first validation violation is returned at `ErrorInfo->Status`. The violation status codes are defined in **"mcstatus.h"**. They are arranged by violation type below:

INFO MESSAGES:	**MC_NO_CONDITION_FUNCTION**	The attribute's DICOM type is "1C" or "2C" and no function was available to check the condition. A user-defined condition function was not correctly linked with the program.

	MC_UNABLE_TO_CHECK_CONDITION	Not enough information is available to check whether or not a DICOM type "1C" or "2C" attribute is required.
WARNINGS:	**MC_NOT_ONE_OF_DEFINED_TERMS**	A value set for this attribute is not one of the valid defined terms assigned to this attribute.
	MC_NON_SERVICE_ATTRIBUTE	The attribute is not one of those defined for the file's service. Note that private attributes will not cause this violation
ERRORS:	**MC_INVALID_VR_CODE**	The attribute's value representation is unknown.
	MC_REQUIRED_ATTRIBUTE_MISSING	An attribute which is required for the message service has been deleted from the file object.
	MC_REQUIRED_VALUE_MISSING	The attribute is required to have a value and does not.
	MC_VALUE_MAY_NOT_BE_NULL	The attribute is DICOM type "1" or type "1C" and it has been encoded with a NULL value.
	MC_VALUE_NOT_ALLOWED	This DICOM type "1C" or type "2C" attribute may not have a value under current conditions.
	MC_TOO_FEW_VALUES	The attribute is required to have more values set.
	MC_TOO_MANY_VALUES	The attribute has more values set than are allowed.
	MC_INVALID_ITEM_ID	This sequence of items (SQ) attribute has an invalid value assigned to it.
	MC_NOT_ONE_OF_ENUMERATED_VALUES	A value set for this attribute is not one of the valid enumerated values assigned to this attribute.
	MC_INVALID_VALUE_FOR_VR	A value for this attribute does not conform to the requirements of its value representation.
	MC_INVALID_CHARS_IN_VALUE	A value for this attribute does not contain valid characters for its value representation.

See Also

MC_Get_Next_Validate_Error **MC_Validate_Message**
MC_Validate_Attribute

MC_Validate_Message

Insures that a message meets DICOM validation criteria.

Synopsis

```
#include "mc3msg.h"

MC_STATUS MC_Validate_Message (
      int MessageID,
      VAL_ERR** ErrorInfo,
      VAL_LEVEL ErrorLevel
)
```

MessageID — The message object's identification number

ErrorInfo — The address of a validation error block describing the first violation encountered is returned here if a validation violation occurs. Otherwise NULL is returned here. The block is the **VAL_ERR** type defined in "mc3msg.h":

```
typedef struct ValErr_struct
{
       unsigned long Tag;     /* Tag of attribute with validation
                                 violation */
       int MsgItemID;         /* ID of message or item object
                                 containing the attribute */
       int ValueNumber;       /* Value number involved - zero if no value
                                 involved */
       MC_STATUS Status;      /* Validation violation  status code */

} VAL_ERR;
```

ErrorLevel — The level of validation checking is specified here. Use one of the enumerated **VAL_LEVEL** types defined in "mc3msg.h":

Validation_Level1:	Report only Errors.
Validation_Level2:	Report Errors and Warnings.
Validation_Level3:	Report Errors, Warnings and Info messages.

Remarks

MC_Validate_Message validates the message identified by *MessageID* to determine that required attributes are present and that all attribute values are valid. The message is validated to insure that it meets the requirements of the service and command specified when the message was opened by **MC_Open_Message**. If the message was opened with **MC_Open_Empty_Message**, the **MC_Set_Service_Command** function must be called to specify the service and command to validate against.

While the validation is quite comprehensive there are limitations in validating messages opened for composite services:

- If the attribute was specified by DICOM Part 3 as being in either a user optional module, or a conditional module, it will always be treated as a Type 3 attribute (optional attribute) by MC_Validate_Message.

- The Nuclear Medicine and Ultrasound Image Objects used in storage services have mutually exclusive Image, Overlay, and Curve information entities. The attributes defined in these entities are all treated as Type 3 (optional attributes) by MC_Validate_Message.

- For normalized services using the N-EVENT-REPORT command, the actual contents of a message specified in DICOM Part 4 are dependent on the Event Type ID being communicated. All attributes used for all Event Type ID's are treated as Type 3 by MC_Validate_Message.

- Unique keys used within C-FIND-RQ messages for each of the Query/Retrieve service classes are not validated properly because the differences in what is required for request and response messages. The C-FIND-RSP messages are validated properly. An error is also reported when a unique key is set to NULL. This is an invalid error for the unique keys at the level that is being queried.

In any case, MC_Validate_Message is a powerful tool that can catch most all problems with a message. The DICOM Standard should always be the final word on whether or not a message satisfies the DICOM protocol.

ErrorLevel specifies the type of validation violation which will cause the **MC_Validate_Message** function to return a status of **MC_DOES_NOT_VALIDATE**. Note, however, that all validation violations encountered will be logged to the MergeCOM-3 log file, regardless of the level specified in *ErrorLevel*.

A list of validation violations is created and the validation error block describing the first violation is returned at **ErrorInfo*. Subsequent violations may be accessed by using the **MC_Get_Next_Validate_Error** function. If the message validates, the status of **MC_MESSAGE_VALIDATES** is returned and NULL is returned at **ErrorInfo* . Also, the results of the validation are logged into the message log (usually merge.log) at the `T5_MESSAGE` level.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_MESSAGE_VALIDATES	The message contained no validation violations of the type specified in *ErrorLevel*.
MC_DOES_NOT_VALIDATE	The message contained at least one validation violation of the type specified in *ErrorLevel*.
MC_INVALID_MESSAGE_ID	*MessageID* does not identify a valid message object.
MC_NULL_POINTER_PARM	One or more of the pointer-type parameters was NULL.
MC_MSGFILE_ERROR	An error occurred involving the message info file for the message. The exact error is logged in the MergeCOM-3 log file.
MC_INVALID_ITEM_ID	The message contains a sequence of items attribute with an invalid item ID value.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

Validation Violations

If a status of **MC_DOES_NOT_VALIDATE** is returned, the status of the first validation violation is returned at `ErrorInfo->Status`. The violation status codes are defined in **"mcstatus.h"**. They are arranged by violation type below:

INFO MESSAGES:	**MC_NO_CONDITION_FUNCTION**	The attribute's DICOM type is "1C" or "2C" and no function was available to check the condition. A user-defined condition function was not correctly linked with the program.
	MC_UNABLE_TO_CHECK_CONDITION	Not enough information is available to check whether or not a DICOM type "1C" or "2C" attribute is required.
WARNINGS:	**MC_NOT_ONE_OF_DEFINED_TERMS**	A value set for this attribute is not one of the valid defined terms assigned to this attribute.
	MC_NON_SERVICE_ATTRIBUTE	The attribute is not one of those defined for the message's service. Note that private attributes will not cause this violation
ERRORS:	**MC_INVALID_VR_CODE**	The attribute's value representation is unknown.
	MC_REQUIRED_ATTRIBUTE_MISSING	An attribute which is required for the message service has been deleted from the message object.
	MC_REQUIRED_VALUE_MISSING	The attribute is required to have a value and does not.
	MC_VALUE_MAY_NOT_BE_NULL	The attribute is DICOM type "1" or type "1C" and it has been encoded with a NULL value.
	MC_VALUE_NOT_ALLOWED	This DICOM type "1C" or type "2C" attribute may not have a value under current conditions.
	MC_TOO_FEW_VALUES	The attribute is required to have more values set.
	MC_TOO_MANY_VALUES	The attribute has more values set than are allowed.
	MC_INVALID_ITEM_ID	This sequence of items (SQ) attribute has an invalid value assigned to it.
	MC_NOT_ONE_OF_ENUMERATED_VALUES	A value set for this attribute is not one of the valid enumerated values assigned to this attribute.
	MC_INVALID_VALUE_FOR_VR	A value for this attribute does not conform to the requirements of its value representation.
	MC_INVALID_CHARS_IN_VALUE	A value for this attribute does not contain valid characters for its value representation.

See Also

MC_Get_Next_Validate_Error **MC_Validate_Attribute**

MC_Wait_For_Association
MC_Wait_For_Association_On_Port

Waits for an association request from a remote DICOM application

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Wait_For_Association (
      char* ServiceList,
      int Timeout,
      int* ApplicationID,
      int* AssociationID
)

MC_STATUS MC_Wait_For_Association_On_Port (
      char* ServiceList,
      int Timeout,
      int ApplicationID,
      int Port,
      int* AssociationID
)
```

ServiceList	Name of a service list in the MergeCOM-3 configuration file.
Timeout	The max time (in seconds) to wait for an association request to arrive. A value of zero(0) means "check one time" and a value of minus one (-1) means "wait forever".
ApplicationID	The identification number of a previously registered application is returned here, or the application ID to negotiate is supplied here.
AssociationID	The identification number of an association object is returned here.
Port	Port number to listen on.

Remarks

MC_Wait_For_Association and **MC_Wait_For_Association_On_Port** are used to wait for a remote DICOM application to make a request for an association with one of the applications previously registered using the **MC_Register_Application** function. **MC_Wait_For_Association** will negotiate for all applications registered with **MC_Register_Application**, and **MC_Wait_For_Association_On_Port** will only negotiate associations for the specific Application passed to the call.

If *Timeout* is a positive number MergeCOM-3 will wait no longer than *Timeout* seconds for an association request. If *Timeout* is negative, MergeCOM-3 will not return until an association request has arrived or an error occurs. If *Timeout* is zero, MergeCOM-3 will check one time for the arrival of an association request and return. A status of MC_TIMEOUT is returned if no valid association request is received in the specified time period.

ServiceList is the name of a section in the MergeCOM-3 configuration file which specifies which DICOM services the registered applications are willing to provide. If a request for any one of these services is received, this function returns MC_NORMAL_COMPLETION and sets **AssociationID* to the identification number of an association object. *AssociationID* is then used in other association function calls to identify this negotiated connection.

Normally, if an association request is received for an application whose title is not registered or is not specified on a call to **MC_Wait_For_Association_On_Port**,

MergeCOM-3 rejects the association request and continues to wait for a valid association request. Optionally, "ACCEPT_ANY_APPLICATION_TITLE = YES" may be specified in the MergeCOM-3 system profile. In that case, requests for associations with an unregistered application will be given to the first registered application entity title which has been registered with **MC_Register_Application**..

MergeCOM-3 validates each received association request based on the protocol rules of DICOM and on configurable specifications contained in the MergeCOM-3 system profile. An association request is rejected if it does not validate.

Upon successful completion (MC_NORMAL_COMPLETION returned) of **MC_Wait_For_Association**, **ApplicationID* is set to the identification number of a registered application. In all other cases, both **ApplicationID* and **AssociationID* are not valid.

Upon successful completion, the caller may examine any expected extended negotiation information by using the **MC_Get_Negotiation_Info** function. The caller must respond to the association requestor by using either **MC_Accept_Association** or **MC_Reject_Association** before any message exchange can occur over the association.

The listen port can be changed!

The port that **MC_Wait_For_Association** listens on can be changed while an application is running by calling **MC_Set_Int_Config_Value** to set the TCPIP_LISTEN_PORT configuration option. The next call to **MC_Wait_For_Association** after this configuration option has changed will stop listening on the previous listen port and start listening on the newly configured listen port.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_TIMEOUT	The *Timeout* period expired.
MC_NEGOTIATION_ABORTED	An incoming association was aborted during negotiation. Normally this situation is handled by retrying the **MC_Wait_For_Association** call.
MC_ASSOCIATION_REJECTED	An incoming association was aborted because no services were acceptable.
MC_SYSTEM_CALL_INTERRUPTED	The system call used to wait for an association request was interrupted by a signal. Normally this situation is handled by retrying the **MC_Wait_For_Association** call.
MC_NULL_POINTER_PARM	*AssociationID*, *ApplicationID* or *ServiceList* parameter was NULL.
MC_NO_APPLICATIONS_REGISTERED	No applications have been registered using **MC_Register_Application.**
MC_INVALID_SERVICE_LIST_NAME	*ServiceList* points at a null string.
MC_CONFIG_INFO_MISSING	Could not access *ServiceList* configuration parameters.

MC_CONFIG_INFO_ERROR	The *ServiceList* contained too many services.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Open_Association
MC_Open_Secure_Association
MC_Get_Negotiation_Info
MC_Accept_Association
MC_Reject_Association
MC_Wait_For_Secure_Association
MC_Close_Listen_Port

MC_Wait_For_Secure_Association
MC_Wait_For_Secure_Association_On_Port

Waits for an association request from a remote DICOM application using a secure socket connection.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Wait_For_Secure_Association (
      char* ServiceList,
      int Timeout,
      int* ApplicationID,
      int* AssociationID,
      SecureSocketFunctions* SS_functions,
      void* SS_context
)

MC_STATUS MC_Wait_For_Secure_Association_On_Port (
      char* ServiceList,
      int Timeout,
      int ApplicationID,
      int Port,
      int* AssociationID,
      SecureSocketFunctions* SS_functions,
      void* SS_context
)
```

ServiceList	Name of a service list in the MergeCOM-3 configuration file.
Timeout	The max time (in seconds) to wait for an association request to arrive. A value of zero(0) means "check one time" and a value of minus one (-1) means "wait forever".
ApplicationID	The identification number of a previously registered application is returned here, or the application ID to negotiate is supplied here.
AssociationID	The identification number of an association object is returned here.
Port	Port number to listen on.
SS_functions	Pointer to a structure containing functions that will be called by MergeCOM-3 while processing network I/O over the secure connection. (see below)
SS_context	An optional pointer to application-specific data that MergeCOM-3 passes to the functions declared in *SS_functions*.

Remarks

Use **MC_Wait_For_Secure_Association** and **MC_Wait_For_Secure_Association_On_Port** to wait for a remote DICOM application to make a request for an association with one of the applications previously registered using the **MC_Register_Application** function. **MC_Wait_For_Secure_Association** will negotiate for all applications registered with **MC_Register_Application**, and **MC_Wait_For_Secure_Association_On_Port** will only negotiate associations for the specific Application passed to the call.

If *Timeout* is a positive number MergeCOM-3 will wait no longer than *Timeout* seconds for an association request to arrive. If *Timeout* is negative, MergeCOM-3 will not return until an association request has arrived or an error occurs. If *Timeout* is zero, MergeCOM-3 will check one time for the arrival of an association request and return. A

status of MC_TIMEOUT is returned if no valid association request is received in the specified time period.

ServiceList is the name of a section in the MergeCOM-3 configuration file which specifies which DICOM services the registered applications are willing to provide. If a request for any one of these services is received, this function returns MC_NORMAL_COMPLETION and sets **AssociationID* to the identification number of an association object. *AssociationID* is then used in other association function calls to identify this negotiated connection.

Normally, if an association request is received for an application whose title is not registered or is not specified on a call to **MC_Wait_For_Secure_Association_On_Port**, MergeCOM-3 rejects the association request and continues to wait for a valid association request. Optionally, "ACCEPT_ANY_APPLICATION_TITLE = YES" may be specified in the MergeCOM-3 system profile. In that case, requests for associations with an unregistered application will be given to the first registered application entity title which has been registered with **MC_Register_Application**.

MergeCOM-3 validates each received association request based on the protocol rules of DICOM and on configurable specifications contained in the MergeCOM-3 system profile. An association request is rejected if it does not validate.

Upon successful completion (MC_NORMAL_COMPLETION returned) of **MC_Wait_For_Secure_Association**, **ApplicationID* is set to the identification number of a registered application. In all other cases, both **ApplicationID* and **AssociationID* are not valid.

Upon successful completion, the caller may examine any expected extended negotiation information by using the **MC_Get_Negotiation_Info** function. The caller must respond to the association requestor by using either **MC_Accept_Association** or **MC_Reject_Association** before any message exchange can occur over the association.

The listen port can be changed!

The port that **MC_Wait_For_Secure_Association** listens on can be changed while an application is running by calling **MC_Set_Int_Config_Value** to set the TCPIP_LISTEN_PORT configuration option. The next call to **MC_Wait_For_Secure_Association** after this configuration option has changed will stop listening on the previous listen port and start listening on the newly configured listen port.

SecureSocketFunctions

When using the **MC_Wait_For_Secure_Association** or **MC_Wait_For_Secure_Association_On_Port** calls, MergeCOM-3 establishes a TCP/IP connection and then calls the functions provided by the *SS_functions* parameter to establish the secure connection and to pass data through the secure connection. The *SS_functions* are responsible for sending and receiving all data through the secure connection, thus allowing them to do so using a secure protocol such as Secure Socket Layer(SSL). MergeCOM-3 closes the underlying TCP/IP connection when all association processing has completed and after it calls the **SS_Session_Shutdown** callback.

The **SecureSocketFunctions** structure is declared in mergecom.h as follows:

```
typedef struct MC_Secure_Socket_Functions_Struct {
      SS_STATUS (NOEXP_FUNC *SS_Session_Start)
                (int          SocketToUse,
                CONN_TYPE     ConnectionType,
                void*         ApplicationContext,
                void**        SecurityContext);
```

```
        SS_STATUS (NOEXP_FUNC *SS_Read)
                  void*          SScontext,
                  void*          ApplicationContext,
                  char*          Buffer,
                  int            BytesToRead,
                  void*          BytesRead,
                  int            Timeout);
        SS_STATUS (NOEXP_FUNC *SS_Write)
                  (void*         SScontext,
                  void*          ApplicationContext,
                  char*          Buffer,
                  int            BytesToWrite,
                  int*           BytesWritten,
                  int            Timeout);
        void (NOEXP_FUNC *SS_Session_Shutdown)
                  (void*         SScontext,
                  void*          ApplicationContext);
} SecureSocketFunctions;
```

You must provide valid function pointers for each of the four fields in the **SecureSocketFunctions** structure.

SS_Session_Start

MergeCOM-3 calls the **SS_Session_Start** function just after it has accepted the TCP/IP connect request made by the remote host. The *SocketToUse* parameter contains the socket assigned to the connection. Please note that the connection is non-blocking. The *ApplicationContext* parameter is the presented by the *SS_context* parameter of the **MC_Wait_For_Secure_Association** call. *ConnectionType* will be the manifest constant ACCEPTOR_CONNECTION if the **SS_Session_Start** function is called as a result of a **MC_Wait_For_Secure_Association** call. (It will be REQUESTER_CONNECTION if it is called as a result of a **MC_Open_Secure_Association** call.)

The **SS_Session_Start** function is responsible for establishing a secure connection using the socket provided. It is assumed, but not required, that the connection will be a Secure Socket Layer (SSL) or Transport Layer Socket (TLS) connection. A pointer to a context block should be returned at **SecurityContext* if the secure connection is established. MergeCOM-3 will provide this pointer when it calls the other callback functions.

SS_Session_Start must return SS_NORMAL_COMPLETION if the secure connection was successfully established, otherwise SS_ERROR must be returned. If it returns SS_ERROR, the TCP/IP connection will be closed and the **MC_Wait_For_Secure_-Association** call will return a status of **MC_NEGOTIATION_ABORTED**.

SS_Session_Shutdown

MergeCOM-3 calls the **SS_Session_Shutdown** function when the association is aborted or closed. It is the responsibility of the **SS_Session_Shutdown** function to gracefully close the secure network connection. MergeCOM-3 closes the TCP/IP socket connection after calling **SS_Session_Shutdown**. The *ApplicationContext* parameter is that presented by the *SS_context* parameter of the **MC_Wait_For_Secure_Association** call, and the *SScontext* parameter is that previously returned by the **SS_Session_Start** function.

SS_Read

MergeCOM-3 calls the **SS_Read** function whenever it needs association data from the secure connection. The *ApplicationContext* parameter is that presented by the *SS_context* parameter of the **MC_Wait_For_Secure_Association** call, and the *SScontext* parameter is that previously returned by the **SS_Session_Start** function. It is the responsibility of

the **SS_Read** function to retrieve into *Buffer* the number of unencrypted data bytes specified by *BytesToRead*. The actual number of bytes placed in the *Buffer* is returned at **BytesRead*. SS_NORMAL_COMPLETION must be returned if the read request was satisfied.

If the **SS_Read** function cannot retrieve *BytesToRead* bytes in *Timeout* seconds, it must return SS_TIMEOUT. Please note that the socket connection passed by MergeCOM-3 to the **SS_Session_Start** function is non-blocking. (Note that if SS_TIMEOUT is returned, an outstanding MC_Read_Message call will return MC_TIMEOUT.)

If it is determined that the secure socket connection has closed, or that the underlying transport has closed, SS_SESSION_CLOSED must be returned. This should only occur during a DICOM association if the remote host aborted the association. If a fatal error occurs while processing the read request, SS_ERROR must be returned. (Note that if SS_SESSION_CLOSED or SS_ERROR is returned, an outstanding MC_Read_Message call will return MC_NETWORK_SHUT_DOWN.)

SS_Write

MergeCOM-3 calls the **SS_Write** function whenever it needs to send association data over the secure connection. The *ApplicationContext* parameter is that presented by the *SS_context* parameter of the **MC_Wait_For_Secure_Association** call, and the *SScontext* parameter is that previously returned by the **SS_Session_Start** function. It is the responsibility of the **SS_Write** function to send *BytesToWrite* bytes of the data in *Buffer* over the secure connection, returning the number of bytes actually written at **BytesWritten*. SS_NORMAL_COMPLETION must be returned if the write request was satisfied.

If the **SS_ Write** function cannot send *BytesToWrite* bytes in *Timeout* seconds, it must return SS_TIMEOUT. Please note that the socket connection passed by MergeCOM-3 to the **SS_Session_Start** function is non-blocking.

If a fatal error occurs while processing the write request, SS_ERROR must be returned. (Note that if SS_ERROR is returned, an outstanding MC_Send_Request_Message or MC_Send_Response_Message call will return MC_SYSTEM_ERROR.)

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_TIMEOUT	The *Timeout* period expired.
MC_NEGOTIATION_ABORTED	An incoming association was aborted during negotiation. Normally this situation is handled by retrying the **MC_Wait_For_-Secure_Association** call. This will also be returned if the **SS_Session_Start** callback function returned an SS_ERROR status.
MC_ASSOCIATION_REJECTED	An incoming association was aborted because no services were acceptable.
MC_SYSTEM_CALL_INTERRUPTED	The system call used to wait for an association request was interrupted by a signal. Normally this situation is handled by

	retrying the **MC_Wait_For_Secure_-Association** call.
MC_NULL_POINTER_PARM	*AssociationID, ApplicationID, SS_functions* or *ServiceList* parameter was NULL.
MC_NULL_VALUE	One or more of the function parameters within *SS_functions* was NULL.
MC_NO_APPLICATIONS_REGISTERED	No applications have been registered using **MC_Register_Application.**
MC_INVALID_SERVICE_LIST_NAME	*ServiceList* points at a null string.
MC_CONFIG_INFO_MISSING	Could not access *ServiceList* configuration parameters.
MC_CONFIG_INFO_ERROR	The *ServiceList* contained too many services.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.

See Also

MC_Open_Association
MC_Open_Secure_Association
MC_Get_Negotiation_Info
MC_Accept_Association
MC_Reject_Association
MC_Wait_For_Association
MC_Close_Listen_Port

MC_Write_File
MC_Write_File_By_Callback

Writes a file object to media.

Synopsis

```
#include "mc3media.h"

MC_STATUS MC_Write_File(
      int FileID,
      int NumBytes,
      void* UserInfo,
      MC_STATUS (*YourToMediaFunction()
)

MC_STATUS MC_Write_File_By_Callback(
      int ApplicationID,
      int FileID,
      int NumBytes,
      void* UserInfo,
      MC_STATUS (*YourToMediaFunction()
)
```

ApplicationID	The identifier for an application object assigned by the **MC_Register_Application** function.
FileID	The identifier assigned to this object by the **MC_Create_Empty_File** or **MC_Create_File** function.
NumBytes	The attribute (FFFC,FFFC) will be added to the file object and given a length such that the total length of the file being written is a multiple of *NumBytes*. If *NumBytes* is set to 0, there will be no padding of the file. *NumBytes* must be an even number.
UserInfo	Address of data which will be passed on to *YourToMediaFunction* each time it is called. This may be NULL.
YourToMediaFunction	Name of a function which will be called repeatedly to provide blocks of DICOM file data.

The function must be prototyped as follows:

```
MC_STATUS YourToMediaFunction (
      char* CBfilename,
      void* CBuserInfo,
      int CBdataSize,
      void* CBdataBuffer,
      int CBisFirst,
      int CBisLast
)
```

CBfilename	The filename associated with the file object by the **MC_Create_Empty_File** or **MC_Create_File** function.
CBuserInfo	Address of data which is being passed from the **MC_Write_File** function. This may be NULL.
CBdataSize	The number of bytes of file data being provided to you in

	CBdataBuffer.
CBdataBuffer	The buffer containing file data from the file object.
CBisFirst	Is TRUE (not zero) when MergeCOM-3 is providing the first block of file data.
CBisLast	Is TRUE (not zero) when MergeCOM-3 is providing the last block of file data.

Remarks

MC_Write_File and **MC_Write_File_By_Callback** request that the contents of the file object identified by *FileID* be put in a form specified by the DICOM standard for files. The file is passed to the user in blocks by calling ***YourToMediaFunction*** repeatedly until the entire file has been transferred.

MC_Write_File and **MC_Write_File_By_Callback** can pass data to ***YourToMediaFunction*** by specifying the data's address in *UserInfo*. MergeCOM-3 passes the address to ***YourToMediaFunction*** in *CBuserInfo* each time it is called. *UserInfo* may be NULL.

MC_Write_File_By_Callback is identical to **MC_Write_File** except that it can be used in conjunction with **MC_Register_Callback_Function** to have pixel data supplied to the function as it is being streamed out.

NOTE: If the file object is a DICOMDIR, **MC_Write_File** and **MC_Write_File_By_Callback** will resolve the directory record offsets within the object before passing the user the file data.

NOTE: If the file contains "group length" attributes (i.e. attributes with tags of the form gggg0000: any group, element zero), **MC_Write_File** and **MC_Write_File_By_Callback** will automatically calculate the group length value when supplying it to ***YourToMediaFunction***.

NOTE: MC_Write_File and **MC_Write_File_By_Callback** will format the byte stream passed to *YourToMediaFunction* in the attribute transfer syntax UID (0002, 0010).

NOTE: **MC_Write_File** and **MC_Write_File_By_Callback** will automatically fill in two group 2 attributes within the file meta information - **if you have not set them**:

- The Implementation Class UID (0002,0012) will be filled in with the value set for the **IMPLEMENTATION_CLASS_UID** configuration value in the mergecom.pro file.
- The Implementation Version Name (0002,0013) will be filled in with the value set for the **IMPLEMENTATION_VERSION_NAME** configuration value in the mergecom.pro file.

YourToMediaFunction

YourToMediaFunction will be called repeatedly to pass blocks of data to it. MergeCOM-3 sets *CBisFirst* to TRUE(non-zero) the first time it calls ***YourToMediaFunction*** for this attribute and it sets *CBisLast* to TRUE(non-zero) when it calls ***YourToMediaFunction*** with the final block of file data.

CBdataBuffer is set to the address of a buffer containing the file data block, and *CBdataSize* is be set to the number of bytes placed at *CBdataBuffer*.

Return Value

One of these enumerated **MC_STATUS** codes defined in **"mcstatus.h"**:

Value	Meaning
MC_NORMAL_COMPLETION	The function completed normally.
MC_NULL_POINTER_PARM	One or more of the pointer-type parameters was NULL.
MC_INVALID_APPLICATION_ID	The *ApplicationID* value is not a valid application object ID.
MC_INVALID_FILE_ID	The *FileID* value is not a valid file object ID.
MC_INVALID_PAD	The *NumBytes* parameter contained too large of a number or an odd number.
MC_SYSTEM_ERROR	An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the MergeCOM-3 log file.
MC_MESSAGE_EMPTY	The file has no attributes in it.
MC_CALLBACK_CANNOT_COMPLY	***YourToMediaFunction*** returned a value other than MC_NORMAL_COMPLETION.
MC_INVALID_TRANSFER_SYNTAX	The transfer syntax is improperly specified or is not specified in the DICOM File Meta Information attributes.
MC_TEMPFILE_ERROR	A value for an attribute is stored temporarily on file and an I/O error occurred.

See Also

MC_Open_File
MC_Create_File
MC_Create_Empty_File

Appendix A: DICOM Conformance Statement

Detailed below is a conformance statement for the MergeCOM-3 Advanced Tool Kit. Since the tool kit is *not* an application, this conformance statement only gives an outline of the DICOM services it supports. Responsibility for full DICOM conformance to particular SOP classes rests with the application developer, since many of the requirements for such conformance lie outside the realm of the tool kit. For example: the high level behavior of Query/Retrieve service class SCUs and SCPs as defined in Part 4 of the DICOM standard, is implemented by the application developer in conjunction with the tool kit functionality.

The tool kit conformance statement that follows must be modified for individual applications to fully describe the implementation of specific SOP classes.

Tool Kit conformance statement

Introduction

This is a conformance statement for the MergeCOM-3 Advanced Tool Kit which provides support for the DICOM services shown in Table A-1 below.

(Application conformance statements will specify introductory information about the DICOM services which are implemented by the application.)

Service Class
BASIC_ANNOTATION_BOX
BASIC_COLOR_IMAGE_BOX
BASIC_FILM_BOX
BASIC_FILM_SESSION
BASIC_GRAYSCALE_IMAGE_BOX
BREAST_IMAGING_RPI_QUERY
CARDIAC_RPI_QUERY
CHEST_CAD_SR
DETACHED_INTERP_MANAGEMENT
DETACHED_PATIENT_MANAGEMENT
DETACHED_RESULTS_MANAGEMENT
DETACHED_STUDY_MANAGEMENT
DETACHED_VISIT_MANAGEMENT
DICOMDIR
ENHANCED_CT_IMAGE
ENHANCED_MR_IMAGE
G_P_PERFORMED_PROCEDURE_STEP
G_P_SCHEDULED_PROCEDURE_STEP
G_P_WORKLIST
GENERAL_RPI_QUERY
HANGING_PROTOCOL
HANGING_PROTOCOL_FIND
HANGING_PROTOCOL_MOVE
IMAGE_OVERLAY_BOX_RETIRED
INSTANCE_AVAIL_NOTIFICATION
KEY_OBJECT_SELECTION_DOC
MAMMOGRAPHY_CAD_SR
MEDIA_CREATION_MANAGEMENT
MODALITY_WORKLIST_FIND
MR_SPECTROSCOPY
PATIENT_ROOT_QR_FIND
PATIENT_ROOT_QR_GET
PATIENT_ROOT_QR_MOVE
PATIENT_STUDY_ONLY_QR_FIND
PATIENT_STUDY_ONLY_QR_GET
PATIENT_STUDY_ONLY_QR_MOVE
PERFORMED_PROCEDURE_STEP
PERFORMED_PROCEDURE_STEP_NOTIFY
PERFORMED_PROCEDURE_STEP_RETR
PRESENTATION_LUT
PRINTER

PRINTER_CONFIGURATION
PRINT_JOB
PRINT_QUEUE_MANAGEMENT
PROCEDURAL_EVENT_LOGGING
PROCEDURE_LOG
PULL_PRINT_REQUEST
REFERENCED_IMAGE_BOX
SC_MULTIFRAME_GRAYSCALE_BYTE
SC_MULTIFRAME_GRAYSCALE_WORD
SC_MULTIFRAME_SINGLE_BIT
SC_MULTIFRAME_TRUE_COLOR
SPATIAL_FIDUCIALS
SPATIAL_REGISTRATION
STANDARD_BASIC_TEXT_SR
STANDARD_COMPREHENSIVE_SR
STANDARD_CR
STANDARD_CT
STANDARD_CURVE
STANDARD_DX_PRESENT
STANDARD_DX_PROCESS
STANDARD_ENHANCED_SR
STANDARD_HARDCOPY_COLOR
STANDARD_HARDCOPY_GRAYSCALE
STANDARD_ECHO
STANDARD_IO_PRESENT
STANDARD_IO_PROCESS
STANDARD_MG_PRESENT
STANDARD_MG_PROCESS
STANDARD_MODALITY_LUT
STANDARD_MR
STANDARD_NM
STANDARD_OPHTHALMIC_16_BIT
STANDARD_OPHTHALMIC_8_BIT
STANDARD_OVERLAY
STANDARD_PET
STANDARD_PET_CURVE
STANDARD_PRINT_STORAGE
STANDARD_RT_BEAMS_TREAT
STANDARD_RT_BRACHY_TREAT
STANDARD_RT_DOSE
STANDARD_RT_IMAGE
STANDARD_RT_PLAN
STANDARD_RT_STRUCTURE_SET
STANDARD_RT_TREAT_SUM
STANDARD_SEC_CAPTURE
STANDARD_US
STANDARD_US_MF
STANDARD_US_MF_RETIRED
STANDARD_US_RETIRED
STANDARD_VIDEO_ENDOSCOPIC
STANDARD_VIDEO_MICROSCOPIC
STANDARD_VIDEO_PHOTOGRAPHIC

STANDARD_VL_ENDOSCOPIC
STANDARD_VL_MICROSCOPIC
STANDARD_VL_PHOTOGRAPHIC
STANDARD_VL_SLIDE_MICROSCOPIC
STANDARD_VOI_LUT
STANDARD_WAVEFORM_12_LEAD_ECG
STANDARD_WAVEFORM_AMBULATORY_ECG
STANDARD_WAVEFORM_BASIC_VOICE_AU
STANDARD_WAVEFORM_CARDIAC_EP
STANDARD_WAVEFORM_GENERAL_ECG
STANDARD_WAVEFORM_HEMODYNAMIC
STANDARD_XRAY_ANGIO
STANDARD_XRAY_ANGIO_BIPLANE
STANDARD_XRAY_RF
STEREOMETRIC_RELATIONSHIP
STORAGE_COMMITMENT_PULL
STORAGE_COMMITMENT_PUSH
STUDY_COMPONENT_MANAGEMENT
STUDY_CONTENT_NOTIFICATION
STUDY_ROOT_QR_FIND
STUDY_ROOT_QR_GET
STUDY_ROOT_QR_MOVE
VOI_LUT_BOX
BASIC_COLOR_PRINT_MANAGEMENT
BASIC_GRAYSCALE_PRINT_MANAGEMENT
DETACHED_PATIENT_MANAGEMENT_META
DETACHED_RESULTS_MANAGEMENT_META
REF_COLOR_PRINT_MANAGEMENT
REF_GRAYSCALE_PRINT_MANAGEMENT
STUDY_MANAGEMENT

Table A-1: DICOM Services supported by the MergeCOM-3 Advanced Tool Kit

Implementation model

MergeCOM-3 Advanced supplies tools for the creation of applications which provide the DICOM services shown in Table A-1 above.

Application data flow diagram

The MergeCOM-3 Advanced Tool Kit provides the core functionality required to facilitate data flow between SCUs and SCPs.

(Application conformance statements will include a data flow diagram. An example is shown below for a simple Storage Service Class SCP.)

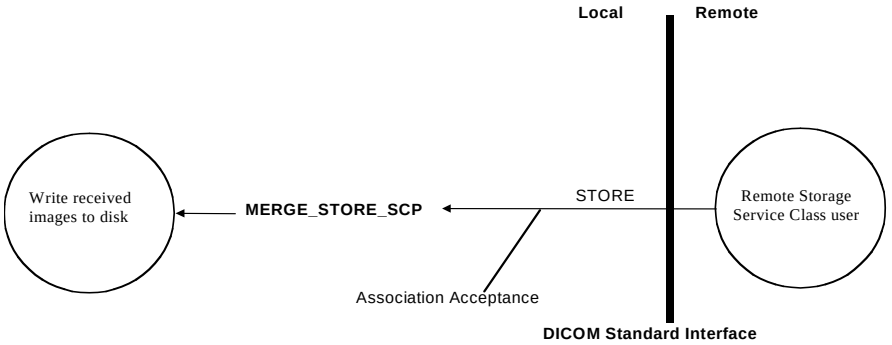

Figure A-1: MERGE_STORE_SCP application data flow diagram

Functional definition of Application Entity (AE)

(Application conformance statements will fill in this section with a general specification of the functions to be performed by SCU or SCP.)

Sequencing of real-world activities

(Application conformance statements will fill in this section with any associated sequence of real-world activities. For example, a Storage Service Class SCP might perform the following real-world activities: store an image, modify it in some defined manner and act as a Storage Service Class SCU and forward the modified image somewhere.)

AE specifications

MergeCOM-3 Advanced provides support for applications which deal with the following Service Object Pairs (SOPs):

MergeCOM-3 Advanced supported SOP Classes	
SOP Class UID	**SOP Class Name**
1.2.840.10008.5.1.1.15	Basic Annotation Box
1.2.840.10008.5.1.1.4.1	Basic Color Image Box
1.2.840.10008.5.1.1.2	Basic Film Box
1.2.840.10008.5.1.1.1	Basic Film Session
1.2.840.10008.5.1.1.4	Basic Grayscale Image Box
1.2.840.10008.5.1.1.24.1	Basic Print Image Overlay Box SOP Class
1.2.840.10008.5.1.4.37.2	Breast Imaging Relevant Patient Information
1.2.840.10008.5.1.4.37.3	Cardiac Relevant Patient Information Query
1.2.840.10008.5.1.4.1.1.88.65	Chest CAD SR Storage
1.2.840.10008.3.1.2.6.1	Detached Interpretation Management
1.2.840.10008.3.1.2.1.1	Detached Patient Management
1.2.840.10008.3.1.2.5.1	Detached Results Management
1.2.840.10008.3.1.2.3.1	Detached Study Management
1.2.840.10008.3.1.2.2.1	Detached Visit Management
1.2.840.10008.5.1.4.1.1.2.1	Enhanced CT Image Storage
1.2.840.10008.5.1.4.1.1.4.1	Enhanced MR Image Storage
1.2.840.10008.5.1.4.32.3	General Purpose Performed Procedure Step
1.2.840.10008.5.1.4.32.2	General Purpose Scheduled Procedure Step
1.2.840.10008.5.1.4.32.1	General Purpose Woklist Information Model - FIND
1.2.840.10008.5.1.4.37.1	General Relevant Patient Information Query
1.2.840.10008.5.1.4.38.1	Hanging Protocol Storage
1.2.840.10008.5.1.4.38.2	Hanging Protocol Information Model - FIND
1.2.840.10008.5.1.4.38.3	Hanging Protocol Information Model – MOVE
1.2.840.10008.5.1.1.24	Image Overlay Box (Retired)
1.2.840.10008.5.1.4.33	Instance Availability Notification
1.2.840.10008.5.1.4.1.1.88.59	Key Object Selection Document
1.2.840.10008.5.1.4.1.1.88.50	Mammography CAD SR
1.2.840.10008.5.1.1.33	Media Creation Management

MergeCOM-3 Advanced supported SOP Classes (continued)	
SOP Class UID	**SOP Class Name**
1.2.840.10008.1.3.10	Media Storage Directory Storage
1.2.840.10008.5.1.4.31	Modality Worklist Information Model - FIND
1.2.840.10008.5.1.4.1.1.4.2	MR Spectroscopy Storage
1.2.840.10008.5.1.4.1.2.1.1	Patient Root Query/Retrieve Information Model - FIND
1.2.840.10008.5.1.4.1.2.1.3	Patient Root Query/Retrieve Information Model - GET
1.2.840.10008.5.1.4.1.2.1.2	Patient Root Query/Retrieve Information Model - MOVE
1.2.840.10008.5.1.4.1.2.3.1	Patient/Study Only Query/Retrieve Information Model - FIND
1.2.840.10008.5.1.4.1.2.3.3	Patient/Study Only Query/Retrieve Information Model - GET
1.2.840.10008.5.1.4.1.2.3.2	Patient/Study Only Query/Retrieve Information Model - MOVE
1.2.840.10008.3.1.2.3.3	Modality Performed Procedure Step
1.2.840.10008.3.1.2.3.5	Modality Performed Procedure Step Notification
1.2.840.10008.3.1.2.3.4	Modality Performed Procedure Step Retrieve
1.2.840.10008.5.1.4.1.1.77.1.5.2	Ophthalmic 16 bit Photography Image Storage
1.2.840.10008.5.1.4.1.1.77.1.5.1	Ophthalmic 8 bit Photography Image
1.2.840.10008.5.1.1.23	Presentation LUT
1.2.840.10008.5.1.1.16	Printer
1.2.840.10008.5.1.1.16.376	Printer Configuration Retrieval
1.2.840.10008.5.1.1.14	Print Job
1.2.840.10008.5.1.1.26	Print Queue Management
1.2.840.10008.1.40	Procedural Event Logging
1.2.840.10008.5.1.4.1.1.88.40	Procedure Log Storage
1.2.840.10008.5.1.1.31	Pull Print Request
1.2.840.10008.5.1.4.1.1.66	Raw Data Storage
1.2.840.10008.5.1.1.4.2	Referenced Image Box
1.2.840.10008.5.1.4.1.1.1	Computed Radiography Image Storage
1.2.840.10008.5.1.4.1.1.2	CT Image Storage
1.2.840.10008.5.1.4.1.1.9	Standalone Curve Storage
1.2.840.10008.5.1.4.1.1.1.1	Digital X-Ray Image Storage - For Presentation
1.2.840.10008.5.1.4.1.1.1.1.1	Digital X-Ray Image Storage - For Processing
1.2.840.10008.1.1	Verification
1.2.840.10008.5.1.4.1.1.11.1	Grayscale Softcopy Presentation State Storage
1.2.840.10008.5.1.1.30	Hardcopy Color Image Storage
1.2.840.10008.5.1.1.29	Hardcopy Grayscale Image Storage
1.2.840.10008.5.1.4.1.1.1.3	Digital Intra-oral X-Ray Image Storage - For Presentation
1.2.840.10008.5.1.4.1.1.1.3.1	Digital Intra-oral X-Ray Image Storage - For Processing
1.2.840.10008.5.1.4.1.1.1.2	Digital Mammography Image Storage - For Presentation
1.2.840.10008.5.1.4.1.1.1.2.1	Digital Mammography Image Storage - For Processing
1.2.840.10008.5.1.4.1.1.10	Standalone Modality LUT Storage
1.2.840.10008.5.1.4.1.1.4	MR Image Storage
1.2.840.10008.5.1.4.1.1.20	Nuclear Medicine Image Storage
1.2.840.10008.5.1.4.1.1.5	Nuclear Medicine Image Storage (Retired)
1.2.840.10008.5.1.4.1.1.8	Standalone Image Overlay Storage
1.2.840.10008.5.1.4.1.1.128	Positron Emission Tomography Image storage
1.2.840.10008.5.1.4.1.1.129	Standalone PET Curve Storage
1.2.840.10008.5.1.1.27	Stored Print Storage
1.2.840.10008.5.1.4.1.1.481.4	RT Beams Treatment Record Storage
1.2.840.10008.5.1.4.1.1.481.6	RT Brachy Treatment Record Storage
1.2.840.10008.5.1.4.1.1.481.2	RT Dose Storage
1.2.840.10008.5.1.4.1.1.481.1	RT Image Storage
1.2.840.10008.5.1.4.1.1.481.5	RT Plan Storage
1.2.840.10008.5.1.4.1.1.481.3	RT Structure Set Storage
1.2.840.10008.5.1.4.1.1.481.7	RT Treatment Summary Record Storage
1.2.840.10008.5.1.4.1.1.7	Secondary Capture Image storage
1.2.840.10008.5.1.4.1.1.7.2	Multi-frame Grayscale Byte Secondary Capture Image Storage
1.2.840.10008.5.1.4.1.1.7.3	Multi-frame Grayscale Word Secondary Capture Image Storage
1.2.840.10008.5.1.4.1.1.7.1	Multi-frame Single Bit Secondary Capture Image Storage
1.2.840.10008.5.1.4.1.1.7.4	Mulfi-frame True Color Secondary Capture Image Storage
1.2.840.10008.5.1.4.1.1.66.1	Spatial Registration Storage
1.2.840.10008.5.1.4.1.1.66.2	Spatial Fiducials Storage

MergeCOM-3 Advanced supported SOP Classes (continued)	
SOP Class UID	**SOP Class Name**
1.2.840.10008.5.1.4.1.1.77.1.5.3	Stereometric Relationship Storage
1.2.840.10008.5.1.4.1.1.12.1	X-Ray Angiographic Image Storage
1.2.840.10008.5.1.4.1.1.12.3	X-Ray Angiographic Bi-Plane Image Storage
1.2.840.10008.5.1.4.1.1.12.2	X-Ray Radiofluoroscopic Image Storage
1.2.840.10008.5.1.1.22	VOI LUT Box
1.2.840.10008.5.1.4.1.1.88.11	Basic Text Structured Reporting
1.2.840.10008.5.1.4.1.1.88.33	Comprehensive Structured Reporting
1.2.840.10008.5.1.4.1.1.88.22	Enhanced Structured Reporting
1.2.840.10008.5.1.4.1.1.6	Ultrasound Image Storage (Retired)
1.2.840.10008.5.1.4.1.1.6.1	Ultrasound Image Storage
1.2.840.10008.5.1.4.1.1.3	Ultrasound Multi-frame Image Storage (Retired)
1.2.840.10008.5.1.4.1.1.3.1	Ultrasound Multi-frame Image Storage
1.2.840.10008.5.1.4.1.1.77.1.1.1	Video Endoscopic Image Storage
1.2.840.10008.5.1.4.1.1.77.1.2.1	Video Microscopic Image Storage
1.2.840.10008.5.1.4.1.1.77.1.4.1	Video Photographic Image Storage
1.2.840.10008.5.1.4.1.1.77.1.1	VL Endoscopic Image Storage
1.2.840.10008.5.1.4.1.1.77.1.2	VL Microscopic Image Storage
1.2.840.10008.5.1.4.1.1.77.1.4	VL Photographic Image Storage
1.2.840.10008.5.1.4.1.1.77.1.3	VL Slide-Coordinates Microscopic Image Storage
1.2.840.10008.5.1.4.1.1.11	Standalone VOI LUT Storage
1.2.840.10008.1.20.2	Storage Commitment Pull Model
1.2.840.10008.1.20.1	Storage Commitment Push Model
1.2.840.10008.3.1.2.3.2	Study Component Management
1.2.840.10008.1.9	Study Content Notification
1.2.840.10008.5.1.4.1.2.2.1	Study Root Query/Retrieve Information Model - FIND
1.2.840.10008.5.1.4.1.2.2.3	Study Root Query/Retrieve Information Model - GET
1.2.840.10008.5.1.4.1.2.2.2	Study Root Query/Retrieve Information Model - MOVE
1.2.840.10008.5.1.4.1.1.9.1.1	12-lead ECG Waveform Storage
1.2.840.10008.5.1.4.1.1.9.1.3	Ambulatory ECG Waveform Storage
1.2.840.10008.5.1.4.1.1.9.4.1	Basic Voice Audio Waveform Storage
1.2.840.10008.5.1.4.1.1.9.3.1	Cardiac Electrophysiology Waveform Storage
1.2.840.10008.5.1.4.1.1.9.1.2	General ECG Waveform Storage
1.2.840.10008.5.1.4.1.1.9.2.1	Hemodynamic Waveform Storage

Table A-2: Valid SOP Classes for the MergeCOM-3 Advanced Tool Kit

MergeCOM-3 Advanced supported Meta SOP Classes	
SOP Class UID	**SOP Class Name**
1.2.840.10008.5.1.1.18	Basic Color Management
1.2.840.10008.5.1.1.9	Basic Grayscale Print Management
1.2.840.10008.3.1.2.1.4	Detached Patient Management
1.2.840.10008.3.1.2.5.4	Detached Results Management
1.2.840.10008.5.1.4.32	General Purpose Worklist Management
1.2.840.10008.5.1.1.32	Pull Stored Print Management
1.2.840.10008.5.1.1.18.1	Referenced Color Print Management
1.2.840.10008.5.1.1.9.1	Referenced Grayscale Print Management
1.2.840.10008.3.1.2.5.5	Detached Study Management

Table A-3: Valid META-SOP Classes for the MergeCOM-3 Advanced Tool Kit

Application conformance statements specify information about the DICOM SOPs which are supported by the application. For SCP applications, the list of supported services is set within the “MC_Wait_For_Association” or “MC_Wait_For_Secure_Association” function call by specifying a service list which appears in the “mergecom.app” file. Please see the “Application Profile” section in the User’s Manual and the “MC_Wait_For_Association” or “MC_Wait_For_Secure_Association” function

definition in the Reference Manual. For SCU applications, the list of supported SOP classes will correspond to the services specified in “mergecom.app” for any SCPs to which the SCU wishes to connect.

Association establishment policies

General

The maximum PDU size is configurable with a minimum of 4,096 bytes.

Application conformance statements specify the chosen PDU(Protocol Data Units) size and any general rules governing the initiation of associations. Please see the “System Profile” section of the Reference Manual for further information about configuring the PDU size.

Number of associations

Application conformance statements specify the number of simultaneous associations supported by the application. This value is configured in the “mergecom.pro” file as the parameter “MAX_PENDING_CONNECTIONS”. Please see the “System Profile” section of the Reference Manual for further information about configuring this parameter.

Asynchronous nature

MergeCOM-3 Advanced does not currently support multiple outstanding transactions over a single association.

Implementation identifying information

Application conformance statements specify the Implementation Class Unique Identifier (UID) for the application. This UID must follow the syntax rules specified in Part 5 of the DICOM standard. If a version name is supplied, it should also be documented here. Please see the “System Profile” section of the Reference Manual for more information about the implementation identifying information.

Association initiation/acceptance by real-world activity

MergeCOM-3 Advanced applications initiate/accept associations for the services specified in the application service list found in the “mergecom.app” file.

Refer to the *Application Profile* sections in the Reference and User manuals for more information.

Associated real-world activity for application operations

Application conformance statements specify real-world activities for all basic operations associated with the application.

Presentation context table

Application conformance statements specify all presentation contexts which will be used for association negotiation. A presentation context consists of:

- An Abstract Syntax which is a DICOM service class name and unique identifier(UID).
- A transfer syntax name and UID. A transfer syntax represents a set of data encoding rules and is specified in the “mergecom.pro” file. Please see the “System Profile” section in the Reference Manual.

- The role that the application will perform within the service class. The roles associated with a particular service class are discussed in Part 4 of the DICOM standard.
- Any extended negotiation information used when creating associations. See the function “MC_Get_Negotiation_Info” in the Reference manual.

Refer to Table A-3 for an example.

Presentation Context Table					
Abstract Syntax		**Transfer Syntax**		**Role**	**Extended Negotiation**
Name	**UID**	**Name List**	**UID List**		
Computed Radiography Image Storage	1.2.840.10008.5.1.4.1.1.1	DICOM Implicit VR Little Endian	1.2.840.10008.1.2	SCP	None
		DICOM Explicit VR Little Endian	1.2.840.10008.1.2.1		
		DICOM Explicit VR Big Endian	1.2.840.10008.1.2.2		

Table A-3: Example Presentation Context

SOP specific conformance

Application conformance statements list conformance issues for specific SOPs.

Presentation context acceptance criterion

Application conformance statements list special criterion that an application uses when accepting a presentation context. See the function “MC_Get_Association_Info” in the Reference manual to learn about the aspects of presentation contexts which are queryable by an application program.

Transfer syntax selection policies

MergeCOM-3 Advanced uses the following policy when selecting transfer syntax:

- An SCU offers any transfer syntaxes which are defined in it’s “mergecom.pro” file.
- The SCP prefers it’s native byte ordering, and will prefer explicit over implicit VR.

Communication profiles

MergeCOM-3 Advanced runs over the TCP/IP protocol stack on any physical interconnection media supporting the TCP/IP stack.

Extensions/specializations/privatizations

Standard extended/specialized/private SOPs

Application conformance statements list extended, specialized, or private SOPs that are supported.

Private Transfer Syntaxes

MergeCOM-3 does not support private transfer syntaxes.

Configuration

Refer to the “Configuration” sections in both the User and Reference manuals for complete configuration information.

Applications reference four (4) configuration files. The first, merge.ini, is found through the MERGE_INI environment variable. They are as follows:

merge.ini	Specifies the names of the other three (3) configuration files and also contains message logging parameters.
mergecom.pro	Specifies run-time parameters for the application
mergecom.app	Defines service lists and applications on other network nodes to which connections are possible.
mergecom.srv	Service and sequence definitions.

AE title/presentation address mapping

Presentation address mapping is configured in the mergecom.app file. The Presentation Address of an SCU/SCP application is specified by configuring the Listen Port in the mergecom.pro file, and specifying the AE title for the SCU/SCP within the application itself.

Configurable parameters

The mergecom.pro configuration file can be used to set or modify other lower-level communication parameters. This includes time-outs and other parameters. Some information about supported SOP classes is also stored here. **Most parameters in this file should NEVER be changed. Doing so may compromise DICOM conformance.** Before modifying any parameters, such as time-out, be sure to have a backup of the originally supplied mergecom.pro file. Also, before modifying other parameters, you should consider contacting Merge eFilm for advice.

Support of extended character sets

Application conformance statements list supported extended character sets.

Appendix B: Configuration Parameters

This appendix describes each configuration parameter in detail. Information contained in these tables is the parameter names, descriptions and sections where it is contained. The parameters are listed alphabetically and organized by the initialization file where they are used.

The structure of initialization and configuration files are detailed in the section titled *Configuration* in this manual and in the *MergeCOM-3 Advanced Tool Kit: Users Guide*.

Initialization File

The following parameters are recognized by MergeCOM-3 in the initialization file.

Name	Section	Description
BLANK_FILL_LOG_FILE	MergeCOM3	This parameter informs the tool kit whether or not to expand the log file to its maximum size on initialization. Setting this value to "NO" will decrease the time spent in MC_Library_Initialization() but increase the time spent doing actual logging while the application is running. DEFAULT: YES
ERROR_MESSAGE	MergeCOM3	This parameter informs the tool kit to log error messages.
INFO_MESSAGE	MergeCOM3	This parameter informs the tool kit to log error messages.
LOG_FILE	MergeCOM3	This is the name of the MergeCOM-3 message log. The file will be [re-]created by MergeCOM-3. This parameter is ignored by embedded tool kits. DEFAULT: ./merge.log
LOG_FILE_BACKUP	MergeCOM3	This is a boolean parameter that tells MergeCOM-3 to create a backup of the log file before starting a new log. If "ON", any existing log file is renamed with a file extension of .L*nn* where *nn* is an integer number between 01 and 99. DEFAULT: OFF.
LOG_FILE_LINE_LENGTH	MergeCOM3	This option specifies the number of characters that occur on a line within the merge.log file. DEFAULT: 78 MINIMUM: 16 MAXIMUM: 254
LOG_FILE_SIZE	MergeCOM3	This is the number of 80-byte records which will be created for the log file, i.e. the number of 80 character lines in the log file. If BLANK_FILL_LOG_FILE is set to YES, the file is initialized to all binary zeros before the first message is logged. DEFAULT: 1000
LOG_MEMORY_SIZE	MergeCOM3	This is the number of 80-byte records which will be created for the memory log, i.e. the number of 80 character lines in the memory file. Used when logging is set to "Memory" in merge.ini. DEFAULT: 1
MERGECOM_3_APPLICATIONS	MergeCOM3	File containing the MergeCOM-3 application configurations
MERGECOM_3_PROFILE	MergeCOM3	File containing the MergeCOM-3 system profile parameters

MERGECOM_3_SERVICES	MergeCOM3	File containing the MergeCOM-3 system service and message definitions
NUM_HISTORICAL_LOG_FILES	MergeCOM3	This parameter informs the tool kit of the number of historical log files to keep. The valid range of number for this parameter is 1 - 99. The historical log files are named *basename*.L01 to *basename*.L*XX* where *basename*.L*XX* is the latest log file. The *basename* is determined by the LOG_FILE parameter. When the maximum number of historical log files is met, the oldest log file is deleted and the log files are renamed. Note that a new log file is created each time the library is initialized. This parameter is only used when LOG_FILE_BACKUP is set to YES.
DESIRED_LAST_PDU_SIZE	MergeCOM3	This parameter allow the user to configure the length of the last PDU sent. This allows for interoperability with other DICOM implementations that may be intolerant with either a zero or two byte final PDU length. The default value used is 8.
T3_MESSAGE	MergeCOM3	This logging level parameter informs the tool kit to log messages relating to association negotiation.
T4_MESSAGE	MergeCOM3	This logging level parameter informs the tool kit to log messages when incoming associations are automatically rejected.
T5_MESSAGE	MergeCOM3	This logging level parameter informs the tool kit to log messages relating to regular and extended validation.
T6_MESSAGE	MergeCOM3	This logging level parameter informs the tool kit to log messages relating to configuration.
T7_MESSAGE	MergeCOM3	This logging level parameter informs the tool kit to log messages relating to logging of command level attributes in messages sent or received.
T8_MESSAGE	MergeCOM3	This logging level parameter informs the tool kit to log messages relating to the streaming in and out of messages and file objects.
T9_MESSAGE	MergeCOM3	This logging level parameter informs the tool kit to log messages relating to PDU's sent and received. NOTE: Receipt and transmission of P-DATA PDU's are logged; not the actual PDU itself.
WARNING_MESSAGE	MergeCOM3	This parameter informs the tool kit to log warning messages.

Table 1: Initialization file parameters

Application Profile

The application profile is a configuration file that is application dependent. The application profile does not set specific parameters. It sets parameters related to characteristics of your own application entity. A detailed description of the application profile can be found in *MergeCOM-3 Advanced Tool Kit: Users Manual*.

This section will define how each parameter should be defined within the application profile.

Sections

The application profile contains the following sections:

Section	Description
<remote_application_title>	Section describing a remote DICOM Application Entity title(s). The remote Application Entity titles listed here must be 1 to 16 bytes in length with no embedded spaces. Simply, this section is where you list the DICOM applications you want to communicate with.
<service_list_name>	List(s) of DICOM services that will be provided by the Application Entities listed in the [<remote_application_title>] sections. The service names listed here must be 1 to 33 bytes in length with no embedded spaces. Simply, this section is where you list the services that are provided by the remote DICOM applications.
<syntax_list_name>	List(s) of DICOM transfer syntaxes that will be supported by the services listed in the [<service_list_name>] sections. The transfer syntaxes must be one of those listed in Table 5.

Table 2: Application profile section headings

Parameters

The application profile contains the following parameters:

Parameter	Section	Description
PORT_NUMBER	<remote_application_title>	This parameter is the TCP/IP port on which the remote DICOM system listens for connections. The commonly used port number is 104. This default value may be overridden by the `MC_Open_Association` or `MC_Open_Secure_Association` function call.
HOST_NAME	<remote_application_title>	This parameter is the name of the remote host as it is known to your TCP/IP system. This default value may be overridden by the `MC_Open_Association` or `MC_Open_Secure_Association` function call. The parameters value must be 1 to 19 bytes in length with no embedded spaces. NOTE that a numeric internet address may be used: e.g. 192.204.32.1
SERVICE_LIST	<remote_application_title>	This parameter is the name of a section in the application profile which provides a list of services for which local applications will negotiate when attempting to establish an association. This is a default list; another list may be specified in the `MC_Open_Association` or `MC_Open_Secure_Association` function call. The parameters value names must be 1 to 33 bytes in length with no embedded spaces.

Table 3: Application profile section headers.

The `SERVICE_LIST` section of the Application Profile is used to describe the DICOM services that will be negotiated by the listed Application Entity. The parameter values are text strings recognizable by the MergeCOM-3 tool kit. These strings are defined in detail in `message.txt`. This file is located in the `mc3msg` directory of your distribution. The following is a list of currently supported services:

MergeCOM-3 Service Parameter	DICOM Service Class
BASIC_ANNOTATION_BOX	Print Management
BASIC_COLOR_IMAGE_BOX	Print Management
BASIC_FILM_BOX	Print Management
BASIC_FILM_SESSION	Print Management
BASIC_GRAYSCALE_IMAGE_BOX	Print Management
BASIC_PRINT_IMAGE_OVERLAY_BOX	Print Management
BREAST_IMAGING_RPI_QUERY	Relevant Patient Information Query
CARDIAC_RPI_QUERY	Relevant Patient Information Query
DETACHED_INTERP_MANAGEMENT	Results Management
DETACHED_PATIENT_MANAGEMENT	Patient Management
DETACHED_RESULTS_MANAGEMENT	Results Management
DETACHED_STUDY_MANAGEMENT	Study Management
DETACHED_VISIT_MANAGEMENT	Patient Management
DICOMDIR	Media Storage
ENHANCED_CT_IMAGE	Storage
ENHANCED_MR_IMAGE	Storage
G_P_PERFORMED_PROCEDURE_STEP	Study Management
G_P_SCHEDULED_PROCEDURE_STEP	Study Management
G_P_WORKLIST	Basic Worklist Management
GENERAL_RPI_QUERY	Relevant Patient Information Query
HANGING_PROTOCOL	Hanging Protocol Storage
HANGING_PROTOCOL_FIND	Hanging Protocol Query/Retrieve
HANGING_PROTOCOL_MOVE	Hanging Protocol Query/Retrieve
IMAGE_OVERLAY_BOX_RETIRED	Print Management
INSTANCE_AVAIL_NOTIFICATION	Instance Availability Notification
KEY_OBJECT_SELECTION_DOC	Storage
MAMMOGRAPHY_CAD_SR	Storage
MEDIA_CREATION_MANAGEMENT	Media Creation Management
MODALITY_WORKLIST_FIND	Modality Work list
MR_SPECTROSCOPY	Storage
PATIENT_ROOT_QR_FIND	Query/Retrieve
PATIENT_ROOT_QR_GET	Query/Retrieve

PATIENT_ROOT_QR_MOVE	Query/Retrieve
PATIENT_STUDY_ONLY_QR_FIND	Query/Retrieve
PATIENT_STUDY_ONLY_QR_GET	Query/Retrieve
PATIENT_STUDY_ONLY_QR_MOVE	Query/Retrieve
PERFORMED_PROCEDURE_STEP	Study Management
PERFORMED_PROCEDURE_STEP_NOTIFY	Study Management
PERFORMED_PROCEDURE_STEP_RETR	Study Management
PRESENTATION_LUT	Print Management
PRINTER	Print Management
PRINTER_CONFIGURATION	Print Management
PRINT_JOB	Print Management
PRINT_QUEUE_MANAGEMENT	Print Management
PROCEDURAL_EVENT_LOGGING	Application Event Logging
PROCEDURE_LOG	Storage
PULL_PRINT_REQUEST	Print Management
RAW_DATA	Storage
REFERENCED_IMAGE_BOX	Print Management
SC_MULTIFRAME_GRAYSCALE_BYTE	Storage
SC_MULTIFRAME_GRAYSCALE_WORD	Storage
SC_MULTIFRAME_SINGLE_BIT	Storage
SC_MULTIFRAME_TRUE_COLOR	Storage
SPATIAL_FIDUCIALS	Storage
SPATIAL_REGISTRATION	Storage
STANDARD_BASIC_TEXT_SR	Storage
STANDARD_COMPREHENSIVE_SR	Storage
STANDARD_CR	Storage
STANDARD_CT	Storage
STANDARD_CURVE	Storage
STANDARD_DX_PRESENT	Storage
STANDARD_DX_PROCESS	Storage
STANDARD_ECHO	Verification
STANDARD_ENHANCED_SR	Storage
STANDARD_GRAYSCALE_SOFTCOPY_PS	Storage
STANDARD_HARDCOPY_COLOR	Storage
STANDARD_HARDCOPY_GRAYSCALE	Storage
STANDARD_IO_PRESENT	Storage
STANDARD_IO_PROCESS	Storage
STANDARD_MG_PRESENT	Storage
STANDARD_MG_PROCESS	Storage

STANDARD_MODALITY_LUT	Storage
STANDARD_MR	Storage
STANDARD_NM	Storage
STANDARD_OPHTHALMIC_16_BIT	Storage
STANDARD_OPHTHALMIC_8_BIT	Storage
STANDARD_OVERLAY	Storage
STANDARD_PET	Storage
STANDARD_PET_CURVE	Storage
STANDARD_PRINT_STORAGE	Storage
STANDARD_RT_BEAMS_TREAT	Storage
STANDARD_RT_BRACHY_TREAT	Storage
STANDARD_RT_DOSE	Storage
STANDARD_RT_IMAGE	Storage
STANDARD_RT_PLAN	Storage
STANDARD_RT_STRUCTURE_SET	Storage
STANDARD_RT_TREAT_SUM	Storage
STANDARD_SEC_CAPTURE	Storage
STANDARD_US	Storage
STANDARD_US_MF	Storage
STANDARD_US_MF_RETIRED	Storage
STANDARD_US_RETIRED	Storage
STANDARD_VIDEO_ENDOSCOPIC	Storage
STANDARD_VIDEO_MICROSCOPIC	Storage
STANDARD_VIDEO_PHOTOGRAPHIC	Storage
STANDARD_VL_ENDOSCOPIC	Storage
STANDARD_VL_MICROSCOPIC	Storage
STANDARD_VL_PHOTOGRAPHIC	Storage
STANDARD_VL_SLIDE_MICROSCOPIC	Storage
STANDARD_VOI_LUT	Storage
STANDARD_WAVEFORM_12_LEAD_ECG	Storage
STANDARD_WAVEFORM_AMBULATORY_ECG	Storage
STANDARD_WAVEFORM_BASIC_VOICE_AU	Storage
STANDARD_WAVEFORM_CARDIAC_EP	Storage
STANDARD_WAVEFORM_GENERAL_ECG	Storage
STANDARD_WAVEFORM_HEMODYNAMIC	Storage
STANDARD_XRAY_ANGIO	Storage
STANDARD_XRAY_ANGIO_BIPLANE	Storage
STANDARD_XRAY_RF	Storage
STEREOMETRIC_RELATIONSHIP	Storage
STORAGE_COMMITMENT_PULL	Storage Commitment

STORAGE_COMMITMENT_PUSH	Storage Commitment
STUDY_COMPONENT_MANAGEMENT	Study Management
STUDY_CONTENT_NOTIFICATION	Study Content Notification
STUDY_ROOT_QR_FIND	Query/Retrieve
STUDY_ROOT_QR_GET	Query/Retrieve
STUDY_ROOT_QR_MOVE	Query/Retrieve
VOI_LUT_BOX	Print Management
BASIC_COLOR_PRINT_MANAGEMENT (META_SOP)	Print Management
BASIC_GRAYSCALE_PRINT_MANAGEMENT (META_SOP)	Print Management
DETACHED_PATIENT_MANAGEMENT_META (META_SOP)	Patient Management
DETACHED_RESULTS_MANAGEMENT_META (META_SOP)	Results Management
G_P_WORKLIST_MANAGMENT_META (META_SOP)	Basic Worklist Management
PULL_STORED_PRINT_MANAGEMENT (META_SOP)	Print Management
REF_COLOR_PRINT_MANAGEMENT (META_SOP)	Print Management
REF_GRAYSCALE_PRINT_MANAGEMENT (META_SOP)	Print Management
STUDY_MANAGEMENT (META_SOP)	Study Management

Table 4: Application profile parameters.

Transfer syntax lists are contained in the service lists. The following is a list of the currently supported transfer syntaxes:

MergeCOM-3 Transfer Syntax Parameter	Description
IMPLICIT_LITTLE_ENDIAN	Implicit VR Little Endian: Default Transfer Syntax for DICOM
IMPLICIT_BIG_ENDIAN	Implicit VR Big Endian
EXPLICIT_LITTLE_ENDIAN	Explicit VR Little Endian
EXPLICIT_BIG_ENDIAN	Explicit VR Big Endian
DEFLATED_EXPLICIT_LITTLE_ENDIAN	Deflated Explicit VR Little Endian
RLE	Run length Encoding
JPEG_BASELINE	JPEG Baseline (Process 1): Default Transfer Syntax for Lossy JPEG 8 Bit Image Compression
JPEG_EXTENDED_2_4	JPEG Extended (Process 2 & 4): Default Transfer Syntax for Lossy JPEG 12 Bit Image Compression (Process 4 only)
JPEG_EXTENDED_3_5	JPEG Extended (Process 3 & 5)
JPEG_SPEC_NON_HIER_6_8	JPEG Spectral Selection, Non-Hierarchical (Process 6 & 8)
JPEG_SPEC_NON_HIER_7_9	JPEG Spectral Selection, Non-Hierarchical (Process 7 & 9)
JPEG_FULL_PROG_NON_HIER_10_12	JPEG Full Progression, Non-Hierarchical (Process 10 & 12)

JPEG_FULL_PROG_NON_HIER_11_13	JPEG Full Progression, Non-Hierarchical (Process 11 & 13)
JPEG_LOSSLESS_NON_HIER_14	JPEG Lossless, Non-Hierarchical (Process 14)
JPEG_LOSSLESS_NON_HIER_15	JPEG Lossless, Non-Hierarchical (Process 15)
JPEG_EXTENDED_HIER_16_18	JPEG Extended, Hierarchical (Process 16 & 18)
JPEG_EXTENDED_HIER_17_19	JPEG Extended, Hierarchical (Process 17 & 19)
JPEG_SPEC_HIER_20_22	JPEG Spectral Selection, Hierarchical (Process 20 & 22)
JPEG_SPEC_HIER_21_23	JPEG Spectral Selection, Hierarchical (Process 21 & 23)
JPEG_FULL_PROG_HIER_24_26	JPEG Full Progression, Hierarchical (Process 24 & 26)
JPEG_FULL_PROG_HIER_25_27	JPEG Full Progression, Hierarchical (Process 25 & 27)
JPEG_LOSSLESS_HIER_28	JPEG Lossless, Hierarchical (Process 28)
JPEG_LOSSLESS_HIER_29	JPEG Lossless, Hierarchical (Process 29)
JPEG_LOSSLESS_HIER_14	JPEG Lossless, Hierarchical, First-Order Prediction (Process 14 [Selection Value 1]): Default Transfer Syntax for Lossless JPEG Image Compression
JPEG_2000_LOSSLESS_ONLY	JPEG 2000, Lossless
JPEG_2000	JPEG 2000, Lossless or Lossy
JPEG_LS_LOSSLESS	JPEG LS Lossless
JPEG_LS_LOSSY	JPEG LS Lossy (Near-Lossless)
MPEG2_MPML	MPEG2 Main Profile @ Main Level
PRIVATE_SYNTAX_1	Private transfer syntax 1 with the characteristics specified by the PRIVATE_SYNTAX_1_LITTLE_ENDIAN, PRIVATE_SYNTAX_1_EXPLICIT_VR, and PRIVATE_SYNTAX_1_ENCAPSULATED configuration options.
PRIVATE_SYNTAX_2	Private transfer syntax 2 with the characteristics specified by the PRIVATE_SYNTAX_2_LITTLE_ENDIAN, PRIVATE_SYNTAX_2_EXPLICIT_VR, and PRIVATE_SYNTAX_2_ENCAPSULATED configuration options.

Table 5: Transfer Syntax List Parameters

System Profile

The System Profile is used to define system-wide parameters. These parameters apply across all associations with other DICOM application entities. The location of this file is provided by the `MERGECOM_3_PROFILE` parameter of the `[MergeCOM3]` section of the `MERGE.INI` file.

The following are a few notes to keep in mind concerning the System Profile:

- You must specify your own unique DICOM Implementation Class UID and place it in this file along with an optional Implementation Version. These need to be documented in your DICOM conformance statement.
- There are several exception options specified at both the association and DIMSE levels of DICOM communication. You should not have to modify these options in normal circumstances and doing so could make your application non DICOM conformant.
- The DICOM Upper Layer section network time-outs can be modified. This is useful on slower or less-predictable networks (e.g. WAN's).
- The section of the System Profile dealing with transport parameters is important. This is where you specify the **TCP/IP listen port** for a DICOM server (SCP) application, along with the **number of simultaneous associations** your server will support over this port.

A detailed description of the system profile can be found in the *MergeCOM-3 Advanced Tool Kit: Users Manual*.

Table 6 defines how each parameter should be defined within the system profile.

Name	Section	Description
ACCEPT_ANY_APPLICATION_TITLE †	ASSOC_PARMS	If set to YES, the remote system need not specify a correct DICOM application title when requesting an association. If set to NO a correct application title must be used. When this value is set to YES, the tool kit will report the remote application as connecting to the first application registered. DEFAULT: NO
ACCEPT_ANY_CONTEXT_NAME †	ASSOC_PARMS	If set to YES, the remote system need not specify the LOCAL_APPL_CONTEXT_NAME when requesting an association. If set to NO, the correct context name must be used. DEFAULT: NO
ACCEPT_ANY_HOSTNAME	ASSOC_PARMS	If set to YES, the tool kit will not check if applications connecting to an SCP can have their hostname resolved through the SCP's hostfile or domain name server. If set to NO, the tool kit will automatically reject associations from unknown hosts. DEFAULT: NO
ACCEPT_ANY_PRESENTATION_CONTEXT †	ASSOC_PARMS	If set to YES, the tool kit will not validate that the presentation context ID contained in a message's PDU header information matches the ID of the presentation context negotiated for the type of message contained in the PDU. If set to NO, the tool kit will abort associations when these values do not match. DEFAULT: NO
ACCEPT_DIFFERENT_IC_UID †	ASSOC_PARMS	If set to NO, the remote system must specify the local IMPLEMENTATION_CLASS_UID when requesting an association. If set to YES, a different implementation class UID may be used. DEFAULT: YES

ACCEPT_DIFFERENT_VERSION †	ASSOC_PARMS	If set to NO, the remote system must specify the local IMPLEMENTATION_VERSION when requesting an association. If set to YES, a different implementation version may be used. DEFAULT: YES
ACCEPT_MULTIPLE_PRES_CONTEXTS	ASSOC_PARMS	If set to YES, SCP applications will allow multiple presentation contexts to be negotiation for a single DICOM service. If set to NO, an SCP will only accept a single presentation context for a DICOM service. DEFAULT:YES
DEFLATED_EXPLICIT_LITTLE_ENDIAN_SYNTAX	ASSOC_PARMS	This value defined the UID of the Deflated explicit VR little endian transfer syntax. DEFAULT: 1.2.840.10008.1.2.1.99
EXPLICIT_BIG_ENDIAN_SYNTAX	ASSOC_PARMS	This value defines the UID of the explicit VR big endian transfer syntax. DEFAULT: 1.2.840.10008.1.2.2
EXPLICIT_LITTLE_ENDIAN_SYNTAX	ASSOC_PARMS	This value defines the UID of the explicit VR little endian transfer syntax. DEFAULT: 1.2.840.10008.1.2.1
HARD_CLOSE_TCP_IP_CONNECTION	ASSOC_PARMS	This parameter specifies how TCP/IP connections are closed by the tool kit. When set to YES, TCP/IP connections are instantaneously closed with a RST packet. When set to NO, TCP/IP connections are closed gracefully with a FIN packet. Note, that in the NO case the tool kit must wait for an operating system dependent amount of time for the response to the FIN packet. DEFAULT: YES
IMPLEMENTATION_CLASS_UID	ASSOC_PARMS	The DICOM Implementation Class UID (as specified in your DICOM conformance statement).
IMPLEMENTATION_VERSION	ASSOC_PARMS	The Implementation Version Number (as specified in your DICOM conformance statement).
IMPLICIT_BIG_ENDIAN_SYNTAX	ASSOC_PARMS	The implicit big endian transfer syntax is not defined by the DICOM standard. This value is provided to supply compatibility with private implementations. DEFAULT: <none>
IMPLICIT_LITTLE_ENDIAN_SYNTAX	ASSOC_PARMS	The implicit little endian transfer syntax is the default network transfer syntax of the DICOM standard. The implicit little endian transfer syntax must always be defined. DEFAULT: 1.2.840.10008.1.2
LICENSE	ASSOC_PARMS	The MergeCOM-3 Tool Kit license number that was supplied when the tool kit was purchased.
RLE_SYNTAX	ASSOC_PARMS	This value defines the UID of the RLE transfer syntax. DEFAULT: 1.2.840.10008.1.2.5
JPEG_BASELINE_SYNTAX	ASSOC_PARMS	This value defines the UID for JPEG Baseline (Process 1) transfer syntax. DEFAULT: 1.2.840.10008.1.2.4.50
JPEG_EXTENDED_2_4_SYNTAX	ASSOC_PARMS	This value defines the UID for JPEG Extended (Process 2 & 4) transfer syntax. DEFAULT: 1.2.840.10008.1.2.4.51
JPEG_EXTENDED_3_5_SYNTAX	ASSOC_PARMS	This value defines the UID for JPEG Extended (Process 3 & 5) transfer syntax. DEFAULT: 1.2.840.10008.1.2.4.52

JPEG_SPEC_NON_HIER_6_8_SYNTAX	ASSOC_PARMS	This value defines the UID for JPEG Spectral Selection, Non-Hierarchical (Process 6 & 8) transfer syntax. DEFAULT: 1.2.840.10008.1.2.4.53
JPEG_SPEC_NON_HIER_7_9_SYNTAX	ASSOC_PARMS	This value defines the UID for JPEG Spectral Selection, Non-Hierarchical (Process 7 & 9) transfer syntax. DEFAULT: 1.2.840.10008.1.2.4.54
JPEG_FULL_PROG_NON_HIER - _10_12_SYNTAX	ASSOC_PARMS	This value defines the UID for JPEG Full Progression, Non-Hierarchical (Process 10 & 12) transfer syntax. DEFAULT: 1.2.840.10008.1.2.4.55
JPEG_FULL_PROG_NON_HIER - _11_13_SYNTAX	ASSOC_PARMS	This value defines the UID for JPEG Full Progression, Non-Hierarchical (Process 11 & 13) transfer syntax. DEFAULT: 1.2.840.10008.1.2.4.56
JPEG_LOSSLESS_NON_HIER - _14_SYNTAX	ASSOC_PARMS	This value defines the UID for JPEG Lossless, Non-Hierarchical (Process 14) transfer syntax. DEFAULT: 1.2.840.10008.1.2.4.57
JPEG_LOSSLESS_NON_HIER - _15_SYNTAX	ASSOC_PARMS	This value defines the UID for JPEG Lossless, Non-Hierarchical (Process 15) transfer syntax. DEFAULT: 1.2.840.10008.1.2.4.58
JPEG_EXTENDED_HIER - _16_18_SYNTAX	ASSOC_PARMS	This value defines the UID for JPEG Extended, Hierarchical (Process 16 & 18) transfer syntax. DEFAULT: 1.2.840.10008.1.2.4.59
JPEG_EXTENDED_HIER - _17_19_SYNTAX	ASSOC_PARMS	This value defines the UID for JPEG Extended, Hierarchical (Process 17 & 19) transfer syntax. DEFAULT: 1.2.840.10008.1.2.4.60
JPEG_SPEC_HIER - _20_22_SYNTAX	ASSOC_PARMS	This value defines the UID for JPEG Spectral Selection, Hierarchical (Process 20 & 22) transfer syntax. DEFAULT: 1.2.840.10008.1.2.4.61
JPEG_SPEC_HIER - _21_23_SYNTAX	ASSOC_PARMS	This value defines the UID for JPEG Spectral Selection, Hierarchical (Process 21 & 23)transfer syntax. DEFAULT: 1.2.840.10008.1.2.4.62
JPEG_FULL_PROG_HIER - _24_26_SYNTAX	ASSOC_PARMS	This value defines the UID for JPEG Full Progression, Hierarchical (Process 24 & 26) transfer syntax. DEFAULT: 1.2.840.10008.1.2.4.63
JPEG_FULL_PROG_HIER - _25_27_SYNTAX	ASSOC_PARMS	This value defines the UID for JPEG Full Progression, Hierarchical (Process 25 & 27) transfer syntax. DEFAULT: 1.2.840.10008.1.2.4.64
JPEG_LOSSLESS_HIER - _28_SYNTAX	ASSOC_PARMS	This value defines the UID for JPEG Lossless, Hierarchical (Process 28) transfer syntax. DEFAULT: 1.2.840.10008.1.2.4.65
JPEG_LOSSLESS_HIER - _29_SYNTAX	ASSOC_PARMS	This value defines the UID for JPEG Lossless, Hierarchical (Process 29) transfer syntax. DEFAULT: 1.2.840.10008.1.2.4.66
JPEG_LOSSLESS_HIER_14_SYNTAX	ASSOC_PARMS	This value defines the UID for JPEG Lossless, Hierarchical, First-Order Prediction (Process 14, Selection Value 1) transfer syntax. DEFAULT: 1.2.840.10008.1.2.4.70
JPEG_2000_LOSSLESS_ONLY_SYNTAX	ASSOC_PARMS	This value defines the UID for JPEG 2000, Lossless. DEFAULT: 1.2.840.10008.1.2.4.90

JPEG_2000_SYNTAX	ASSOC_PARMS	This value defines the UID for JPEG 2000. DEFAULT: 1.2.840.10008.1.2.4.91
JPEG_LS_LOSSLESS_SYNTAX	ASSOC_PARMS	This value defines the UID for JPEG LS Lossless. DEFAULT: 1.2.840.10008.1.2.4.80
JPEG_LS_LOSSY_SYNTAX	ASSOC_PARMS	This value defines the UID for JPEG LS Lossy (Near Lossless). DEFAULT: 1.2.840.10008.1.2.4.81
MPEG2_MPML_SYNTAX	ASSOC_PARMS	This value defines the UID for MPEG2 Main Profile @ Main Level. DEFAULT: 1.2.840.10008.1.2.4.100
LOCAL_APPL_CONTEXT_NAME	ASSOC_PARMS	The DICOM Application Context Name (UID) (as specified in the DICOM Standard). DEFAULT: "1.2.840.10008.3.1.1.1"
PDU_MAXIMUM_LENGTH *	ASSOC_PARMS	The maximum size of DICOM Protocol Data Units that can be received by this MergeCOM-3 implementation. This value is used during association negotiation to specify the maximum sized PDUs that your application can receive. DEFAULT: 28K MINIMUM: 4K MAXIMUM: NONE
PRIVATE_SYNTAX_1_SYNTAX	ASSOC_PARMS	The unique identifier (UID) MergeCOM-3 will use to identify private transfer syntax 1. When this value is set to "<none>", private transfer syntax support is shut off. DEFAULT: <none>
PRIVATE_SYNTAX_1_LITTLE_ENDIAN	ASSOC_PARMS	When set to YES, MergeCOM-3 will interpret private transfer syntax 1 as being encoded in little endian format. DEFAULT: YES
PRIVATE_SYNTAX_1_EXPLICIT_VR	ASSOC_PARMS	When set to YES, MergeCOM-3 will interpret private transfer syntax 1 as being encoded in explicit VR format. DEFAULT: YES
PRIVATE_SYNTAX_1_ENCAPSULATED	ASSOC_PARMS	When set to YES, MergeCOM-3 will interpret private transfer syntax 1 as having its pixel data tag (7fe0,0010) being encoded as undefined length in the same manner as the JPEG and RLE transfer syntaxes are encoded. DEFAULT: NO
PRIVATE_SYNTAX_2_SYNTAX	ASSOC_PARMS	The unique identifier (UID) MergeCOM-3 will use to identify private transfer syntax 2. When this value is set to "<none>", private transfer syntax support is shut off. DEFAULT: <none>
PRIVATE_SYNTAX_2_LITTLE_ENDIAN	ASSOC_PARMS	When set to YES, MergeCOM-3 will interpret private transfer syntax 2 as being encoded in little endian format. DEFAULT: YES
PRIVATE_SYNTAX_2_EXPLICIT_VR	ASSOC_PARMS	When set to YES, MergeCOM-3 will interpret private transfer syntax 2 as being encoded in explicit VR format. DEFAULT: YES
PRIVATE_SYNTAX_2_ENCAPSULATED	ASSOC_PARMS	When set to YES, MergeCOM-3 will interpret private transfer syntax 2 as having its pixel data tag (7fe0,0010) being encoded as undefined length in the same manner as the JPEG and RLE transfer syntaxes are encoded. DEFAULT: NO

INITIATOR_NAME †	DIMSE_PARMS	The DICOM standard has retired the old ACR/NEMA Initiator Name attribute in command messages. To generated such an attribute in command messages, specify an initiator name. <none> means do not put initiator name in messages. DEFAULT: <none>
RECEIVER_NAME †	DIMSE_PARMS	The DICOM standard has retired the old ACR/NEMA Receiver Name attribute in command messages. To generated such an attribute in command messages, specify a receiver name. <none> means do not put receiver name in messages. DEFAULT: <none>
SEND_ECHO_PRIORITY †	DIMSE_PARMS	The DICOM standard has retired the message priority attribute in echo command messages. To generated such an attribute in command messages, specify YES. To NOT use message priority in echo messages, specify NO. DEFAULT: NO
SEND_LENGTH_TO_END †	DIMSE_PARMS	The DICOM standard has retired the old Group-Length-To-End attribute in command messages. To generate such an attribute in command messages, specify YES. If you do not want to generate Group-Length-To-End, specify NO. DEFAULT: NO
SEND_MSG_ID_RESPONSE †	DIMSE_PARMS	The DICOM standard has retired the message ID attribute in response command messages. To generated such an attribute in command messages, specify YES. To NOT use message ID in response messages, specify NO. DEFAULT: NO
SEND_RECOGNITION_CODE †	DIMSE_PARMS	The DICOM standard has retired the old Recognition Code attribute in command messages. To generated such an attribute in command messages, specify YES. If you do not want to generate such an attribute, specify NO, ACR-NEMA 2.0. DEFAULT: NO
SEND_RESPONSE_PRIORITY †	DIMSE_PARMS	The DICOM standard has retired the message priority attribute in response messages. To generated such an attribute in response messages, specify YES. To NOT use message priority in response messages, specify NO. DEFAULT: NO
SEND_SOP_CLASS_UID †	DIMSE_PARMS	Certain DICOM service classes demand that the affected SOP class UID be present in the message. To prevent the library from insuring that this is done, specify NO. To insure that Affected SOP class UID is present, specify YES. DEFAULT: YES
SEND_SOP_INSTANCE_UID †	DIMSE_PARMS	Certain DICOM service classes demand that the affected SOP instance UID be present in the message. To prevent the library from insuring that this is done, specify NO. To insure that Affected SOP instance UID is present, specify YES. DEFAULT: YES
ARTIM_TIMEOUT	DUL_PARMS	The number of seconds to use as a time out waiting for an association request or waiting for the peer to shut down an association. DEFAULT: 30
ASSOC_REPLY_TIMEOUT	DUL_PARMS	The number of seconds to wait for a reply to an associate request. DEFAULT: 15.

CONNECT_TIMEOUT	DUL_PARMS	The number of seconds to wait for a network connect to be accepted. DEFAULT: 15.
INACTIVITY_TIMEOUT	DUL_PARMS	The number of seconds to wait in between packets of data received over the network after the initial packet of data in a message is received. Used by the `MC_Read_Message()` function. DEFAULT: 15.
INSURE_EVEN_UID_LENGTH †	DUL_PARMS	Set to NO, if odd-length UIDs in PDU's should NOT be padded with a NULL to insure even length unique Ids. Set to YES to insure even UIDs in PDUs. DEFAULT: NO
RELEASE_TIMEOUT	DUL_PARMS	The number of seconds to wait for a reply to an associate release. DEFAULT: 15.
WRITE_TIMEOUT	DUL_PARMS	The number of seconds to wait for a network write to be accepted. DEFAULT: 15.
EXPORT_PRIVATE_ATTRIBUTES_ - TO_MEDIA	MEDIA_PARMS	When set to NO, disable the exporting of private attributes in files written with the `MC_Write_File()` and `MC_Write_File_By_Callback()` functions. DEFAULT: YES
EXPORT_UN_VR_TO_MEDIA	MEDIA_PARMS	When set to NO, disable the exporting of attributes with a VR of UN in files written with the `MC_Write_File()` and `MC_Write_File_By_Callback()` functions. DEFAULT: YES
EXPORT_GROUP_LENGTHS_TO_MEDIA *	MEDIA_PARMS	When set to NO, do not write group length attributes with MC_Write_File() and MC_Write_File_By_Callback() DEFAULT: YES
EXPORT_UNDEFINED_LENGTH_SQ_ - IN_DICOMDIR *	MEDIA_PARMS	When set to NO, DICOMDIRs written with MC_Write_File() are created with their sequence attributes having defined lengths. Setting this option to Yes will increase performance. DEFAULT: YES
ALLOW_COMMA_IN_DS_FL_FD_STRINGS	MESSAGE_PARMS	When set to Yes, a comma or a period will be allowed in the value passed to MC_Set_Value_From_String for attributes with a VR of DS, FL or FD. When set to No, only a period will be acceptable as the radix character. Note that the toolkit will always ensure that DS attributes use a period radix character when streaming to the network or to a file, regardless of current locale settings. DEFAULT: NO
ALLOW_INVALID_PRIVATE_ - ATTRIBUTES	MESSAGE_PARMS	When reading messages or file objects, this parameter specifies if private attributes encoded in an invalid format should be ignored or parsed. DEFAULT: NO
ALLOW_INVALID_PRIVATE_ - CREATOR_CODES	MESSAGE_PARMS	When reading messages or file objects, this parameter specifies if private creator codes encoded with invalid characters should be ignored or parsed. DEFAULT: NO
CALLBACK_MIN_DATA_SIZE	MESSAGE_PARMS	When using the `MC_Register_Callback_Function()` call to store large data such as pixel data, this option specifies the minimum size of value for which the callback function should be used. This option was specifically added so pixel data contained in icons are not managed with a callback function. DEFAULT: 1

COMPRESSION_LUM_FACTOR	MESSAGE_PARMS	Configuration Parameter for MC_Standard_Compressor. Value 0 through 255. 0 is the highest quality, giving a quantization table of all 1's. 32 corresponds to the standard quantization tables. For values between 0 and 128, the standard tables are scaled linearly. For values between 128 and 255, the standard tables are scaled non-linearly and the compression increases (and the quality decreases) by a very large amount. DEFAULT: 32
COMPRESSION_CHROM_FACTOR	MESSAGE_PARMS	Configuration Parameter for MC_Standard_Compressor.Value 0 through 255.The chrominance compression factor is used to adjust the default chrominance quantization table values. When ChromFactor is 32, the default chrominance quantization table values are used as is. A value of 255 corresponds to high compression, low quality. DEFAULT: 32
COMPRESSION_ALLOW_FRAGS	MESSAGE_PARMS	Configuration Parameter for MC_Standard_Compressor. The Pegasus libraries allow compressed image data to be returned as it continues to compress more image data. This may result in an image frame having one or more fragments. This is perfectly legal, however some viewers may not be able to display the image if they do not support multiple fragments per frame.
COMPRESSION_WHEN_J2K_USE_LOSSY	MESSAGE_PARMS	Configuration Parameter for MC_Standard_Compressor. When JPEG_2000 is used as a transfer syntax, this could mean either lossy or lossless compression. This parameter specifies the intended syntax.
COMPRESSION_J2K_LOSSY_RATIO	MESSAGE_PARMS	Configuration Parameter for MC_Standard_Compressor. When JPEG_2000 with COMPRESSION_WHEN_J2K_USE_LOSSY = Yes, and COMPRESSION_J2K_LOSSY_USE_QUALITY = No, a ratio can be specified. The compressor attempts to reduce the image size to 1/COMPRESSION_J2K_LOSSY_RATIO.
COMPRESSION_J2K_LOSSY_QUALITY	MESSAGE_PARMS	Configuration Parameter for MC_Standard_Compressor. When JPEG_2000 with COMPRESSION_WHEN_J2K_USE_LOSSY = Yes, and COMPRESSION_J2K_LOSSY_USE_QUALITY = Yes, a quality can be specified. Valid values are 1 to 10, 1 being highest quality image.
COMPRESSION_J2K_LOSSY_USE_ - QUALITY	MESSAGE_PARMS	Configuration Parameter for MC_Standard_Compressor. When JPEG_2000 with COMPRESSION_WHEN_J2K_USE_LOSSY = Yes, this indicates which metric should be used for lossy compression, ratio or quality.
COMPRESSION_RGB_TRANSFORM_ - FORMAT	MESSAGE_PARMS	This parameter allows the user to select the output format when doing Lossy JPEG compression of RGB images. The value can be set to YBR_FULL or YBR_FULL_422 to specifiy what photometric interpretion MergeCOM-3 should compress into when compressing RGB images. DEFAULT: YBR_FULL_422
DEFLATE_COMPRESSION_LEVEL	MESSAGE_PARMS	Allows the compression level of deflate to be specified when using deflated explicit VR little endian transfer syntax. 0 is no compression, 1 is fastest, and 9 compresses best. DEFAULT: -1
DEFLATE_ALLOW_FLUSH	MESSAGE_PARMS	Allows deflate to flush data occasionally to limit buffering. DEFAULT: Yes

DICTIONARY_ACCESS	MESSAGE_PARMS	This parameter specifies whether or not the DICOM dictionary is to be loaded into memory, or accessed from the dictionary file. FILE means access information directly from the dictionary file. MEM means load the dictionary into memory and access it there. NOTE: This parameter is ignored if the dictionary is pre-compiled and loaded as part of the program (using the gendict utility) NOTE: Embedded systems must always use pre-compiled dictionary entries. DEFAULT: MEM
DICTIONARY_FILE	MESSAGE_PARMS	This parameter specifies the name (path) of the DICOM dictionary. An absolute or relative path may be specified. NOTE: This parameter is ignored if the dictionary has been pre-compiled. DEFAULT: ../mc3msg/mrgcom3.dct
DUPLICATE_ENCAPSULATED_ICON	MESSAGE_PARMS	When duplicating to an encapsulated transfer syntax, this configuration value specifies whether an ICON IMAGE SEQUENCE should also be encapsulated.
ELIMINATE_ITEM_REFERENCES *	MESSAGE_PARMS	This parameter specifies the behavior of the message/item/file handling functions `MC_Free_Message()`, `MC_Empty_Message()`, `MC_Free_Item()`, `MC_Empty_Item()`, `MC_Free_File()` and `MC_Empty_File()`. If this parameter is set to YES, the above functions will search for references in every currently open object to delete when they encounter an item to free within an object. DEFAULT: YES.
EMPTY_PRIVATE_CREATOR_CODES	MESSAGE_PARMS	If set to NO, private creator codes contained in messages are not emptied when the `MC_Empty_Message()` or `MC_Empty_File()` function calls are made. DEFAULT: YES
EXPORT_PRIVATE_ATTRIBUTES_ - TO_NETWORK	MESSAGE_PARMS	When set to NO, disable the exporting of private attributes in messages written to the network with the `MC_Send_Request_Message()` and `MC_Send_Response_Message()` functions. DEFAULT: YES
EXPORT_UN_VR_TO_NETWORK	MESSAGE_PARMS	When set to NO, disable the exporting of attributes with a VR of UN in messages written to the network with the `MC_Send_Request_Message()` and `MC_Send_Response_Message()` functions. DEFAULT: YES
EXPORT_UNDEFINED_LENGTH_SQ *	MESSAGE_PARMS	If YES, messages transferred over the network or written to disk have their sequence attributes encoded as undefined length. This increases performance of the library. DEFAULT: NO
EXPORT_GROUP_LENGTHS_TO_NETWORK *	MESSAGE_PARMS	When set to NO, do not export group length attributes when using the MC_Send_Request_Message() and MC_Send_Response_Message() functions DEFAULT: YES
FLATE_GROW_OUTPUT_BUF_SIZE *	MESSAGE_PARMS	The size that the output buffer of deflate or inflate should grow when it's size is insufficient. An Info message is logged each time the buffer grows. DEFAULT: 1024

FORCE_OPEN_EMPTY_ITEM *	MESSAGE_PARMS	When set to YES, the `MC_Open_Item()` function will act similar to the `MC_Open_Empty_Message()` function. The up-front performance cost of the `MC_Open_Item()` function will be reduced, but the amount of validation done when adding tags to the item is reduced. Setting this value to YES will also improve the performance of the DICOMDIR directory functions. This configuration value does not have any effect on embedded platforms. DEFAULT: NO
IGNORE_JPEG_BAD_SUFFIX	MESSAGE_PARMS	Config Parameter for MC_Standard_Decompressor to deal with lossless JPEG images whose suffix have been invalidly written according to the JPEG specification. These images have a 16-zero-bit suffix following a -32768 prefix where the JPEG spec says the suffix is omitted following a -32768 prefix. The following are the valid settings: -1 = Default, fail on these images 0 = Ignore when user detects such images 1 = Let the toolkit detect and ignore automatically
LARGE_DATA_SIZE	MESSAGE_PARMS	Defines “Large Data” to the tool kit. "Large Data" is defined as an attribute value which has a length of LARGE_DATA_SIZE or more. DEFAULT: 200.
LARGE_DATA_STORE	MESSAGE_PARMS	This parameter specifies where "Large Data" values should be stored. FILE means store the values in temporary files. MEM means store the values in memory. NOTE: Embedded systems should ignore this parameter and always use MEM. DEFAULT: MEM
MSG_FILE_ITEM_OBJ_TRACE	MESSAGE_PARMS	This parameter allows the tracking of the creation, referencing, and freeing of message, file, and item objects. This option can be used if the user suspects a memory leak in their application from not freeing one of these object types. The logging is done at the T1 trace level which must be enabled in the merge.ini file. DEFAULT: NO
MSG_INFO_FILE	MESSAGE_PARMS	This parameter specifies the name (path) of the DICOM message information file. An absolute or relative path may be specified. DEFAULT: ../mc3msg/mrgcom3.msg
NULL_TYPE3_VALIDATION	MESSAGE_PARMS	This parameter specifies how the toolkit will validate a single NULL value in a type 3 attribute with VM>1. DEFAULT: ERR
OBOW_BUFFER_SIZE	MESSAGE_PARMS	This parameter specifies the number of bytes of “Large Data” that should be buffered before it is written to disk. This value is only used when the parameter LARGE_DATA_STORE is set to FILE. DEFAULT: 4096
PEGASUS_OP_*_NAME	MESSAGE_PARMS	When using your own Pegasus license to remove the 3 frames/second limit, this should have the company name that was used to generate your Pegasus license.
PEGASUS_OP_*_REGISTRATION	MESSAGE_PARMS	When using your own Pegasus license to remove the 3 frames/second limit, this should have the registration code that goes with it's respective PEGASUS_OP_*_NAME.
REMOVE_PADDING_CHARS	MESSAGE_PARMS	When set to Yes, MergeCOM-3 will remove space padding characters from all text based attributes. This removal will occur when the attribute is encoded with one of the MC_Set_Value functions, or when the attribute is read with one of the streaming or network read functions. DEFAULT: No

RETURN_COMMA_IN_DS_FL_FD_ - STRINGS	MESSAGE_PARMS	When set to Yes, MergeCOM-3 will return a comma character as the radix character in a value when MC_Get_Value_To_String is called for an attribute with a VR of DS, FL, or FD. When set to No, a period will always be returned for the radix character. Note that DS values will always be properly encoded with a period in DICOM message objects. DEFAULT: No
TEMP_FILE_DIRECTORY	MESSAGE_PARMS	This parameter specifies the directory in which temporary files should be created. This parameter is used only if LARGE_DATA_STORE = FILE. An absolute or relative path may be specified. DEFAULT: ./
UN_VR_CODE	MESSAGE_PARMS	VR Code to use for attributes with unknown VRs. This may be set to 'OB' if an implementation does not understand 'UN'. DEFAULT: UN VALID VALUES: UN, OB
WORK_BUFFER_SIZE *	MESSAGE_PARMS	This parameter specifies the amount of data that is buffered in the tool kit before being stored internally or passed to a user's callback function. This option impacts the `MC_Message_To_Stream()`, `MC_Stream_To_Message()`, `MC_Send_Request_Message()`, `MC_Send_Response_Message()`, `MC_Read_Message()`, `MC_Open_File()`, `MC_Open_File_Bypass_OBOW()`, `MC_Open_File_Upto_Tag()`, `MC_Write_File()`, and `MC_Write_File_By_Callback()` functions. Setting this option to values larger than 28K will in most cases cause the tool kit to use the operating system's memory management scheme instead of the tool kit's internal mechanism. DEFAULT: 28K
CAPTURE_FILE	TRANSPORT_PARMS	This parameter specifies the base name to use for capture files. (Capture files are generated if the NETWORK_CAPTURE value is set to Yes.) If only one capture file is requested (see NUMBER_OF_CAP_FILES), the capture file will have the name specified. If more than one is requested, nnn will be appended to the base file name specified (e.g. merge001.cap) DEFAULT: merge.cap (in the current directory)
CAPTURE_FILE_SIZE	TRANSPORT_PARMS	This parameter specifies the maximum size (in kilobytes) that capture files are allowed to grow. (Capture files are generated if the NETWORK_CAPTURE value is set to Yes.) If more than one capture file is requested (see NUMBER_OF_CAP_FILES), each file generated will have this maximum size. If a value less than 1 is specified only one capture file of unlimited length will be generated. DEFAULT: 0
COMBINE_DATA_WITH_HEADER	TRANSPORT_PARMS	This parameter specifies if you would like the 12 byte PDU header combined with the PDU data. This is only for data given to the network layer that is <= 256 bytes.
MAX_PENDING_CONNECTIONS	TRANSPORT_PARMS	This parameter specifies the maximum number of open listen channels. It's value is used as the second argument of a TCP listen() call. DEFAULT: 5

NETWORK_CAPTURE	TRANSPORT_PARMS	This parameter specifies whether or not network data should be captured in files suitable to be read by the MergeDPM utility. Use these parameters to customize the network capture: CAPTURE_FILE CAPTURE_FILE_SIZE NUMBER_OF_CAP_FILES REWRITE_CAPTURE_FILES DEFAULT: No
NUMBER_OF_CAP_FILES	TRANSPORT_PARMS	This parameter specifies the number of capture files to generate. (Capture files are generated if the NETWORK_CAPTURE value is set to Yes.) Each capture file generated will have maximum size specified by CAPTURE_FILE_SIZE. If CAPTURE_FILE_SIZE is less than 1 (unlimited size) this parameter's value is ignored. DEFAULT: 1
REWRITE_CAPTURE_FILES	TRANSPORT_PARMS	This parameter specifies the whether or not capture files should be rewritten when all files have reached the maximum size specified by CAPTURE_FILE_SIZE. (Capture files are generated if the NETWORK_CAPTURE value is set to Yes.) If Yes is specified, the oldest written file will be rewritten. If No is specified and all requested files have been written (see NUMBER_OF_CAP_FILES), no more data will be captured. DEFAULT: Yes
TCPIP_DISABLE_NAGLE	TRANSPORT_PARMS	This parameter specifies if the Nagle Algorithm should be used when sending packets at the TCP/IP level. Most operating systems enable this by default. It allows small segments of data to delay sending a fixed amount of time to possibly be combined with other small segments and be sent as one larger packet. Disabling this may cause high network traffic.
TCPIP_LISTEN_PORT	TRANSPORT_PARMS	This parameter specifies the TCP/IP port on which server applications are to listen for associate requests. DEFAULT: 104
TCPIP_RECEIVE_BUFFER_SIZE *	TRANSPORT_PARMS	This parameter specifies the TCP/IP receive buffer size for each connection. For optimum performance, this value should be set slightly higher than the value for PDU_MAXIMUM_LENGTH. DEFAULT: 29K MAXIMUM: Operating System dependent
TCPIP_SEND_BUFFER_SIZE *	TRANSPORT_PARMS	This parameter specifies the TCP/IP send buffer size for each connection. For optimum performance, this value should be set slightly higher than the value for PDU_MAXIMUM_LENGTH. DEFAULT: 29K MAXIMUM: Operating System dependent

Table 6: System profile parameters.

† **These options allow for non-standard DICOM operations**. Such exceptions, if used, should be noted in your DICOM conformance statement.

* ***Performance Tuning***

Service Profile

The Service Profile is generated by Merge eFilm and contains DICOM standard services and commands and is a useful reference (along with the `message.txt` file mentioned previously) to find the MergeCOM-3 names for the standard DICOM services and items. It is used by the library to negotiate the proper SOP Class UIDs and to access the binary dictionary and message information files when creating instances of message objects and validating messages.

In most cases, it will not be necessary to modify the Service Profile. However, if you are using an extended tool kit to create your own private services, you will need to add specifications for these private services to the Service Profile. See the *MergeCOM-3 Advanced Tool Kit: Extended Tool Kit Manual* for further details.

The location of the Service Profile is provided by the `MERGECOM_3_SERVICES` parameter of the `[MergeCOM3]` section of the `MERGE.INI` file.

Remember, the Service Profile is GENERATED by the MergeCOM-3 Profile Database Utilities at Merge eFilm. Unless you are absolutely confident about changes being made, DO NOT CHANGE THE CONTENTS OF THIS FILE.

The Service Profile contains the following sections:

Name	Description
[SERVICE_TABLE]	List of service names and numbers. This list registers every service available to an Application Entity. The parameters associated with [SERVICE_LIST] are NUMBER_OF_SERVICES_SUPPORTED (the number of service names that will be listed immediately following NUMBER_OF_SERVICES_SUPPORTED) and one entry for each supported service.
[<service_number>]	One section number for each of the above services registered in [SERVICE_TABLE]. Each section contains a Service Name, a DICOM SOP Class UID for the Service, a flag that tells whether it is a BASE or META Service (SOP) and a list of commands supported for that service.
[ITEM_TABLE]	One item name and number for each DICOM item that can be encoded in an attribute of Value representation SQ (Sequence of Items).

Table 6: Service profile parameters.

Index