Symbols Glossary:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>REF</td>
<td>Part or Catalog number</td>
</tr>
<tr>
<td></td>
<td>Manufacturer</td>
</tr>
</tbody>
</table>

The full symbols glossary can be viewed at https://www.merative.com/content/dam/merative/documents/brief/Merge_Healthcare_Symbols_Glossary.pdf.

INSTRUCTIONS FOR USE:

Instructions for use are available electronically in PDF format. Paper format can be requested for supported languages by contacting Customer Support and will be provided free of charge within seven calendar days of receiving the request. The paper format can also be provided at the time of delivery if so requested at the time of order.

For application support or to report issues with user documentation, contact Customer Support:

📞 1-877-741-5369 (North America)
+31.20.514.5073 (Europe, the Middle East and Africa)
1800 316 746 (Australia)

✉️ MC3Support@merative.com

<table>
<thead>
<tr>
<th>Part</th>
<th>Date</th>
<th>Revision</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COM-5278</td>
<td>January 2023</td>
<td>1.0</td>
<td>Updated bi-annually</td>
</tr>
</tbody>
</table>

The latest version of this document can be found at https://mergecustomer.force.com/mergeusercommunity/login.
MC_Dir_Delete_Referreded_Entity .............................................................. 60
MC_Dir_Entity_Count............................................................................... 61
MC_Dir_First_Record............................................................................. 62
MC_Dir_Item_Count................................................................................ 64
MC_Dir_Next_Entity............................................................................... 65
MC_Dir_Next_Record.............................................................................. 66
MC_Dir_Open_MRDR............................................................................... 67
MC_Dir_Reference_MRDR ....................................................................... 68
MC_Dir_Remove_Ref_MRDR ..................................................................... 69
MC_Dir_Root_Count................................................................................ 70
MC_Dir_Root_Entity............................................................................... 71
MC_Dir_Sort............................................................................................ 72
MC_Dir_Stream_Directory_Record................................................................. 75
MC_Duplicate_Message.......................................................................... 76
MC_Empty_File......................................................................................... 80
MC_Empty_Item......................................................................................... 81
MC_Empty_Message................................................................................. 81
MC_Enable_Unicode_Conversion................................................................. 82
MC_Error_Message................................................................................ 83
MC_File_To_Message............................................................................... 84
MC_Free_Item......................................................................................... 84
MC_Free_File.......................................................................................... 85
MC_Free_Message................................................................................... 86
MC_FreeService..................................................................................... 87
MC_FreeServiceList................................................................................. 88
MC_FreeSyntaxList................................................................................ 89
MC_Get_Association_Info....................................................................... 89
MC_Get_Attribute_Info.......................................................................... 91
MC_Get_Bool_Config_Value...................................................................... 92
MC_Get_Encapsulated_Value_To_Function.................................................. 94
MC_Get_Enum_From_Transfer_Syntax....................................................... 97
MC_Get_File_Length................................................................................ 99
MC_Get_File_Preamble.......................................................................... 100
MC_Get_Filename.................................................................................... 101
MC_Get_First_Acceptable_Service.............................................................. 101
MC_Get_First.Attribute........................................................................ 103
MC_Get_Frame_To_Function................................................................... 104
MC_Get_Int_Config_Value....................................................................... 106
<table>
<thead>
<tr>
<th>Function Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_Get_Listen_Socket</td>
<td>108</td>
</tr>
<tr>
<td>MC_Get_Listen_Socket_For_Port</td>
<td>109</td>
</tr>
<tr>
<td>MC_Get_Log_Destination</td>
<td>110</td>
</tr>
<tr>
<td>MC_Get_Long_Config_Value</td>
<td>111</td>
</tr>
<tr>
<td>MC_Get_MergeCOM_Service</td>
<td>112</td>
</tr>
<tr>
<td>MC_Get_MergeCOM_Service_From_UID</td>
<td>113</td>
</tr>
<tr>
<td>MC_Get_Message_Service</td>
<td>113</td>
</tr>
<tr>
<td>MC_Get_Message_Transfer_Syntax</td>
<td>115</td>
</tr>
<tr>
<td>MC_Get_Meta_ServiceName</td>
<td>116</td>
</tr>
<tr>
<td>MC_Get_Negotiation_Info</td>
<td>117</td>
</tr>
<tr>
<td>MC_Get_Next_Acceptable_Service</td>
<td>119</td>
</tr>
<tr>
<td>MC_Get_Next_Attribute</td>
<td>120</td>
</tr>
<tr>
<td>MC_Get_Next_Encapsulated_Value_To_Function</td>
<td>121</td>
</tr>
<tr>
<td>MC_Get_Next_pValue... Functions</td>
<td>124</td>
</tr>
<tr>
<td>MC_Get_Next.Validate_Error</td>
<td>128</td>
</tr>
<tr>
<td>MC_Get_Next_Value... Functions</td>
<td>130</td>
</tr>
<tr>
<td>MC_Get_Offset_Table_To_Function</td>
<td>134</td>
</tr>
<tr>
<td>MC_Get_pAttribute_Info</td>
<td>137</td>
</tr>
<tr>
<td>MC_Get_pTag_Info</td>
<td>138</td>
</tr>
<tr>
<td>MC_Get_pTags_Dict_Info</td>
<td>139</td>
</tr>
<tr>
<td>MC_Get_pValue... Functions</td>
<td>140</td>
</tr>
<tr>
<td>MC_Get_pValue_Count</td>
<td>145</td>
</tr>
<tr>
<td>MC_Get_pValue_Length</td>
<td>147</td>
</tr>
<tr>
<td>MC_Get_pValue_To_Function</td>
<td>148</td>
</tr>
<tr>
<td>MC_Get_Stream_Length</td>
<td>151</td>
</tr>
<tr>
<td>MC_Get_String_Config_Value</td>
<td>153</td>
</tr>
<tr>
<td>MC_Get_Tag_Info</td>
<td>156</td>
</tr>
<tr>
<td>MC_Get_Tags_Dict_Info</td>
<td>157</td>
</tr>
<tr>
<td>MC_Get_Tag_Keyword</td>
<td>158</td>
</tr>
<tr>
<td>MC_Get_Transfer_Syntax_From_Enum</td>
<td>158</td>
</tr>
<tr>
<td>MC_Get_UID_From_MergeCOM_Service</td>
<td>160</td>
</tr>
<tr>
<td>MC_Get_User_Identity_Info</td>
<td>161</td>
</tr>
<tr>
<td>MC_Get_User_Identity_Length</td>
<td>163</td>
</tr>
<tr>
<td>MC_Get_Value... Functions</td>
<td>164</td>
</tr>
<tr>
<td>MC_Get_Value_Count</td>
<td>168</td>
</tr>
<tr>
<td>MC_Get_Value_Length</td>
<td>169</td>
</tr>
<tr>
<td>MC_Get_Value_To_Function</td>
<td>171</td>
</tr>
<tr>
<td>MC_Get_Version_String</td>
<td>173</td>
</tr>
<tr>
<td>Function</td>
<td>Page</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>MC_Json_To_Message</td>
<td>173</td>
</tr>
<tr>
<td>MC_Library_INITIALIZATION</td>
<td>175</td>
</tr>
<tr>
<td>MC_Library_Release</td>
<td>176</td>
</tr>
<tr>
<td>MC_Library_Reset</td>
<td>177</td>
</tr>
<tr>
<td>MC_List_File (All toolkits except Windows versions)</td>
<td>178</td>
</tr>
<tr>
<td>MC_List_File (Windows toolkit versions)</td>
<td>179</td>
</tr>
<tr>
<td>MC_List_Item (All toolkits except Windows versions)</td>
<td>179</td>
</tr>
<tr>
<td>MC_List_Item (Windows toolkit versions)</td>
<td>180</td>
</tr>
<tr>
<td>MC_List_Message (All toolkits except Windows versions)</td>
<td>181</td>
</tr>
<tr>
<td>MC_List_Message (Windows toolkit versions)</td>
<td>182</td>
</tr>
<tr>
<td>MC_MemoryLog_To_Function</td>
<td>182</td>
</tr>
<tr>
<td>MC_Message_To_File</td>
<td>184</td>
</tr>
<tr>
<td>MC_Message_To.JSON</td>
<td>184</td>
</tr>
<tr>
<td>MC_Message_To_SR</td>
<td>187</td>
</tr>
<tr>
<td>MC_Message_To_Stream</td>
<td>187</td>
</tr>
<tr>
<td>MC_Message_To_XML</td>
<td>191</td>
</tr>
<tr>
<td>MC_Message_To_XML_Native</td>
<td>194</td>
</tr>
<tr>
<td>MC_NewProposedServiceList</td>
<td>196</td>
</tr>
<tr>
<td>MC_NewProposedServiceListAsync</td>
<td></td>
</tr>
<tr>
<td>MC_NewSyntaxList</td>
<td>197</td>
</tr>
<tr>
<td>MC_NewService... Functions</td>
<td>198</td>
</tr>
<tr>
<td>MC_Open_Association</td>
<td>200</td>
</tr>
<tr>
<td>MC_Open_Association_With_Connect_Timeout</td>
<td>200</td>
</tr>
<tr>
<td>MC_Open_Association_With_Identity</td>
<td>200</td>
</tr>
<tr>
<td>MC_Open_Association_With_Identity_With_Reject_Info</td>
<td>200</td>
</tr>
<tr>
<td>MC_Open_Association_With_Identity_With_Reject_Info_With_Connect_Timeout</td>
<td>200</td>
</tr>
<tr>
<td>MC_Open_Association_With_Reject_Info</td>
<td>200</td>
</tr>
<tr>
<td>MC_Open_Association_With_Callback</td>
<td>200</td>
</tr>
<tr>
<td>MC_Open_Association_With_All_Optional_Parameters</td>
<td>200</td>
</tr>
<tr>
<td>MC_Open_Empty_Item</td>
<td>207</td>
</tr>
<tr>
<td>MC_Open_Empty_Message</td>
<td>208</td>
</tr>
<tr>
<td>MC_Open_File</td>
<td></td>
</tr>
<tr>
<td>MC_Open_File_Bypass_OBOW</td>
<td>209</td>
</tr>
<tr>
<td>MC_Open_File_Upto_Tag</td>
<td>209</td>
</tr>
<tr>
<td>MC_Open_File_Upto_Tag_Bypass_Value</td>
<td></td>
</tr>
<tr>
<td>MC_Open_Item</td>
<td>214</td>
</tr>
<tr>
<td>MC_Open_Message</td>
<td>215</td>
</tr>
<tr>
<td>MC_Open_Secure_Association</td>
<td>217</td>
</tr>
<tr>
<td>MC_Open_Secure_Association_With_Reject_Info</td>
<td>217</td>
</tr>
<tr>
<td>MC_Process_Association_Request</td>
<td>222</td>
</tr>
<tr>
<td>MC_Read_Message</td>
<td>225</td>
</tr>
<tr>
<td>MC_Read_Message_To_Tag</td>
<td>228</td>
</tr>
</tbody>
</table>
MC_Read_To_Stream ..............................................................231
MC_Register_Application .....................................................234
MC_Register_Callback_FUNCTION MC_Register_pCallback_FUNCTION ..................................................235
MC_Register_Compression_Callbacks .........................................243
MC_Register_Enhanced_MemoryLog_FUNCTION .............................246
MC_Register_MemoryLog_FUNCTION ........................................248
MC_Register_Network_Capture_Callbacks (deprecated) .................250
MC_Reject_Association MC_Reject_Association_With_Reason_Codes ...................................................253
MC_Release_Application ........................................................255
MC_Release_Callback_FUNCTION MC_Release_pCallback_FUNCTION ..................................................255
MC_Release_Library_Exception_Handler .....................................256
MC_Release_Parent_Association (UNIX Only) ..................................257
MC_Release_Parent_Connection (UNIX Only) .................................258
MC_Report_Memory .............................................................258
MC_Reset_Filename ...............................................................259
MC_Reset_Message_Transfer_Syntax ........................................260
MC_Send_Request_Message .....................................................261
MC_Send_Request_Message_For_Service ....................................263
MC_Send_Request .......................................................................265
MC_Send_Request_For_Service ................................................267
MC_Send_Response_Message ....................................................269
MC_Send_Response ...............................................................271
MC_Set_Bool_Config_Value .....................................................273
MC_Set_Encapsulated_Value_From_Function ................................275
MC_Set_File_Preamble ..........................................................278
MC_Set_Int_Config_Value .........................................................278
MC_Set_Library_Exception_Handler ...........................................280
MC_Set_Log_Destination ........................................................281
MC_Set_Log_Prefix .................................................................282
MC_Set_Long_Config_Value .....................................................282
MC_Set_MergeINI MC_Set_MergeINI_Unicode ................................283
MC_Set_Message_Callbacks ......................................................284
MC_Set_Message_Transfer_Syntax ............................................285
MC_Set_Negotiation_Info .........................................................286
MC_Set_Negotiation_Info_For_Association .................................287
MC_Set_Next_Encapsulated_Value_From_Function ...........................289
MC_Set_Next_pValue... Functions ...........................................291
MC_Set_Next_pValue_To_NULL .................................................296

© Copyright Merge Healthcare Solutions Inc. 2023
MC_Set_Next_Value... Functions ................................................................. 297
MC_Set_Next_Value_To_NULL ................................................................... 302
MC_Set_pValue... Functions ................................................................... 303
MC_Set_pValue_From_Function ............................................................. 308
MC_Set_pValue_Representation ............................................................ 311
MC_Set_pValue_To_Empty ....................................................................... 312
MC_Set_pValue_To_NULL ....................................................................... 314
MC_Set_Service_Command ..................................................................... 315
MC_Set_String_Config_Value ................................................................. 317
MC_Set_Value... Functions ................................................................... 320
MC_Set_Value_From_Function ............................................................. 325
MC_Set_Value_Representation ............................................................... 327
MC_Set_Value_To_Empty ....................................................................... 328
MC_Set_Value_To_NULL ....................................................................... 330
MC_SR_Add_Child .................................................................................. 331
MC_SR_Add_Root .................................................................................. 331
MC_SR_Delete_Child .............................................................................. 332
MC_SR_Get_Child_Count ........................................................................ 333
MC_SR_Get_First_Child ......................................................................... 334
MC_SR_Get_Location ............................................................................. 335
MC_SR_Get_Next_Child ......................................................................... 336
MC_SR_Get_Root ................................................................................... 336
MC_SR_To_Message ............................................................................... 337
MC_Stream_To_Message MC_Stream_To_Message_With_Offset ................. 338
MC_Thread_Release ............................................................................... 343
MC_Unicode_Get_Substitution_Characters .......................................... 344
MC_Unicode_To.Byte .......................................................................... 345
MC.Validate_Attribute .......................................................................... 346
MC.Validate_File .................................................................................. 349
MC.Validate_Message .......................................................................... 352
MC.Wait_For_Association MC.Wait_For_Association_On_Port MC.Wait_For_Association_On_Address 355
MC.Wait_For_Connection MC.Wait_For_Connection_On_Port MC.Wait_For_Connection_On_Address 358
MC.Wait_For_Secure_Association MC.Wait_For_Secure_Association_On_Port
MC.Wait_For_Secure_Association_On_Address ........................................ 360
MC_Write_File MC_Write_File_By_Callback ......................................... 365
MC.XML_To_Message ............................................................................ 368
MC.XML_Native_To_Message .................................................................. 370
High Level API Reference .................................................................... 371
MC_DDH_Create .................................................................371
MC_DDH_Open .................................................................373
MC_DDH_Update .............................................................374
MC_DDH_Traverse_Records ..............................................376
MC_DDH_Get_Record_Type ...............................................377
MC_DDH_Get_Parent_Record ............................................378
MC_DDH_Get_Next_Record ...............................................379
MC_DDH_Get_First_Lower_Record .................................381
MC_DDH_Copy_Values ......................................................382
MC_DDH_Add_Record .......................................................383
MC_DDH_Delete_Record ....................................................385
MC_DDH_Release_Record ..................................................386
MC_SRH_Create_SR ..........................................................387
MC_SRH_Free_SR .............................................................388
MC_SRH_Add_child ..........................................................388
MC_SRH_Add_TEXT_Child ..................................................389
MC_SRH_Create(TEXT)_Node ............................................390
MC_SRH_Set_Concept_Name ..............................................391
MC_SRH_Create_CODE_Child ..............................................392
MC_SRH_Create_CODE_Node .............................................393
MC_SRH_CREATE_NUM_Child ............................................394
MC_SRH_Create_NUM_Node ...............................................395
MC_SRH_Set_NUM_Qualifier .............................................396
MC_SRH_SET_NUM_Next_Data .........................................397
MC_SRH_ADD_DATETIME_Child ........................................397
MC_SRH_Create_DATETIME_Node ......................................398
MC_SRH_ADD_DATE_Child ..................................................399
MC_SRH_Create_DATE_Node .............................................400
MC_SRH_ADD_TIME_Child ..................................................401
MC_SRH_Create_TIME_Node .............................................402
MC_SRH_ADD_UIDREF_Child ..............................................403
MC_SRH_Create_UIDREF_Node .........................................404
MC_SRH_ADD_PNAME_Child ...............................................405
MC_SRH_Create_PNAME_Node ..........................................406
MC_SRH_ADD_SCOORD_Child ............................................407
MC_SRH_SET_SCOORD_Next_Data ...................................408
MC_SRH_CREATE_SCOORD_Node ......................................409
MC_SRH_ADD_TCOORD_R_Child ........................................411
Service Profile
Overview

This reference manual contains a detailed description of the functionality of the Merge DICOM Toolkit Library. This includes library configuration, application programmer's interface (API) specification, and a DICOM conformance statement for the toolkit.

Structure of Documentation

The Merge DICOM Toolkit documentation is structured as shown in Figure 1.

*Figure 1: Merge DICOM Toolkit Documentation Roadmap*

The User’s Manual is the foundation for all other documentation because it explains the concepts of DICOM and the toolkit. 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 DICOM Toolkit. This includes the Application Programming Interface (API), toolkit configuration, the runtime object database, and status logging. The Reference Manual also includes a DICOM conformance statement for the toolkit.

The DICOM Extended Toolkit Manual is an optional extension that describes the organization of the Merge DICOM Toolkit DICOM Database and how to use it 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.

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 DICOM Toolkit. 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 DICOM Toolkit on your target platform are
specified in Platform Notes. This includes supported compilers, compiler options, link options,
configuration, and run-time related issues.

**Conventions**

This manual follows a few formatting conventions.

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

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

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

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.
Configuration

Toolkit configuration is accomplished through the use of initialization or configuration files. Initialization files, also called “ini” files, contain configuration information the toolkit will use to initialize its internal settings.

Each of the four toolkit initialization files use 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.](image_url)

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 Merge DICOM Toolkit is very versatile and configurable, it uses initialization files extensively. The toolkit makes use of four configuration files: the Merge Initialization File, the Library Profile, the Application Profile and the Service Profile. The Appendix B: Configuration Parameters discusses each of these files in detail.

Any application using the Merge DICOM Toolkit 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 Merge DICOM Toolkit allows initialization files to be integrated into an application at “link” time. The toolkit 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.

Generating a Static Configuration Function (genconf)

The genconf utility supplied with Merge DICOM Toolkit 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 toolkit 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 environment 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 (tm)   Configuration Source Generator   Version 3.9.0
   (c) 2008 Merge OEM. 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 toolkit that was shipped to you. Using mixed version of the library and `genconf` will cause your applications to fail.

**Generating a Static Dictionary Function (gendict)**

The `gendict` utility supplied with Merge DICOM Toolkit 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 toolkit 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` environment 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 (tm) DICOM Dictionary Source Generator Version 3.9.0
   (c) 2008 Merge OEM. 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

Merge DICOM Toolkit 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 base API may be divided into eleven major groups:

1. **Library initialization and reset functions**
   Must be called initially and when wishing to reset the library to its initially configured state.

2. **Run time configuration**
   Allow your application to change configurable parameters at runtime.

3. **Application Registration**
   Register your DICOM application entity with the library.

4. **Association functions**
   Deal with opening DICOM associations and querying the characteristics of a proposed or open association.

5. **Message object functions**
   Deal with the creation and manipulation of message objects.

6. **Item object functions**
   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**
   Deal with the creation, manipulation, and freeing of file objects.

8. **Message, file and item object functions**
   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**.
Additional to the base API the toolkit provides higher level functionality that can facilitate the manipulation of specific DICOM objects. The current version of the toolkit contains the following categories of high level functions:

1. **DICOMDIR Manipulation Functions (MC_DDH_functions)**
   These functions provide enhanced access to DICOMDIR files.

2. **Structured Report Manipulation Functions (MC_SRH_functions)**
   These functions provide enhanced access to Structured Reporting (SR) objects.

**Library Initialization and Reset**

The Merge DICOM Toolkit 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 Merge DICOM Toolkit run-time configuration functions follow:

   - MC_Release_Library_Exception_Handler
   - MC_Set_Bool_Config_Value
   - MC_Set_Int_Config_Value
   - MC_Set_Library_Exception_Handler
   - 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 Merge DICOM Toolkit 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 Merge DICOM Toolkit association functions may be grouped as follows:
1. Association object creation and release.
   - MC_Open_Association
   - MC_Open_Association_With_Connect_Timeout
   - MC_Open_Association_With_Identity
   - MC_Open_Association_With_Identity_With_Reject_Info
   - MC_Open_Association_With_Identity_With_Reject_Info_With_Connect_Timeout
   - MC_Open_Association_With_Reject_Info
   - MC_Open_Association_With_Callback
   - MC_Open_Association_With_AllOptional_Parameters
   - MC_Open_Secure_Association
   - MC_Open_Secure_Association_With_Reject_Info
   - MC_Close_Association
   - MC_Abort_Association
   - MC_Release_Parent_Association
   - MC_Wait_For_Association
   - MC_Wait_For_Association_On_Port
   - MC_Wait_For_Connection
   - MC_Wait_For_Connection_On_Port
   - MC_Wait_For_Secure_Association
   - MC_Wait_For_Secure_Association_On_Port
   - MC_Close_Listen_Port
   - MC_Process_Association_Request
   - MC_Process_Secure_Association_Request
   - MC_Accept_Association
   - MC_Accept_Association_With_Identity
   - MC_Reject_Association
   - MC_Reject_Association_With_Reason_Codes
   - MC_Set_Negotiation_Info (deprecated)
   - MC_Set_Negotiation_Info_For_Association
   - MC_Get_Negotiation_Info
   - MC_Clear_Negotiation_Info (deprecated)
   - 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
- MC_Get_User_Identity_Info
- MC_Get_User_Identity_Length

**Message Object Functions**

The Merge DICOM Toolkit 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 ensuring 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

7. Message XML conversion functions: those dealing with converting a message to an XML string or reading attributes from an XML string into a message using either Merge DICOM Model or Native DICOM Model.
MC_Message_To_XML 
MC.XML_To_Message 
MC_Message_To_XML_Native 
MC.XML_Native_To_Message 

8. Message JSON conversion functions: those dealing with converting a message to an JSON string or reading attributes from a JSON string into a message using DICOM JSON Model.

MC_Message_To_Json 
MC.Json_To_Message 

Item Object Functions
The Merge DICOM Toolkit 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 Merge DICOM Toolkit 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 ensuring 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.Open_File.Upto_Tag.Bypass_Value 
   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 Merge DICOM Toolkit message and item object functions can be grouped as follows:

1. 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

   - 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

3. 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

4. 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

5. 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

6. 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_To_Function
MC_Get_Next_Encapsulated_Value_To_Function
MC_Get_Frame_To_Function
MC_Get_Offset_Table_To_Function

7. 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 Merge DICOM Toolkit message transfer functions follow:

MC_Send_Request_Message
MC_Send_Request_Message_For_Service
MC_Send_Request
MC_Send_Request_For_Service
MC_Send_Response_Message
MC_Send_Response
MC_Read_Message
MC_Read_Message_To_Tag
MC_Continue_Read_Message
MC_Continue_Read_Message_To_Tag
MC_Read_To_Stream
MC_Reset_Message_Transfer_Syntax
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

6. **DICOMDIR storage functions**: this function which allows users to set the storage for specific directory records in a DICOMDIR. This routine reduces memory usage for DICOMDIRs.
   - MC_Dir_Stream_Directory_Record
Structured Reporting Object Functions

The Structured Reporting object functions are grouped as follows:

1. Getting values of SR objects:
   - MC_SR_Get_First_Child
   - MC_SR_Get_Next_Child
   - MC_SR_Get_Child_Count
   - MC_SR_Get_Root

2. Adding values to SR objects:
   - MC_SR_Add_Child
   - MC_SR_Add_Root
   - MC_SR_Get_Location

3. Converting messages to or from SR objects:
   - MC_Message_To_SR
   - MC_SR_To_Message

4. 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_Tag_Keyword
- MC_Get_Enum_From_Transfer_Syntax
- MC_Get_Transfer_Syntax_From_Enum
- MC_Set_Log_Prefix
- MC_Enable_Unicode_Conversion
- MC_Byte_To_Unicode
- MC_Unicode_To_Byte
- MC_Register_Network_Capture_Callbacks

Enhanced DICOMDIR Manipulation Functions

The enhanced DICOMDIR manipulation functions may be grouped as follows:

1. DICOMDIR file functions: functions that deal with reading and creating DICOMDIR files.
   - MC_DDH_Create
   - MC_DDH_Open
2. Directory record navigation functions: functions that deal with the traversal of the directory record hierarchy:

   MC_DDH_Get_First_Lower_Record
   MC_DDH_Get_Next_Record
   MC_DDH_Get_Parent_Record
   MC_DDH_Traverse_Records
   MC_DDH_Release_Record
   MC_DDH_Get_Record_Type

3. DICODIR modification function: functions that deal with modifying the directory record hierarchy:

   MC_DDH_Add_Record
   MC_DDH_Delete_Record
   MC_DDH_Update
   MC_DDH_Copy_Values

High Level Structured Reporting Manipulation Functions

The High Level Structured Reporting (SR) manipulation functions are grouped as follows:

1. SR creation and setup function:

   MC_SRH_Create_SR
   MC_SRH_Free_SR
   MC_SRH_Add_TEXT_Child
   MC_SRH_Add_CODE_Child
   MC_SRH_Set_NUM_Qualifier
   MC_SRH_Set_NUM_Next_Data
   MC_SRH_Add_NUM_Child
   MC_SRH_Add_DATETIME_Child
   MC_SRH_Add_DATE_Child
   MC_SRH_Add_TIME_Child
   MC_SRH_Add_UIDREF_Child
   MC_SRH_Add_PNAME_Child
   MC_SRH_Add_SCOORD_Child
   MC_SRH_Set_SCOORD_Next_Data
   MC_SRH_Add_TCOORD_R_Child
   MC_SRH_Set_TCOORD_R_Next_Data
   MC_SRH_Add_TCOORD_O_Child
   MC_SRH_Set_TCOORD_O_Next_Data
   MC_SRH_Add_TCOORD_D_Child
   MC_SRH_Set_TCOORD_D_Next_Data
   MC_SRH_Add_COMPOSITE_Child
   MC_SRH_Add_IMAGE_Child
   MC_SRH_Set_IMAGE_Frames
   MC_SRH_Add_WAVEFORM_Child
   MC_SRH_Set_WAVEFORM_Channels
   MC_SRH_Add_CONTAINER_Child
MC_SRH_Add_TABLE_Child
MC_SRH_Set_TABLE_Next_Row_Definition
MC_SRH_Set_TABLE_Next_Column_Definition
MC_SRH_Set_TABLE_Next_Cell_Values
MC_SRH_Set_Concept_Name

2. SR parsing and reading functions:

MC_SRH_Get_NodeType
MC_SRH_Get_First_Child
MC_SRH_Get_Next_Child
MC_SRH_Get_Concept_Name
MC_SRH_Get_TEXT_Data
MC_SRH_Get_CODE_Data
MC_SRH_Get_NUM_Data
MC_SRH_Get_NUM_Next_Data
MC_SRH_Get_NUM_Qualifier
MC_SRH_Get_DATETIME_Data
MC_SRH_Get_DATE_Data
MC_SRH_Get_TIME_Data
MC_SRH_Get_UIDREF_Data
MC_SRH_Get_PNAME_Data
MC_SRH_Get_SCOORD_First_Data
MC_SRH_Get_SCOORD_Next_Data
MC_SRH_Get_TCOORD_D_First_Data
MC_SRH_Get_TCOORD_D_Next_Data
MC_SRH_Get_TCOORD_O_First_Data
MC_SRH_Get_TCOORD_O_Next_Data
MC_SRH_Get_TCOORD_R_First_Data
MC_SRH_Get_TCOORD_R_Next_Data
MC_SRH_Get_COMPOSITE_Data
MC_SRH_Get_IMAGE_Data
MC_SRH_Get_IMAGE_Frames
MC_SRH_Get_WAVEFORM_Data
MC_SRH_Get_WAVEFORM_Channels
MC_SRH_Get_TABLE_First_Cell_Values_Counts
MC_SRH_Get_TABLE_First_Column_Definition
MC_SRH_Get_TABLE_First_Row_Definition
MC_SRH_Get_TABLE_Next_Cell_Values_Counts
MC_SRH_Get_TABLE_Next_Column_Definition
MC_SRH_Get_TABLE_Next_Row_Definition
MC_SRH_Get_TABLE_Cell_Values_Item
MC_SRH_Get_CONTAINER_Data
MC_SRH_Get_Reference
3. SR utility functions:

    MC_SRH_Add_Child
    MC_SRH_Add_Reference
    MC_SRH_Create_TEXT_Node
    MC_SRH_Create_CODE_Node
    MC_SRH_Create_NUM_Node
    MC_SRH_Create_DATETIME_Node
    MC_SRH_Create_DATE_Node
    MC_SRH_Create_TIME_Node
    MC_SRH_Create_UIDREF_Node
    MC_SRH_Create_PNAME_Node
    MC_SRH_Create_SCOORD_Node
    MC_SRH_Create_TCOORD_R_Node
    MC_SRH_Create_TCOORD_O_Node
    MC_SRH_Create_TCOORD_D_Node
    MC_SRH_Create_COMPOSITE_Node
    MC_SRH_Create_IMAGE_Node
    MC_SRH_Create_WAVEFORM_Node
    MC_SRH_Create_TABLE_Node
    MC_SRH_Create_CONTAINER_Node

**API Function Reference**

The next section of this manual contains an alphabetically arranged function reference section. Each function reference provides:

- 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
- A “See also” cross reference to other functions.

**MC_Abort_Association**

Terminates an open association immediately.

**Synopsis**

```c
#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 the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>The AssociationID parameter was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_ASSOC_ID</td>
<td>*AssociationID is not a valid association object ID.</td>
</tr>
<tr>
<td>MC_LIBRARY_NOT_INITIALIZED</td>
<td>The library has not been properly initialized.</td>
</tr>
</tbody>
</table>

See Also

MC_Close_Association
MC_Open_Association
MC_Open_Secure_Association
MC_Wait_For_Association
MC_Wait_For_Secure_Association

MC_Accept_Association MC_Accept_Association_With_Identity

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

Synopsis

```c
#include "mergecom.h"

MC_STATUS MC_Accept_Association (int AssociationID)

MC_STATUS MC_Accept_Association_With_Identity (int AssociationID,
                                               void *ServerResponse,
                                               unsigned short ServerResponseLength)
```

© Copyright Merge Healthcare Solutions Inc. 2023
**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.

`MC_Accept_Association_With_Identity` can be used to send a user identity response to the association request when requested for the association as defined in DICOM Supplement 99. User Identity information within the association request can be retrieved through the use of the `MC_Get_Association_Info`, `MC_Get_User_Identity_Info`, and `MC_Get_User_Identity_Length` functions.

If the association request does not request a positive response or if no user identity information was contained in the association request, the `MC_Accept_Association` function should be used. When a positive response is requested, the `MC_Accept_Association_With_Identity` function can be used. If no server response information is required, the `ServerResponse` field can be set to NULL and the `ServerResponseLength` field can be set to 0.

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 the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MC_NORMAL_COMPLETION</code></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><code>MC_INVALID_ASSOC_ID</code></td>
<td><code>AssociationID</code> is not a valid association object ID.</td>
</tr>
<tr>
<td><code>MC_NO_REQUEST_PENDING</code></td>
<td>There are no pending association request for this <code>AssociationID</code>.</td>
</tr>
<tr>
<td><code>MC_SYSTEM_ERROR</code></td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td><code>MC_ASSOCIATION_ABORTED</code></td>
<td>The association has been aborted.</td>
</tr>
</tbody>
</table>

**See Also**

`MC_Wait_For_Association` `MC_Get_Association_Info` `MC_Get_User_Identity_Info` `MC_Open_Association_With_Identity` `MC_Reject_Association`
**MC_Add_Nonstandard_Attribute**

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

**Synopsis**

```cpp
#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`, `UC`, `UR`,  
  `UT`, `UI`, `SS`, `US`, `AT`, `SL`, `UL`, `SV`, `UV`, `FL`, `FD`, `OB`, `OW`, `OV`, `OL`,  
  `OD`, `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 defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MC_NORMAL_COMPLETION</code></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><code>MC_INVALID_TAG</code></td>
<td>The attribute with an ID of <code>Tag</code> is a standard attribute found in the Data Dictionary. Instead, use <code>MC_Add_Standard_Attribute</code>.</td>
</tr>
<tr>
<td><code>MC_TAG_ALREADY_EXISTS</code></td>
<td>The message already contains an attribute with an ID of <code>Tag</code>.</td>
</tr>
<tr>
<td><code>MC_INVALID_VR_CODE</code></td>
<td><code>ValueRep</code> is invalid.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td><code>MsgFileItemID</code> is not a valid message object ID, file object ID, or item object ID.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**See Also**
- MC_Add_Standard_Attribute
- MC_Delete_Attribute
- MC_Add_Private_Attribute
- MC_Delete_Private_Attribute
- MC_Add_Private_Block
- MC_Delete_Private_Block
- MC_Delete_Range

### MC_Add_Private_Attribute

Adds a private DICOM attribute to a message object.

**Synopsis**

```c
#include "mc3msg.h"

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MsgFileItemID</code></td>
<td>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.</td>
</tr>
<tr>
<td><code>PrivateCode</code></td>
<td>The code string which identifies which block in the private Group is to “own” the new attribute.</td>
</tr>
<tr>
<td><code>Group</code></td>
<td>The number identifying the private group. It must be an odd number.</td>
</tr>
<tr>
<td><code>ElementByte</code></td>
<td>The number identifying this attribute within the private block.</td>
</tr>
<tr>
<td><code>ValueRep</code></td>
<td>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, UC, UR, UT, UI, SS, US, SV, UV, AT, SL, UL, FL, FD, OB, OW, OV, OL, OD, OF, SQ</td>
</tr>
</tbody>
</table>
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 defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>PrivateCode was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>MsgFileItemID is not a valid message object ID, file object ID, or item object ID.</td>
</tr>
<tr>
<td>MC_NOT_FOUND</td>
<td>There is no private block identified by PrivateCode.</td>
</tr>
<tr>
<td>MC_INVALID_GROUP</td>
<td>Group is not an odd number.</td>
</tr>
<tr>
<td>MC_INVALID_VR_CODE</td>
<td>ValueRep is invalid.</td>
</tr>
<tr>
<td>MC_TAG_ALREADY_EXISTS</td>
<td>The private attribute already exists in the message object.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

See Also

MC_Add_Standard_Attribute    MC_Delete_Attribute
MC_Add_Private_Attribute     MC_Delete_Private_Attribute
MC_Add_Nonstandard_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 defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td><code>PrivateCode</code> was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td><code>MsgFileItemID</code> is not a valid message object ID, file object ID or item object ID.</td>
</tr>
<tr>
<td>MC_INVALID_GROUP</td>
<td><code>Group</code> is not an odd number.</td>
</tr>
<tr>
<td>MC_INVALID_VR_CODE</td>
<td><code>ValueRep</code> is invalid.</td>
</tr>
<tr>
<td>MC_TOO_MANY_BLOCKS</td>
<td>There already exists the maximum 240 private blocks in the group.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>
See Also
MC_Add_Standard_Attribute  MC_Delete_Attribute
MC_Add_Private_Attribute  MC_Delete_Private_Attribute
MC_Add_Nonstandard_Attribute  MC_Delete_Private_Block
MC_Delete_Range

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 defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>MsgFileItemID is not a valid message object ID, file object ID or item object ID.</td>
</tr>
<tr>
<td>MC_TAG_ALREADY_EXISTS</td>
<td>The message already contains an attribute with an ID of Tag.</td>
</tr>
<tr>
<td>MC_INVALID_TAG</td>
<td>The Data Dictionary does not contain an attribute with an ID of Tag.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>
See Also

MC_Add_Nonstandard_Attribute  MC_Delete_Attribute
MC_Add_Private_Attribute  MC_Delete_Private_Attribute
MC_Add_Private_Block  MC_Delete_Private_Block
MC_Delete_Range

MC_Byte_To_UniCode

A utility function to convert DICOM character set to Unicode.

Synopsis

```c
#include "mc3msg.h"

MC_STATUS MC_Byte_To_UniCode (  
    char *DefaultCharset,  
    const char *Val,  
    int InLen,  
    intBufferSize,  
    int *OutLen,  
    MC_Uhar *OutVal  
) 
```

**DefaultCharset**  The DICOM specific character set name for the default character set.

**Val**  Input bytes array

**InLen**  Input bytes count (excluding NULL terminator)

**BufferSize**  Output buffer size

**OutLen**  Output length (returned by this call)

**OutVal**  Output buffer to hold the Unicode

Remarks

MC_Byte_To_UniCode requires MC_Enable_Unicode_Conversion being called first.

DefaultCharset should be set to the first character set of (0008,0005). If set to NULL or empty string, the default ISO-IR 6 (ASCII) will be used.

InLen can be set to -1 and the toolkit will calculate length if the input buffer is NULL terminated.

When the return status is MC_NORMAL_COMPLETION, OutVal contains the output Unicode string with U+0000 terminator. OutLen contains the number of Unicode characters present in the OutVal buffer (excluding the U+0000 terminator).

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
</tbody>
</table>

© Copyright Merge Healthcare Solutions Inc. 2023
**MC_CANNOT_COMPLY**
Fail to process the input. Check the Merge DICOM Toolkit log file for detail of failure.

**MC_BUFFER_TOO_SMALL**
The output buffer doesn’t have enough space to hold the output.

**MC_SYSTEM_ERROR**
An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.

**See Also**
- MC_Unicode_To_Byte
- MC_Enable_Unicode_Conversion
- MC_Get_Value_To_UnicodeString
- MC_Get_Next_Value_To_UnicodeString
- MC_Set_Value_From_UnicodeString
- MC_Set_Next_Value_From_UnicodeString

### MC_BByte_Swap_OBOW

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

**Synopsis**

```c
#include "mc3msg.h"

MC_STATUS MC_BByte_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_BByte_Swap_OBOW** byte swaps attributes of value representation OB, OW, or OF.

**MC_BByte_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 Merge DICOM Toolkit 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 defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td><em>MsgFileItemID</em> is not a valid message object ID, file object ID, or item object ID.</td>
</tr>
</tbody>
</table>
MC_INVALID_TAG  
Tag is not a valid tag in MsgFileItemID.

MC_EMPTY_VALUE  
The attribute has no value assigned to it.

MC_INCOMPATIBLE_VR  
The VR of Tag was not OB, OW, OD or OF.

MC_CALLBACK_REGISTERED  
This function is not allowed when the MC_Register_Callback_Function function was issued for the same 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 Merge DICOM Toolkit 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 Merge DICOM Toolkit library.

#### Synopsis

```c
#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 the enumerated MC_STATUS codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_LIBRARY_NOT_INITIALIZED</td>
<td>Call made prior to library initialization</td>
</tr>
<tr>
<td>MC_TIMEOUT</td>
<td>The timeout expired before the entire cleanup was finished.</td>
</tr>
<tr>
<td>MC_VALUE_OUT_OF_RANGE</td>
<td>The timeout was less than 0.</td>
</tr>
</tbody>
</table>
**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 Merge DICOM Toolkit with extended negotiation information which it will use when establishing associations.

When a **MC_Open_Association**, **MC_Open_Association_With_Identity** or **MC_Open_Secure_Association** call is made, Merge DICOM Toolkit 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, Merge DICOM Toolkit 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. Merge DICOM Toolkit 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 the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MC_NORMAL_COMPLETION</code></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><code>MC_INVALID_APPLICATION_ID</code></td>
<td><code>ApplicationID</code> is not a valid application identifier.</td>
</tr>
<tr>
<td><code>MC_NULL_POINTER_PARM</code></td>
<td><code>ServiceName</code> was NULL.</td>
</tr>
<tr>
<td><code>MC_UNKNOWN_SERVICE</code></td>
<td><code>ServiceName</code> was not registered in the Merge DICOM Toolkit configuration files.</td>
</tr>
<tr>
<td><code>MC_NO_INFO_REGISTERED</code></td>
<td>There was no extended negotiation information registered for <code>ServiceName</code> for this application.</td>
</tr>
</tbody>
</table>

### 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

```c
#include "mergecom.h"

MC_STATUS MC_Close_Association (  
    int *AssociationID
)
```

- `AssociationID` The identifier assigned to this object by the `MC_Open_Association`, `MC_Open_Secure_Association`, `MC_Process_Association_Request`, `MC_Process_Secure_Association_Request`, `MC_Wait_For_Association` or `MC_Wait_For_Secure_Association` function.

#### 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 the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>The <code>AssociationID</code> parameter was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_ASSOC_ID</td>
<td>‘<code>AssociationID</code>’ is not a valid association object ID.</td>
</tr>
<tr>
<td>MC_ASSOCIATION_ABORTED</td>
<td>There was an error while closing the association. Error message logged.</td>
</tr>
</tbody>
</table>

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**

```c
#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 the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
</tbody>
</table>
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
MC_Close_Listen_Port_On_Address

Stops listening for incoming DICOM connections on a port

Synopsis
#include “mergecom.h”
MC_STATUS MC_Close_Listen_Port (  
  int Port
)
MC_STATUS MC_Close_Listen_Port_On_Address (  
  int Port
  const char *Address
)

Port  Listen port opened by a call to
  MC_Wait_For_Association,
  MC_Wait_For_Association_On_Port,
  MC_Wait_For_Association_On_Address,
  MC_Wait_For_Connection,
  MC_Wait_For_Connection_On_Port,
  MC_Wait_For_Connection_On_Address
  MC_Wait_For_Secure_Association,
  MC_Wait_For_Secure_Association_On_Port or
  MC_Wait_For_Secure_Association_On_Address
  function.

Address  The network name or IP address of the network interface
  on which the listener was started by a call to
  MC_Wait_For_Association_On_Address,
  MC_Wait_For_Connection_On_Address or
  MC_Wait_For_Secure_Association_On_Address
  function, exactly as passed to those functions.

Remarks
MC_Close_Listen_Port causes Merge DICOM Toolkit 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
TCP/IP_LISTEN_PORT option from MC_Wait_For_Association, MC_Wait_For_Connection 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, MC_Wait_For_Connection_On_Port, or MC_Wait_For_Secure_Association_On_Port functions. If a specific network interface was specified for the listener via a call to MC_Wait_For_Association_On_Address, MC_Wait_For_Connection_On_Address or MC_Wait_For_Secure_Association_On_Address, MC_Close_ListenPort_On_Address should be used to stop the listener.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_LIBRARY_NOT_INITIALIZED</td>
<td>The library has not been properly initialized.</td>
</tr>
<tr>
<td>MC_INVALID_PORT_NUMBER</td>
<td>Port is not a valid listen port.</td>
</tr>
</tbody>
</table>

**See Also**

MC_Wait_For_Association, MC_Wait_For_Association_On_Port
MC_Wait_For_Connection, MC_Wait_For_Connection_On_Port
MC_Wait_For_Secure_Association, MC_Wait_For_Secure_Association_On_Address
MC_Wait_For_Secure_Association_On_Address
MC_Continue_Read_Message

**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**

```c
#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 the enumerated `MC_STATUS` codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td><code>MessageID</code> is not a valid message object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>The <code>MessageID</code> parameter was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_ASSOC_ID</td>
<td><code>AssociationID</code> is not a valid association object ID.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NETWORK_SHUT_DOWN</td>
<td>The network connect unexpectedly dropped.</td>
</tr>
<tr>
<td>MC_ASSOCIATION_ABORTED</td>
<td>The association has been aborted. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_ASSOCIATION_CLOSED</td>
<td>The association has been closed.</td>
</tr>
<tr>
<td>MC_INACTIVITY_TIMEOUT</td>
<td>A timeout occurred in the middle of receiving a message.</td>
</tr>
<tr>
<td>MC_CONFIG_INFO_ERROR</td>
<td>The message information file describing the message’s service/command pair could not be accessed.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_RECEIVED</td>
<td>An improperly formatted DICOM message was received. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

See Also

- `MC_Read_Message`
- `MC_Continue_Read_Message_To_Tag`
- `MC_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 application 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

```c
#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:

```c
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 Merge DICOM Toolkit is providing the first block of stream data.

- **CBisLast**  
Is TRUE (not zero) when Merge DICOM Toolkit 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 Merge DICOM Toolkit 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 the enumerated `MC_STATUS` codes defined in “mcstatus.h”: 
<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>MessageID is not a valid message object ID or MC_Read_Message_To_Tag was not called for MessageID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>The MessageID parameter was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_ASSOC_ID</td>
<td>AssociationID is not a valid association object ID.</td>
</tr>
<tr>
<td>MC_STATE_VIOLATION</td>
<td>AssociationID is not a valid association that has been accepted.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NETWORK_SHUT_DOWN</td>
<td>The network connect unexpectedly dropped.</td>
</tr>
<tr>
<td>MC_ASSOCIATION_ABORTED</td>
<td>The association has been aborted. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_ASSOCIATION_CLOSED</td>
<td>The association has been closed.</td>
</tr>
<tr>
<td>MC_INACTIVITY_TIMEOUT</td>
<td>A timeout occurred in the middle of receiving a message.</td>
</tr>
<tr>
<td>MC_CONFIG_INFO_ERROR</td>
<td>The message information file describing the message's service/command pair could not be accessed.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_RECEIVED</td>
<td>An improperly formatted DICOM message was received. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_CALLBACK_CANNOT_COMPLY</td>
<td>YourReceiveStreamFunction returned an error.</td>
</tr>
</tbody>
</table>

See Also

- MC_Read_Message
- MC_Read_Message_To_Tag
- MC_Continue_Read_Message
- MC_Get_Message_Transfer_Syntax
- MC_Continue_Read_Message_To_Tag

Continues the reading of the current message that arrived from the remote application 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**

```c
#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 the enumerated MC_STATUS codes defined in “mcstatus.h”:

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NETWORK_SHUT_DOWN</td>
<td>The network connect unexpectedly dropped.</td>
</tr>
<tr>
<td>MC_ASSOCIATION_ABORTED</td>
<td>The association has been aborted. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_ASSOCIATION_CLOSED</td>
<td>The association has been closed.</td>
</tr>
<tr>
<td>MC_INACTIVITY_TIMEOUT</td>
<td>A timeout occurred in the middle of receiving a message.</td>
</tr>
<tr>
<td>MC_CONFIG_INFO_ERROR</td>
<td>The message information file describing the message’s service/command pair could not be accessed.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_RECEIVED</td>
<td>An improperly formatted DICOM message was received. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>
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

```c
#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 the enumerated **MC_STATUS** codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>The FileIDPtr or Filename parameter was NULL.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment.</td>
</tr>
</tbody>
</table>

**See Also**

- **MC_Create_File**
- **MC_Open_Message**
- **MC_Open_Item**
- **MC_Reset_Filename**

**MC_Create_File**

Creates a new file object

**Synopsis**

```c
#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, Merge DICOM Toolkit 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 the enumerated `MC_STATUS` codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MC_NORMAL_COMPLETION</code></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><code>MC_NULL_POINTER_PARM</code></td>
<td>The <code>FileIDPtr</code>, <code>Filename</code>, or <code>ServiceName</code> parameter was NULL.</td>
</tr>
<tr>
<td><code>MC_INVALID_COMMAND</code></td>
<td><code>Command</code> is not a supported command.</td>
</tr>
<tr>
<td><code>MC_MSGFILE_ERROR</code></td>
<td>An error occurred attempting to access the configuration data file.</td>
</tr>
<tr>
<td><code>MC_SYSTEM_ERROR</code></td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

See Also

`MC_Create_Empty_File` `MC_Open_Message` `MC_Open_Item` `MC_Delete_Attribute`

Removes an attribute from a message object.

Synopsis

```c
#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 defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_TAG</td>
<td>The message does not contain an attribute with an ID of Tag.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>MsgFileItemID is not a valid message object ID, file object ID or item object ID.</td>
</tr>
</tbody>
</table>

**See Also**

MC_Add_Standard_Attribute  
MC_Delete_Range  
MC_Add_Private_Attribute  
MC_Delete_Private_Attribute  
MC_Add_Private_Block  
MC_Delete_Private_Block  
MC_Add_Nonstandard_Attribute  
MC_Delete_Current_Value

**MC_Delete_Current_Value**

Deletes the current value within an attribute’s value list.

**Synopsis**

```c
#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 value 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.

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 the enumerated **MC_STATUS** codes defined in “**mcstatus.h**”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
</tbody>
</table>
| **MC_INVALID_MESSAGE_ID**         | *MsgFileItemID* is not a valid message object ID, file object ID 
or item object ID.                                      |
| **MC_NO_MORE_VALUES**             | There are no more values for this attribute.            |
| **MC_INVALID_TAG**                | The message does not contain an attribute with an ID of *Tag*. |
| **MC_INCOMPATIBLE_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 Merge DICOM Toolkit log file.       |
**MC_Delete_Private_Attribute**

Removes a private attribute from a message object.

### Synopsis

```c
#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 defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_NULL_POINTER_PARM</strong></td>
<td><code>PrivateCode</code> was NULL.</td>
</tr>
<tr>
<td><strong>MC_INVALID_TAG</strong></td>
<td>The private attribute does not exist in the message object.</td>
</tr>
<tr>
<td><strong>MC_INVALID MESSAGE ID</strong></td>
<td><code>MsgFileItemID</code> is not a valid message object ID, file object ID or item object ID.</td>
</tr>
<tr>
<td><strong>MC_INVALID_GROUP</strong></td>
<td><code>Group</code> was not an odd number.</td>
</tr>
</tbody>
</table>
The private block identified by PrivateCode was not in Group.

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.

The code string which identifies which block in the private Group is to be deleted.

The number identifying the private group. It must be an odd number.

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.

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

The function completed normally.

PrivateCode was NULL.

MsgFileItemID is not a valid message object ID, file object ID or item object ID.

Group was not an odd number.

The private block identified by PrivateCode was not in Group.
**MC_Delete_Range**

Removes a range of attributes from a message object.

**Synopsis**

```c
#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 defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>(MsgFileItemID) is not a valid message object ID, file object ID or item object ID.</td>
</tr>
</tbody>
</table>

**See Also**

MC_Add_Specialized_Attribute MC_Delete_Attribute
MC_Add_Private_Attribute MC_Delete_Private_Attribute
MC_Add_Private_Block MC_Delete_Range
MC_Delete_Range
**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**

```c
#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 Merge DICOM Toolkit.

**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 the enumerated **MC_STATUS** codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_DICOMDIR_ID</td>
<td>The <strong>DirID</strong> value is not a valid DICOMDIR object ID.</td>
</tr>
</tbody>
</table>
MC_INVALID_ITEM_ID
The ItemID value is not a valid directory record object ID.

MC_NULL_POINTER_PARM
The NewEntityID,NewItemID, orNewItemName 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 Merge DICOM Toolkit log file.

MC_INVALID_LOWER_DIR_RECORD
The directory record type specified byNewItemName 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 typeNewItemName and appends it to the end of the entity object specified by EntityID. A pointer to the new directory record is returned inNewItemID.
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 Merge DICOM Toolkit to reflect the new record.

**Return Value**

One of the enumerated `MC_STATUS` codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_DICOMDIR_ID</td>
<td>The <code>DirID</code> value is not a valid DICOMDIR object ID.</td>
</tr>
<tr>
<td>MC_INVALID_ENTITY_ID</td>
<td>The <code>EntityID</code> value is not a valid entity object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>The <code>NewItemID</code> or <code>NewItemName</code> parameter was NULL.</td>
</tr>
<tr>
<td>MC_UNABLE_TO_GET_ITEM_ID</td>
<td>Merge DICOM Toolkit found a corrupted record ID in a directory record offset attribute while trying to traverse the DICOMDIR object pointed to by <code>DirID</code>.</td>
</tr>
<tr>
<td>MC_MSGFILE_ERROR</td>
<td>An error occurred attempting to access the configuration data file for the directory record.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_INVALID_LOWER_DIR_RECORD</td>
<td>The directory record type specified by <code>NewItemName</code> is an invalid lower level type for the parent directory record of <code>EntityID</code>.</td>
</tr>
<tr>
<td>MC_BAD_DIR_RECORD_TYPE</td>
<td>The directory record type specified in the parent directory record of <code>EntityID</code> is invalid.</td>
</tr>
</tbody>
</table>

**See Also**

<table>
<thead>
<tr>
<th>See Also</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_Dir_Add_Entity</td>
<td><code>MC_Dir_Delete_Record</code> <code>MC_Dir_DeleteReferenced_Entity</code> <code>MC_Dir_First_Record</code> <code>MC_Dir_Next_Entity</code> <code>MC_Dir_Next_Record</code> <code>MC_Dir_Root_Entity</code></td>
</tr>
</tbody>
</table>

**MC_Dir_Delete_Record**

Deletes a DICOMDIR directory record and any directory entities it references

**Synopsis**

```c
#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 Merge DICOM Toolkit.

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_DICOMDIR_ID</td>
<td>The DirID value is not a valid DICOMDIR object ID.</td>
</tr>
<tr>
<td>MC_INVALID_ITEM_ID</td>
<td>The ItemID value is not a valid record object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>The DirID parameter was NULL.</td>
</tr>
<tr>
<td>MC_UNABLE_TO_GET_ITEM_ID</td>
<td>Merge DICOM Toolkit found a corrupted record ID in a directory record offset attribute while trying to traverse the DICOMDIR object pointed to by DirID.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

See Also

MC_Dir_Add_Entity 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 the enumerated **MC_STATUS** codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_INVALID_DICOMDIR_ID</strong></td>
<td>The <strong>DirID</strong> value is not a valid DICOMDIR object ID.</td>
</tr>
<tr>
<td><strong>MC_INVALID_ITEM_ID</strong></td>
<td>The <strong>ItemID</strong> value is not a valid record object ID.</td>
</tr>
<tr>
<td><strong>MC_INVALID_ENTITY_ID</strong></td>
<td>The <strong>EntityID</strong> value is not a valid entity object ID.</td>
</tr>
<tr>
<td><strong>MC_NULL_POINTER_PARM</strong></td>
<td>The <strong>DirID</strong> parameter was NULL.</td>
</tr>
<tr>
<td><strong>MC_UNABLE_TO_GET_ITEM_ID</strong></td>
<td>Merge DICOM Toolkit found a corrupted record ID in a directory record offset attribute while trying to traverse the DICOMDIR object pointed to by <strong>DirID</strong>.</td>
</tr>
<tr>
<td><strong>MC_SYSTEM_ERROR</strong></td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

### 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

```c
#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 the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_DICOMDIR_ID</td>
<td>The <code>DirID</code> value is not a valid DICOMDIR object ID.</td>
</tr>
<tr>
<td>MC_INVALID_ITEM_ID</td>
<td>The <code>ItemID</code> value is not a valid directory record object ID.</td>
</tr>
<tr>
<td>MC_INVALID_ENTITY_ID</td>
<td>The <code>EntityID</code> value is not a valid DICOMDIR entity object ID.</td>
</tr>
<tr>
<td>MC_UNABLE_TO_GET_ITEM_ID</td>
<td>The <code>DirID</code> value does not contain a reference to a valid root item.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>The <code>Count</code> parameter was NULL.</td>
</tr>
<tr>
<td>MC_MSGFILE_ERROR</td>
<td>An error occurred attempting to access the configuration data file for the directory record.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

#### 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
```c
#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  
icreate 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 the enumerated **MC_STATUS** codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_INVALID_DICOMDIR_ID</strong></td>
<td>The <strong>DirID</strong> value is not a valid DICOMDIR object ID.</td>
</tr>
<tr>
<td><strong>MC_INVALID_ENTITY_ID</strong></td>
<td>The <strong>EntityID</strong> value is not a valid entity object ID.</td>
</tr>
<tr>
<td><strong>MC_NULL_POINTER_PARM</strong></td>
<td>The <strong>ItemID</strong>, <strong>ItemName</strong>, or <strong>IsLast</strong> parameter was NULL.</td>
</tr>
<tr>
<td><strong>MC_UNABLE_TO_GET_ITEM_ID</strong></td>
<td>Merge DICOM Toolkit found a corrupted record ID in a directory record offset attribute while trying to traverse the DICOMDIR object pointed to by <strong>DirID</strong>.</td>
</tr>
<tr>
<td><strong>MC_EMPTY_ROOT_ENTITY</strong></td>
<td>The entity requested is empty.</td>
</tr>
</tbody>
</table>

**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

```c
#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 the enumerated `MC_STATUS` codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_DICOMDIR_ID</td>
<td>The <code>DirID</code> value is not a valid DICOMDIR object ID.</td>
</tr>
<tr>
<td>MC_INVALID_ITEM_ID</td>
<td>The <code>ItemID</code> value is not a valid directory record object ID.</td>
</tr>
<tr>
<td>MC_INVALID_ENTITY_ID</td>
<td>The <code>EntityID</code> value is not a valid DICOMDIR entity object ID.</td>
</tr>
<tr>
<td>MC_UNABLE_TO_GET_ITEM_ID</td>
<td>The <code>DirID</code> value does not contain a reference to a valid root item.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>The <code>Count</code> parameter was NULL.</td>
</tr>
<tr>
<td>MC_MSGFILE_ERROR</td>
<td>An error occurred attempting to access the configuration data file for the directory record.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>
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 the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_DICOMDIR_ID</td>
<td>The DirID value is not a valid DICOMDIR object ID.</td>
</tr>
<tr>
<td>MC_INVALID_ITEM_ID</td>
<td>The ItemID value is not a valid item object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>The NextEntityID, NextItemID, or NextItemName parameter was NULL.</td>
</tr>
<tr>
<td>MC_UNABLE_TO_GET_ITEM_ID</td>
<td>Merge DICOM Toolkit found a corrupted record ID in a directory record offset attribute while trying to traverse the DICOMDIR object pointed to by DirID.</td>
</tr>
</tbody>
</table>

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

```c
#include "mc3media.h"

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

<table>
<thead>
<tr>
<th>DirID</th>
<th>The identifier assigned to this DICOMDIR object by the MC_Create_File or MC_Create_Empty_File functions.</th>
</tr>
</thead>
<tbody>
<tr>
<td>EntityID</td>
<td>The identifier assigned to this directory entity object.</td>
</tr>
<tr>
<td>ItemID</td>
<td>Upon successful completion, the item object identifier of the directory record is returned here.</td>
</tr>
<tr>
<td>ItemName</td>
<td>Upon successful completion, a pointer to the string name of the item associated with the item object ItemID is returned here.</td>
</tr>
<tr>
<td>IsLast</td>
<td>Upon successful completion, this parameter is set to true (non-zero) if the retrieved directory record is the last record in EntityID.</td>
</tr>
</tbody>
</table>
**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 the enumerated **MC_STATUS** codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_DICOMDIR_ID</td>
<td>The <em>DirID</em> value is not a valid DICOMDIR object ID.</td>
</tr>
<tr>
<td>MC_INVALID_ITEM_ID</td>
<td>The <em>ItemID</em> value is not a valid item or record object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>The <em>ItemID</em>, <em>ItemName</em>, or <em>IsLast</em> parameter was NULL.</td>
</tr>
<tr>
<td>MC_UNABLE_TO_GET_ITEM_ID</td>
<td>Merge DICOM Toolkit found a corrupted record ID in a directory record offset attribute while trying to traverse the DICOMDIR object pointed to by <em>DirID</em>.</td>
</tr>
</tbody>
</table>

**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.
**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 Merge DICOM Toolkit. To remain DICOM conformant, the parameter `MrdrName` should always be the default data dictionary name for an MRDR item.

**Return Value**

One of the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MC_NORMAL_COMPLETION</code></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><code>MC_INVALID_DICOMDIR_ID</code></td>
<td>The <code>DirID</code> value is not a valid DICOMDIR object ID.</td>
</tr>
<tr>
<td><code>MC_INVALID_ITEM_ID</code></td>
<td>The <code>ItemID</code> value is not a valid item or record object ID.</td>
</tr>
<tr>
<td><code>MC_NULL_POINTER_PARM</code></td>
<td>The <code>MrdrID</code> or <code>MrdrName</code> parameter was NULL.</td>
</tr>
<tr>
<td><code>MC_UNABLE_TO_GET_ITEM_ID</code></td>
<td>Merge DICOM Toolkit found a corrupted record ID in a directory record offset attribute while trying to traverse the DICOMDIR object pointed to by <code>DirID</code>.</td>
</tr>
<tr>
<td><code>MC_SYSTEM_ERROR</code></td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**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**

```c
#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. Merge DICOM Toolkit 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 the enumerated **MC_STATUS** codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_INVALID_DICOMDIR_ID</strong></td>
<td>The <strong>DirID</strong> value is not a valid DICOMDIR object ID.</td>
</tr>
<tr>
<td><strong>MC_INVALID_ITEM_ID</strong></td>
<td>The <strong>ItemID</strong> value is not a valid item object ID.</td>
</tr>
<tr>
<td><strong>MC_INVALID_MRDR_ID</strong></td>
<td>The <strong>MrdrID</strong> value is not a valid MRDR object ID.</td>
</tr>
<tr>
<td><strong>MC_UNABLE_TO_GET_ITEM_ID</strong></td>
<td>While trying to traverse the DICOMDIR object pointed to by <strong>DirID</strong>, Merge DICOM Toolkit found an invalid item ID contained in a directory record offset attribute.</td>
</tr>
<tr>
<td><strong>MC_SYSTEM_ERROR</strong></td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**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**

```c
#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. Merge DICOM Toolkit decrements the use count attribute of the MRDR. If the use count equals zero, the MRDR will be automatically freed.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_DICOMDIR_ID</td>
<td>The DirID value is not a valid DICOMDIR object ID.</td>
</tr>
<tr>
<td>MC_INVALID_ITEM_ID</td>
<td>The ItemID value is not a valid item or record object ID.</td>
</tr>
<tr>
<td>MC_UNABLE_TO_GET_ITEM_ID</td>
<td>While trying to traverse the DICOMDIR object pointed to by DirID, Merge DICOM Toolkit found an invalid item ID contained in a directory record offset attribute.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**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 the enumerated MC_STATUS codes defined in "mcstatus.h":
<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_DICOMDIR_ID</td>
<td>The DirID value is not a valid DICOMDIR object ID.</td>
</tr>
<tr>
<td>MC_UNABLE_TO_GET_ITEM_ID</td>
<td>The DirID value does not contain a reference to a valid root item.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>The Count parameter was NULL.</td>
</tr>
<tr>
<td>MC_MSGFILE_ERROR</td>
<td>An error occurred attempting to access the configuration data file for the directory record.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**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**

```c
#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 the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_DICOMDIR_ID</td>
<td>The DirID value is not a valid DICOMDIR object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>The RootEntityID, ItemID, or ItemName parameter was NULL.</td>
</tr>
<tr>
<td>MC_UNABLE_TO_GET_ITEM_ID</td>
<td>While trying to traverse the DICOMDIR object pointed to by DirID, Merge DICOM Toolkit found an invalid item ID contained in a directory record offset attribute.</td>
</tr>
</tbody>
</table>

**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**

```c
#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 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 the enumerated `MC_STATUS` codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_LIBRARY_NOT_INITIALIZED</strong></td>
<td>The library has not been properly initialized.</td>
</tr>
<tr>
<td><strong>MC_INVALID_DICOMDIR_ID</strong></td>
<td><code>DirID</code> value is not a valid DICOMDIR object ID.</td>
</tr>
<tr>
<td><strong>MC_INVALID_LOWER_DIR_RECORD</strong></td>
<td>While trying to traverse the DICOMDIR object pointed to by <code>DirID</code>, Merge DICOM Toolkit found an invalid item ID contained in a lower-level directory entity attribute.</td>
</tr>
<tr>
<td><strong>MC_UNABLE_TO_GET_ITEM_ID</strong></td>
<td>While trying to traverse the DICOMDIR object pointed to by <code>DirID</code>, Merge DICOM Toolkit found an invalid item ID contained in a directory record offset attribute.</td>
</tr>
<tr>
<td><strong>MC_VALUE_OUT_OF_RANGE</strong></td>
<td>While trying to traverse the DICOMDIR object pointed to by <code>DirID</code>, Merge DICOM Toolkit found an item ID that contained an attribute with an invalid value.</td>
</tr>
<tr>
<td><strong>MC_SYSTEM_ERROR</strong></td>
<td>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.</td>
</tr>
</tbody>
</table>

**MC_Dir_Stream_Directory_Record**

Stores a DICOMDIR directory record internally in stream format to reduce memory usage.

**Synopsis**

```c
#include "mc3media.h"

MC_STATUS MC_Dir_Stream_Directory_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 item object identifier of the directory record whose storage method will be changed.

---

**Remarks**

**MC_Dir_Stream_Directory_Record** changes the internal method for storing a directory record. When called, Merge DICOM Toolkit changes the storage method for the specified directory record to a DICOM byte stream instead of Merge DICOM Toolkit’s internal structure. Storing as a DICOM byte stream reduces the memory required by Merge DICOM Toolkit. This can be especially useful when creating and working with large DICOMDIRs.

When a directory record is stored as a DICOM byte stream, Merge DICOM Toolkit will automatically change it back to its internal format if a tag is accessed within the directory record.

When reading a DICOMDIR from disk, the **DICOMDIR_STREAM_STORAGE** configuration option can also be used to change how directory records are stored. When enabled, this option will force all directory records in a DICOMDIR to be stored as a DICOM byte stream.

---

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_INVALID_DICOMDIR_ID</strong></td>
<td>The DirID value is not a valid DICOMDIR object ID.</td>
</tr>
<tr>
<td><strong>MC_INVALID_ITEM_ID</strong></td>
<td>The ItemID value is not a valid item or record object ID.</td>
</tr>
<tr>
<td><strong>MC_SYSTEM_ERROR</strong></td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

---

**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. To use the built in RLE compressor, set this to MC_RLE_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. To use the built in RLE decompressor, set this to MC_RLE_Decompressor.

The functions must be prototyped as follows:

```c
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.
CBoutdataLengt
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
ENCAPSULATED_UNCOMPRESSED_ELE
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.

Registering MC_RLE_Compressor and/or MC_RLE_Decompressor will use Merge DICOM Toolkit’s build in RLE compressor and decompressor.
NOTE: The built in compressor/decompressor
MC_Standard_Compressor/MC_Standard_Decompressor is only available on platforms supported by the Pegasus libraries from Accusoft (Windows, Solaris, and Linux). The build in compressor/decompressor MC_RLE_Compressor/MC_RLE_Decompressor is supported on all Merge DICOM Toolkit platforms.

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

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 YourDesMsg(De)CompressionCallback is called. This provides a clear mechanism for the function to know it is being called the last time.

YourDesMsg(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 defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_COMPRESSOR_REQUIRED</td>
<td>A compression function must be passed in for the destination message.</td>
</tr>
<tr>
<td>MC_DECOMPRESSOR_REQUIRED</td>
<td>A decompression function must be registered with the source message.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>SourceMsgID is not a valid message object ID.</td>
</tr>
<tr>
<td>MC_CALLBACK_CANNOT_COMPLY</td>
<td>YourDesMsg(De)CompressionCallback returned with MC_CANNOT_COMPLY, or the callback function registered for this attribute, and providing data, returned MC_CANNOT_COMPLY.</td>
</tr>
</tbody>
</table>
**See Also**

MC_Free_Message  
MC_Register_Compression_Callbacks

---

**MC_Empty_File**

Sets all the attributes in a DICOM file to empty

**Synopsis**

```c
#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 ensure 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 the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_FILE_ID</td>
<td>The <em>FileID</em> value is not a valid file object ID.</td>
</tr>
</tbody>
</table>

**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

```c
#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 ensure 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 defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_ITEM_ID</td>
<td>The ItemID value is not a valid item object ID.</td>
</tr>
</tbody>
</table>

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

```c
#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 ensure 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 defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>The <code>MessageID</code> value is not a valid message object ID.</td>
</tr>
</tbody>
</table>

**See Also**

MC_Empty_Item  
MC_Set_Value_To_Empty  
MC_Set_pValue_To_Empty

---

**MC_Enable_Unicode_Conversion**

Enable/disable Unicode conversion functions.

**Synopsis**

```c
#include “mc3msg.h”

MC_STATUS MC_Enable_Unicode_Conversion ( 
    int Enable
)
```

**Enable**
Set to non-zero to load the Unicode conversion library and enable its conversion functions. Set to zero to unload the Unicode conversion library and disable its conversion functions.

**Remarks**

**MC_Enable_Unicode_Conversion** set to non-zero initializes the Unicode conversion library by loading the Unicode shared objects and its dependency files as specified in Platform Notes. When success, all Unicode conversion functions are available for use. When sets to zero, the function
unloads the Unicode conversion library shared objects, its dependency files and disable all Unicode conversion functions.

**NOTE:** If enabling Unicode conversion fails, check the location of shared objects or dependency files, or loading path as specified in the Platform Notes.

### Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_CANNOT_COMPLY</td>
<td>The Unicode library fails to initialize or load.</td>
</tr>
</tbody>
</table>

### See Also

- `MC_Byte_To_Unicode`
- `MC_Unicode_To_Byte`
- `MC_Get_Value_To_UnicodeString`
- `MC_Get_Next_Value_To_UnicodeString`
- `MC_Set_Value_From_UnicodeString`
- `MC_Set_Next_Value_From_UnicodeString`

### MC_Error_Message

Returns a string description of a Merge DICOM Toolkit status code.

#### Synopsis

```c
#include "mc3msg.h"

char *MC_Error_Message ( MC_STATUS StatusCode )

StatusCode A Merge DICOM Toolkit 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

```c
#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 Merge DICOM Toolkit 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 the enumerated MC_STATUS codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_FILE_ID</td>
<td>The <em>FileID</em> value is not a valid file object ID.</td>
</tr>
</tbody>
</table>

### See Also

MC_Message_To_File

## MC_Free_Item

Releases a Merge DICOM Toolkit item object.

### Synopsis

```c
#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 defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_ITEM_ID</td>
<td>The *ItemID value is not a valid item object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>*ItemID was NULL.</td>
</tr>
</tbody>
</table>

**See Also**

MC_Free_File
MC_Free_Message
MC_Open_Item

**MC_Free_File**

Releases a Merge DICOM Toolkit 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 the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MC_NORMAL_COMPLETION</code></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><code>MC_INVALID_FILE_ID</code></td>
<td>The <code>*FileID</code> value is not a valid file object ID.</td>
</tr>
<tr>
<td><code>MC_NULL_POINTER_PARM</code></td>
<td><code>FileID</code> was NULL.</td>
</tr>
</tbody>
</table>

**See Also**

`MC_Free_Item`

`MC_Free_Message`

---

### MC_Free_Message

Releases a Merge DICOM Toolkit message object.

**Synopsis**

```c
#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 defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>The <code>MessageID</code> value is not a valid message object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td><code>MessageID</code> was NULL.</td>
</tr>
</tbody>
</table>

**See Also**

MC_Open_Message
MC_Free_Item
MC_Free_File

**MC_FreeService**

Releases a Merge DICOM Toolkit service that was dynamically created using MC_NewService.

**Synopsis**

```c
#include "mergecom.h"

MC_STATUS MC_FreeService (char *ServiceName)
```

Where `ServiceName` is 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 defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_SERVICE_IN_USE</td>
<td>The service is in use in a service list</td>
</tr>
<tr>
<td>MC_INVALID_SERVICE_NAME</td>
<td>The service name does not represent a name currently defined.</td>
</tr>
</tbody>
</table>
See Also

MC_NewService
MC_NewProposedServiceList
MC_FreeServiceList

MC_FreeServiceList

Releases a Merge DICOM Toolkit service list.

Synopsis

```c
#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 defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SERVICE_LIST_NAME</td>
<td>The service list name does not represent a name currently defined.</td>
</tr>
</tbody>
</table>

See Also

MC_NewService
MC_NewProposedServiceList
MC_FreeService
**MC_FreeSyntaxList**

Releases a Merge DICOM Toolkit message object.

**Synopsis**

```c
#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 defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_INVALID_SYNTAX_LIST_NAME</strong></td>
<td>The syntax list name does not represent a name currently defined.</td>
</tr>
<tr>
<td><strong>MC_SYNTAX_IN_USE</strong></td>
<td>The syntax list is in use by a service definition.</td>
</tr>
</tbody>
</table>

**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**

```c
#include "mc3msg.h"
MC_STATUS MC_Get_Association_Info (  
    int AssociationID,  
    AssocInfo *AssociationInfo
)
```
**AssociationID** The identifier assigned to this object by the
`MC_Open_Association`, `MC_Open_Secure_Association`,
`MC_Process_Association_Request`,
`MC_Process_Secure_Association_Request`,
`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:

```c
typedef enum {
    NO_USER_IDENTITY = 0,
    USERNAME = 1,
    USERNAME_AND_PASSCODE = 2,
    KERBEROS_SERVICE_TICKET = 3,
    SAML_ASSERTION = 4
} USER_IDENTITY_TYPE;

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 */
    MC_SOCKET 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 */
    USER_IDENTITY_TYPE UserIdentityType; /* User Identity negotiation
                                             type as defined in DICOM
                                             Supplement 99 */
    unsigned char PositiveResponseRequested; /* Set according to if a
                                                  positive response is
                                                  requested when User
                                                  Identity is specified */
    unsigned char PositiveResponseReceived; /* Set for association
                                             requestors if a positive
                                             response was received from
                                             the association acceptor */
} 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_Association_With_Identity, MC_Open_Secure_Association, MC_Process_Association_Request, MC_Process_Secure_Association_Request, MC_Wait_For_Association or MC_Wait_For_Secure_Association call, and before any MC_Read_Message, MC_Send_Request_Message, MC_Send_Request, MC_Send_Response_Message or MC_Send_Response calls.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_ASSOC_ID</td>
<td>AssociationID is not a valid association object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>The AssociationInfo pointer had a null value.</td>
</tr>
<tr>
<td>MC_STATE_VIOLATION</td>
<td>The call was made after messages have been exchanged over the association.</td>
</tr>
<tr>
<td>MC_LIBRARY_NOT_INITIALIZED</td>
<td>The library has not been properly initialized.</td>
</tr>
</tbody>
</table>

**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**

```c
#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:

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 the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>MsgFileItemID is not a valid message object ID, file object ID or item object ID.</td>
</tr>
<tr>
<td>MC_INVALID_TAG</td>
<td>The message does not contain an attribute with an ID of Tag.</td>
</tr>
</tbody>
</table>

See Also
MC_Get_First_Attribute MC_Get_Tags_Dict_Info
MC_Get_Next_Attribute MC_Get_pAttribute_Info

MC_Get_Bool_Config_Value
Used to get the value of a boolean toolkit 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:
ACCEPT_ANY_APPLICATION_TITLE
ACCEPT_ANYCONTEXT_NAME
ACCEPT_ANY_HOSTNAME
ACCEPT_ANY_PRESENTATION_CONTEXT
ACCEPT_DIFFERENT_IC_UID
ACCEPT_DIFFERENT_VERSION
ACCEPT_MULTIPLE_PRES_CONTEXTS
ACCEPTRELATED_GENERAL_SERVICES
ACCEPT_STORAGE_SERVICE_CONTEXTS
ALLOW_COMMA_IN_DS_FL_FD_STRINGS
ALLOW_EMPTY_PDV_LENGTH
ALLOW_INVALID_PRIVATE_ATTRIBUTES
ALLOW_INVALID_PRIVATE_CREATOR_CODES
ALLOW_LIBRARY_EXCEPTION_HANDLER
ALLOW_OUT_OF_RANGE_BITS_JPEG LOSSLESS
ATT_00081190_USE UT VR
ATT_00287FE0_USE UT VR
ATT_0040E010_USE_UT VR
ATT_0074100A_USE_ST VR
AUTO_ECHO_SUPPORT
BLANK_FILL_LOG_FILE
CALCULATE_DEFINED_LENGTH_FOR_CB
COMPRESSION_ALLOW_FRAGS
COMPRESSION_J2K_LOSSY_USE_QUALITY
COMPRESSION_USE_HEADER_QUERY
COMPRESSION_WHEN_J2K_USE_LOSSY
CREATE_OFFSET_TABLE
DEFLATE_ALLOW_FLUSH
DICOMDIR_STREAM_STORAGE
DUPLICATE_ENCAPSULATED_ICON
ELIMINATE_ITEM_REFERENCES
EMPTY_PRIVATE_CREATOR_CODES
EXPLICIT VR_TO_UN FOR LENGTH_GT 64K
EXPORTEMPTY_PRIVATE_CREATOR_CODES
EXPORTGROUP_LENGTHSTOMEDIA
EXPORTGROUP_LENGTHSTONETWORK
EXPORTPRIVATE_ATTRIBUTES_TO_MEDIA
EXPORTPRIVATE_ATTRIBUTES_TO_NETWORK
EXPORTUNVR_TO_MEDIA
EXPORTUNVR_TO_NETWORK
EXPORTUNDEFINED_LENGTH_SQ
EXPORTUNDEFINED_LENGTH_SQ_INDICOMDIR
FORCEOPENEMPTYITEM
HARD_CLOSE_TCP_IP_CONNECTION
INSURE_EVEN_UID_LENGTH
LIST_UNATTRIBUTES
LOG_FILE_BACKUP
MSG_FILEITEM_OBJ_TRACE
NETWORK_CAPTURE
PRIVATE_SYNTAX_1_ENCAPSULATED
PRIVATE_SYNTAX_1_EXPLICIT VR
PRIVATE_SYNTAX_1_LITTLE_ENDIAN
PRIVATE_SYNTAX_2_ENCAPSULATED
PRIVATE_SYNTAX_2_EXPLICIT VR
PRIVATE_SYNTAX_2_LITTLE_ENDIAN
REJECT_INVALID VR
RELEASE_SQITEMS
REMOVE_PADDING_CHARS
Value The boolean value to which Parm is set is returned here.

Remarks
The Merge DICOM Toolkit 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 toolkit configuration elsewhere in this manual.

Return Value
One of the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>Value was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_PARAMETER_NAME</td>
<td>Parameter specified is not one of the above listed parameters.</td>
</tr>
<tr>
<td>MC_LIBRARY_NOT_INITIALIZED</td>
<td>The library has not been properly initialized.</td>
</tr>
</tbody>
</table>

See Also
MC_Get_Int_Config_Value 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_Set_Bool_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,
```

© Copyright Merge Healthcare Solutions Inc. 2023

94
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
accompany this, one passes the **MC_Get_Encapsulated_Value_To_Function** function the name of a function (*YourGetFunction*) which Merge DICOM Toolkit, 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_Encapsulated_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 the enumerated **MC_STATUS** codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally, and there are no more frames in the message.</td>
</tr>
<tr>
<td><strong>MC_INVALID_TRANSFER_SYNTAX</strong></td>
<td>The message’s transfer syntax is non-encapsulated.</td>
</tr>
<tr>
<td><strong>MC_END_OF_FRAME</strong></td>
<td>The function completed normally, and there are more frames in the message.</td>
</tr>
<tr>
<td><strong>MC_NULL_POINTER_PARM</strong></td>
<td><em>YourGetFunction</em> is NULL</td>
</tr>
<tr>
<td><strong>MC_INVALID_MESSAGE_ID</strong></td>
<td>CBMsgFileItemID is not a valid message object ID, file object ID or item object ID.</td>
</tr>
<tr>
<td><strong>MC_INVALID_TAG</strong></td>
<td>The message does not contain an attribute with an ID of Tag.</td>
</tr>
<tr>
<td><strong>MC_INCOMPATIBLE_VR</strong></td>
<td>The attribute does not contain OB or OW data</td>
</tr>
<tr>
<td><strong>MC_EMPTY_VALUE</strong></td>
<td>No value has been assigned to this attribute yet.</td>
</tr>
<tr>
<td><strong>MC_NULL_VALUE</strong></td>
<td>The attribute has a value of NULL.</td>
</tr>
<tr>
<td><strong>MC_NO_MORE_VALUES</strong></td>
<td>The attribute has no more values.</td>
</tr>
<tr>
<td><strong>MC_CALLBACK_CANNOT_COMPLY</strong></td>
<td><em>YourGetFunction</em> returned MC_CANNOT_COMPLY.</td>
</tr>
<tr>
<td><strong>MC_COMPRESSION_FAILURE</strong></td>
<td>There was an error while trying to decompress the data.</td>
</tr>
</tbody>
</table>
MC_MISSING_DELIMITER
The expected delimiter can not be detected

See Also
MC_Get_Next_Encapsulated_Value_To_Function
MC_Get_Frame_To_Function
MC_Get_Offset_Table_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
ENCAPSULATED_UNCOMPRESSED_ELE
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
HEVC_H265_M10P_LEVEL_5_1
HEVC_H265_MP_LEVEL_5_1
JPIP_REFERENCED
JPIP_REFERENCED_DEFLATE
MPEG2_MPML
MPEG2_MPHL
MPEG4_AVC_H264_HP_LEVEL_4_1
MPEG4_AVC_H264_BDC_HP_LEVEL_4_1
MPEG4_AVC_H264_HP_LEVEL_4_2_2D
MPEG4_AVC_H264_HP_LEVEL_4_2_3D
MPEG4_AVC_H264_STEREO_HP_LEVEL_4_2
JPEG_2000_MC_LOSSLESS_ONLY
JPEG_2000_MC
SMPT_E2110_20_UNCOMPRESSED_PROGRESSIVE_ACTIVE_VIDEO
SMPT_E2110_20_UNCOMPRESSED_INTERLACED_ACTIVE_VIDEO
MC_Get_Enum_From_Transfer_Syntax converts a DICOM transfer syntax UID into a Merge DICOM Toolkit enumerated value. The UID values used by MC_Get_Enum_From_Transfer_Syntax are specified in Merge DICOM Toolkit’s configuration files. Please see the detailed description of toolkit configuration elsewhere in this manual.

**Remarks**

MC_Get_File_Length gets the length in bytes of a file object.

**Synopsis**

```c
#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 the enumerated MC_STATUS codes defined in “mcstatus.h”:

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

MC_Get_File_Preamble

Gets the preamble for a file object.

Synopsis

```c
#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 the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_FILE_ID</td>
<td>The FileID value is not a valid file object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>Preamble was NULL.</td>
</tr>
</tbody>
</table>

See Also

MC_Set_File_Preamble
**MC_Get_Filename**

Gets the filename associated with a file object.

**Synopsis**

```c
#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 the enumerated **MC_STATUS** codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_FILE_ID</td>
<td>The <strong>FileID</strong> value is not a valid file object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td><strong>Filename</strong> was NULL.</td>
</tr>
<tr>
<td>MC_BUFFER_TOO_SMALL</td>
<td><strong>Filename</strong> was too small to contain the complete filename.</td>
</tr>
</tbody>
</table>

**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**

```c
#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_Process_Association_Request`
- `MC_Process_Secure_Association_Request`
- `MC_Wait_For_Association`
- `MC_Wait_For_Secure_Association`

**AserviceInfo**

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

```c
typedef struct MC_Service_Info {
    char ServiceName[50];       /* Merge DICOM Toolkit service name */
    TRANSFER_SYNTAX SyntaxType; /* transfer syntax negotiated for the service */
    ROLE_TYPE RoleNegotiated;   /* The role negotiated for the service */
    int PresentationContextID;  /* Presentation Context ID */
    char SOPClassUID[66];       /* SOP Class UID associated with ServiceName */
    char ServiceClassUID[66];   /* Optional Service Class UID for the service if defined in the association request */
    int NumberRelatedSOPClasses; /* Number of related general SOP Classes contained in the association request */
    RelatedSOPClass RelatedSOPClasses[10]; /* Array containing Related general SOP classes for this negotiated service */
} ServiceInfo;
```

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

**Remarks**

The Merge DICOM Toolkit 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 the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

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

Retrieves information about the first attribute in a message.

**Synopsis**

```c
#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, UC, UR,*

*UT, UI, SS, US, AT, SL, UL, SV, UV, FL, FD, OB, OW, OV, OL,*

*OD, 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 the enumerated **MC_STATUS** codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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_Frame_To_Function`

**MC_Get_Frame_To_Function**

Retrieves the requested frame of encapsulated or non-encapsulated pixel data from a message. For encapsulated data, if a decompressor is registered, then the data will be returned decompressed.

**Synopsis**

```c
#include "mc3msg.h"

MC_STATUS MC_Get_Frame_To_Function ( 
  int  MsgFileItemID,  
  int  Frame,  
  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.

*Frame*  
An integer identifying a frame number of encapsulated pixel data (starting from 0).

*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:

```c
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_Frame_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_Frame_To_Function* function is used to fetch an encapsulated or non-encapsulated frame of a pixel data attribute. In the encapsulated case, the functionality is similar to 
*MC_Get_Encapsulated_Value_To_Function* and *MC_Get_Next_Encapsulated_Value_To_Function* except that the user can retrieve the encapsulated frame by its number (starting from 0).

As pixel data attributes usually have values of great length the most effective way to handle it is to use 
*MC_Open_File_Upto_Tag_Bypass_Value* and the callback mechanism with 
*MC_Register_Callback_Function* to retrieve the attribute values from media.

The (*YourGetFunction*) function is used to provide repeatedly the 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_Frame_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*.

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

*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 the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally, and there are no more frames in the message.</td>
</tr>
<tr>
<td>MC_INVALID_TRANSFER_SYNTAX</td>
<td>The message’s transfer syntax is non-encapsulated.</td>
</tr>
<tr>
<td>MC_END_OF_FRAME</td>
<td>The function completed normally, and there are more frames in the message.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>YourGetFunction is NULL</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>MsgFileItemID is not a valid message object ID, file object ID or item object ID.</td>
</tr>
<tr>
<td>MC_INVALID_TAG</td>
<td>The message does not contain an attribute with an ID of Tag.</td>
</tr>
<tr>
<td>MC_INCOMPATIBLE_VR</td>
<td>The attribute does not contain OB or OW data</td>
</tr>
<tr>
<td>MC_EMPTY_VALUE</td>
<td>No value has been assigned to this attribute yet.</td>
</tr>
<tr>
<td>MC_NULL_VALUE</td>
<td>The attribute has a value of NULL.</td>
</tr>
<tr>
<td>MC_NO_MORE_VALUES</td>
<td>The attribute has no more values.</td>
</tr>
<tr>
<td>MC_CALLBACK_CANNOT_COMPLY</td>
<td>YourGetFunction returned MC_CANNOT_COMPLY.</td>
</tr>
<tr>
<td>MC_COMPRESSION_FAILURE</td>
<td>There was an error while trying to decompress the data.</td>
</tr>
<tr>
<td>MC_MISSING_DELIMITER</td>
<td>The expected delimiter cannot be detected</td>
</tr>
</tbody>
</table>

**See Also**

- MC_Get_Encapsulated_Value_To_Function
- MC_Get_Next_Encapsulated_Value_To_Function
- MC_Open_File_Upto_Tag_Bypass_Value
- MC_Register_Compression_Callbacks
- MC_Get_Int_Config_Value

**MC_Get_Int_Config_Value**

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

**Synopsis**

```c
#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:

- ARTIM_TIMEOUT
ASSOC_REPLY_TIMEOUT  
COMPRESSION_CHROM_FACTOR  
COMPRESSION_J2K_LOSSY_QUALITY  
COMPRESSION_J2K_LOSSY_RATIO  
COMPRESSION_LUM_FACTOR  
CONNECT_TIMEOUT  
DEFLATE_COMPRESSION_LEVEL  
DESIRED_LAST_PDU_SIZE  
FLATE_GROW_OUTPUT_BUF_SIZE  
IGNORE_JPEG_BAD_SUFFIX  
INACTIVITY_TIMEOUT  
LARGE_DATA_SIZE  
LIST_SQ_DEPTH_LIMIT  
LIST_VALUE_LIMIT  
LOG_FILE_LINE_LENGTH  
LOG_FILE_SIZE  
LOG_MEMORY_SIZE  
MAX_PENDING_CONNECTIONS  
NUM_HISTORICAL_LOG_FILES  
NUMBER_OF_CAP_FILES  
OBOW_BUFFER_SIZE  
PEGASUS_NUMBER_OF_THREADS  
RELEASE_TIMEOUT  
TCP/IP_KEEP_ALIVE_TIME  
TCP/IP_KEEP_ALIVE_INTERVAL  
TCP/IP_LISTEN_PORT  
TCP/IP_RECEIVE_BUFFER_SIZE  
TCP/IP_SEND_BUFFER_SIZE  
WORK_BUFFER_SIZE  
WRITE_TIMEOUT

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

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

**Remarks**

The Merge DICOM Toolkit 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 toolkit configuration elsewhere in this manual.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_NULL_POINTER_PARM</strong></td>
<td><em>Value was NULL.</em></td>
</tr>
<tr>
<td><strong>MC_INVALID_PARAMETER_NAME</strong></td>
<td>Parameter specified is not one of the above listed parameters.</td>
</tr>
<tr>
<td><strong>MC_MUST_BE_POSITIVE</strong></td>
<td>Parameter value cannot be negative.</td>
</tr>
</tbody>
</table>

© Copyright Merge Healthcare Solutions Inc. 2023
MC_LIBRARY_NOT_INITIALIZED

The library has not been properly initialized.

See Also

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_Set_Int_Config_Value</td>
<td></td>
</tr>
<tr>
<td>MC_Get_Log_Destination</td>
<td></td>
</tr>
<tr>
<td>MC_Get_Long_Config_Value</td>
<td></td>
</tr>
<tr>
<td>MC_Get_String_Config_Value</td>
<td></td>
</tr>
</tbody>
</table>

MC_Get_Listen_Socket

Returns the id of the socket being listened on

Synopsis

```c
#include "mergecom.h"

MC_STATUS MC_Get_Listen_Socket ( 
    MC_SOCKET *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 the enumerated **MC_STATUS** codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td><em>AlistenSocket</em> was NULL.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment.</td>
</tr>
</tbody>
</table>
**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**

```c
#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 the enumerated **MC_STATUS** codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td><code>ListenSocket</code> was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_PORT_NUMBER</td>
<td><code>Port</code> is not currently being listened on.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment.</td>
</tr>
</tbody>
</table>
See Also
MC_Get_Association_Info

MC_Get_Log_Destination

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

Synopsis

```c
#include "mc3msg.h"
MC_STATUS MC_Get_Log_Destination (  
    LogParm Parm,  
    int *Destination
)
```

Parm  An enumerated constant identifying the class of logging that is to be retrieved. 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

Destination  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 Merge DICOM Toolkit initialization file contains the setting used at startup. Please see the detailed description of toolkit configuration elsewhere in this manual.

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
</table>

© Copyright Merge Healthcare Solutions Inc. 2023
**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_Set_Int_Config_Value
- MC_Get_Int_Config_Value
- MC_Set_Long_Config_Value
- MC_Get_Long_Config_Value
- MC_Set_String_Config_Value
- MC_Get_String_Config_Value
- MC_Set_Log_Destination

---

**MC_Get_Long_Config_Value**

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

### Synopsis

```c
#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:
- CAPTURE_FILE_SIZE
- PDU_MAXIMUM_LENGTH
- CALLBACK_MIN_DATA_SIZE
- PIXELBUFFER_SIZE

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

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

### Remarks

The Merge DICOM Toolkit 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 toolkit configuration elsewhere in this manual.

### Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_NULL_POINTER_PARM</strong></td>
<td>Value was NULL.</td>
</tr>
<tr>
<td><strong>MC_INVALID_PARAMETER_NAME</strong></td>
<td>Parameter specified is not one of the above listed parameters.</td>
</tr>
<tr>
<td><strong>MC_LIBRARY_NOT_INITIALIZED</strong></td>
<td>The library has not been properly initialized.</td>
</tr>
</tbody>
</table>
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 Merge DICOM Toolkit service name associated with a DICOM SOP Class UID.

Synopsis

```c
#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 Merge DICOM Toolkit service name.

**ServiceName**
The name of the Merge DICOM Toolkit 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 the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td><code>UID</code> or <code>ServiceName</code> was NULL.</td>
</tr>
<tr>
<td>MC_BUFFER_TOO_SMALL</td>
<td><code>ServiceName</code> was not large enough to contain the Merge DICOM Toolkit service name.</td>
</tr>
<tr>
<td>MC_INVALID_SOP_CLASS_UID</td>
<td><code>UID</code> does not contain a Merge DICOM Toolkit supported SOP Class UID.</td>
</tr>
</tbody>
</table>
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_MergeCOM_Service_From_UID

Converts a DICOM SOP Class UID into a Merge DICOM Toolkit service name.

Synopsis

```c
#include "mc3msg.h"

MC_STATUS MC_Get_MergeCOM_Service_From_UID(
    char *Uid,
    char *ServiceName,
    int ServiceNameLength
)
```

*Uid* The service's SOP Class UID is returned here.

*ServiceName* String name of a DICOM service

*ServiceNameLength* The length in bytes of *ServiceName* buffer

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_UNKNOWN_SERVICE</td>
<td><em>ServiceName</em> was not registered in the Merge DICOM Toolkit configuration files.</td>
</tr>
<tr>
<td>MC_BUFFER_TOO_SMALL</td>
<td><em>Uid</em> is not large enough to contain the unique identifier.</td>
</tr>
</tbody>
</table>

See Also

MC_Get_MergeCOM_Service
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

```c
#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 Merge DICOM Toolkit:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C_STORE_RQ</td>
<td>DICOM Composite Store Service Request</td>
</tr>
<tr>
<td>C_STORE_RSP</td>
<td>DICOM Composite Store Service Response</td>
</tr>
<tr>
<td>C_ECHO_RQ</td>
<td>DICOM Verification Service Request</td>
</tr>
<tr>
<td>C_ECHO_RSP</td>
<td>DICOM Verification Service Response</td>
</tr>
<tr>
<td>C_FIND_RQ</td>
<td>DICOM Composite Find Service Request</td>
</tr>
<tr>
<td>C_FIND_RSP</td>
<td>DICOM Composite Find Service Response</td>
</tr>
<tr>
<td>C_CANCEL_FIND_RQ</td>
<td>Cancel DICOM Composite Find Service Request</td>
</tr>
<tr>
<td>C_GET_RQ</td>
<td>DICOM Composite Get Service Request</td>
</tr>
<tr>
<td>C_GET_RSP</td>
<td>DICOM Composite Get Service Response</td>
</tr>
<tr>
<td>C_CANCEL_GET_RQ</td>
<td>Cancel DICOM Composite Get Service Request</td>
</tr>
<tr>
<td>C_MOVE_RQ</td>
<td>DICOM Composite Move Service Request</td>
</tr>
<tr>
<td>C_MOVE_RSP</td>
<td>DICOM Composite Move Service Response</td>
</tr>
<tr>
<td>C_CANCEL_MOVE_RQ</td>
<td>Cancel DICOM Composite Move Service Request</td>
</tr>
<tr>
<td>N_EVENT_REPORT_RQ</td>
<td>DICOM Normalized Report Service Request</td>
</tr>
<tr>
<td>N_EVENT_REPORT_RSP</td>
<td>DICOM Normalized Report Service Response</td>
</tr>
<tr>
<td>N_GET_RQ</td>
<td>DICOM Normalized Get Service Request</td>
</tr>
</tbody>
</table>
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 the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>ServiceName or Command was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>MessageID is not a valid message object ID.</td>
</tr>
</tbody>
</table>

**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**

```c
#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. It will be one of the enumerated TRANSFER_SYNTAX types defined in “mc3msg.h” (see MC_Get_Transfer_Syntax_From_Enum).
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 Merge DICOM Toolkit User's Manual for a discussion on negotiating multiple transfer syntaxes for a service.

`MC_Get_Message_Transfer_Syntax` can also be used to get the transfer syntax of a DICOM file object. When used with a DICOM file object, the routine will return the transfer syntax set for the tag (0002,0010) Transfer Syntax UID in the file object.

Return Value

One of the enumerated `MC_STATUS` codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>TransferSyntax was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>MessageID is not a valid message object ID.</td>
</tr>
<tr>
<td>MC_INVALID_TRANSFER_SYNTAX</td>
<td>The transfer syntax of the DICOM file or message is not set.</td>
</tr>
</tbody>
</table>

See Also

`MC_Set_Message_Transfer_Syntax`  `MC_Read_Message`  `MC_Reset_Message_Transfer_Syntax`  

`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`,
- `MC_Process_Association_Request`,
- `MC_Process_Secure_Association_Request` 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 or MC_Send_Request_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 or MC_Read_To_Stream 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 the enumerated MC_STATUS codes defined in “mcstatus.h”:

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

See Also

MC_Read_Message
MC_Read_To_Stream
MC_Send_Request_Message_For_Service
MC_Send_Request_For_Service

MC_Get_Negotiation_Info

Retrieves received extended negotiation information.

Synopsis

```c
#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,
MC_Process_Association_Request,
MC_Process_Secure_Association_Request 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. Merge DICOM Toolkit 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
deprecated.

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

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_ASSOC_ID</td>
<td>AssociationID is not a valid association identifier.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>Either ServiceName or ExtInfoBuffer or ExtInfoLength was NULL.</td>
</tr>
</tbody>
</table>
MC_UNKNOWN_SERVICE  
*ServiceName* was not registered in the Merge DICOM Toolkit 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**

```c
#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_Process_Association_Request,**  
**MC_Process_Secure_Association_Request,**  
**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:

```c
typedef struct MC_Service_Info {  
    char ServiceName[50]; /* Merge DICOM Toolkit service name */  
    TRANSFER_SYNTAX SyntaxType; /* transfer syntax negotiated for the service */  
    ROLE_TYPE RoleNegotiated; /* The role negotiated for the service */  
    int PresentationContextID; /* Presentation Context ID */  
    char SOPClassUID[66]; /* SOP Class UID associated with ServiceName */  
    char ServiceClassUID[66]; /* Optional Service Class UID for the service if defined in association request */  
    int NumberRelatedSOPClasses; /* Number of related general SOP Classes contained in the association request */  
    RelatedSOPClass RelatedSOPClasses[10]; /* Array containing Related general SOP classes for this negotiated service */  
} ServiceInfo;
```
"TRANSFER_SYNTAX" and "ROLE_TYPE" are defined in mc3msg.h.

**Remarks**

The Merge DICOM Toolkit 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 the enumerated **MC_STATUS** codes defined in "mcstatus.h":

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

**See Also**

MC_Open_Association, MC_Open_Secure_Association, MC_Wait_For_Association, MC_Wait_For_Secure_Association, MC_Get_First_Acceptable_Service, MC_Process_Association_Request, MC_Process_Secure_Association_Request

**MC_Get_Next_Attribute**

Retrieves information about the next attribute in a message.

**Synopsis**

```c
#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, UC, UR,
UT, UI, SS, US, AT, SL, UL, SV, UV, FL, FD, OB, OW, OV, OL,
OD, 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 the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>MsgFileItemID is not a valid message object ID, file object ID or item object ID.</td>
</tr>
<tr>
<td>MC_MESSAGE_EMPTY</td>
<td>The message contains no attributes.</td>
</tr>
<tr>
<td>MC_NO_MORE_ATTRIBUTES</td>
<td>There are no more attributes in the message.</td>
</tr>
</tbody>
</table>

See Also
MC_Get_First_Attribute  
MC_Get_pAttribute_Info  
MC_Get_Attribute_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
*M pièce_Open_Message, MC_Open_Empty_Message,*
*M_create_File, MC_Create_Empty_File,* or
*M pièce_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:

```c
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

**CBtag**  DICOM tag which identifies the attribute.

**CBuserInfo**  Address of user data which is being passed from the
*M pièce_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 Merge DICOM Toolkit, 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_Encapsulated_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 the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

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

MC_Get_Encapsulated_Value_To_Function
MC_Get_Frame_To_Function
MC_Get_Offset_Table_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,  
    MC_size_t BufferSize,  
    void *Value,  
    int *ValueSize  
)

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_Double (  
    int MsgFileItemID,  
    char *PrivateCode,  
    unsigned short Group,  
    unsigned char ElementByte,  
    double *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_UShortInt (  
    int MsgFileItemID,  
    char *PrivateCode,  
    unsigned short Group,  
    unsigned char ElementByte,  
    unsigned short *Value  
)

MC_STATUS MC_Get_Next_pValue_To_Int (  
    int MsgFileItemID,  
    char *PrivateCode,  
    unsigned short Group,  
    unsigned char ElementByte,  
    int *Value  
)
int MsgFileItemID,
char *PrivateCode,
unsigned short Group,
unsigned char ElementByte,
int *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_LongInt (  
  int MsgFileItemID,
  char *PrivateCode,
  unsigned short Group,
  unsigned char ElementByte,
  long int *Value
)

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

MC_STATUS MC_Get_Next_pValue_To_LongLong (  
  int MsgFileItemID,
  char *PrivateCode,
  unsigned short Group,
  unsigned char ElementByte,
  long long *Value
)

MC_STATUS MC_Get_Next_pValue_To_ULongLong (  
  int MsgFileItemID,
  char *PrivateCode,
  unsigned short Group,
  unsigned char ElementByte,
  unsigned long long *Value
)

MC_STATUS MC_Get_Next_pValue_To_String (  
  int MsgFileItemID,
  char *PrivateCode,
  unsigned short Group,
  unsigned char ElementByte,
  MC_size_t BufferSize,
  char *Value
)

MC_STATUS MC_Get_Next_pValue_To_UnicodeString (  
  int MsgFileItemID,
  char *PrivateCode,
  unsigned short Group,
  unsigned char ElementByte,
  MC_size_t BufferSize,
int *ValueSize,
MC_Uhar *Value)

(MsgFileItem) 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
LongLong_Type        Binary 64-bit integer number
UlongLong_Type       Binary 64-bit unsigned integer
number Float_Type    Binary Floating point number
Double_Type          Binary Floating point number
Buffer_Type          Binary byte value

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. This parameter is type
size_t on 64-bit Windows and int on all other platforms.

Remarks

These functions fetch the next value of a multi-valued private attribute. If no value has yet been
fetched from this attribute, these functions behave 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 and
MC_Get_Next_pValue_To_UnicodeString, 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.

For `MC_Get_Next_pValue_To_UnicodeString`, it is required to call `MC_EnableUnicodeConversion` first. See documentation for `MC_EnableUnicodeConversion`.

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.

<table>
<thead>
<tr>
<th>Function</th>
<th>May be used to retrieve values from attributes with these Value Representations</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_Get_Next_pValue_To_Float</td>
<td>DS, FD, FL, IS, SL, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Get_Next_pValue_To_Double</td>
<td>DS, FD, FL, IS, SL, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Get_Next_pValue_To_ShortInt</td>
<td>DS, FD, FL, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Get_Next_pValue_To_UShortInt</td>
<td>DS, FD, FL, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Get_Next_pValue_To_Int</td>
<td>DS, FD, FL, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Get_Next_pValue_To_UIInt</td>
<td>DS, FD, FL, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Get_Next_pValue_To_LongInt</td>
<td>AT, DS, FD, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Get_Next_pValue_To_UlongInt</td>
<td>AT, DS, FD, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Get_Next_pValue_To_LongLong</td>
<td>AT, DS, FD, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Get_Next_pValue_To_UlongLong</td>
<td>AT, DS, FD, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Get_Next_pValue_To_UnicodeString</td>
<td>LO, LT, PN, SH, ST, UC, UT</td>
</tr>
</tbody>
</table>

**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`.

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

**Return Value**

One of the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
</tbody>
</table>
MC_EMPTY_VALUE  The attribute has no value assigned to it.
MC_NULL_VALUE   The attribute’s value is NULL (i.e. its 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 any 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”.

```c
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 */
    int ParentMsgID; /* ID of parent of message or item object containing the attribute */
    int MsgLevel; /* Nesting level in data set hierarchy of the message or item object containing the attribute */
} 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 the enumerated **MC_STATUS** codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>Another validation violation exists and its information is returned at “ErrorInfo”.</td>
</tr>
<tr>
<td><strong>MC_INVALID_MESSAGE_ID</strong></td>
<td>The MessageID value is not a valid message object ID.</td>
</tr>
<tr>
<td><strong>MC_NULL_POINTER_PARM</strong></td>
<td>ErrorInfo was NULL.</td>
</tr>
<tr>
<td><strong>MC_END_OF_LIST</strong></td>
<td>The error list has been exhausted, or no <strong>MC_Validate_Message</strong> call has been made for this message.</td>
</tr>
</tbody>
</table>

**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,
MCgetStatus MC_Get_Next_Value_To_Float (int MsgFileItemID,
        unsigned long Tag,
        float *Value)
)
MCgetStatus MC_Get_Next_Value_To_Double (int MsgFileItemID,
        unsigned long Tag,
        double *Value)
)
MCgetStatus MC_Get_Next_Value_To_ShortInt (int MsgFileItemID,
        unsigned long Tag,
        short int *Value)
)
MCgetStatus MC_Get_Next_Value_To_UShortInt (int MsgFileItemID,
        unsigned long Tag,
        unsigned short *Value)
)
MCgetStatus MC_Get_Next_Value_To_Int (int MsgFileItemID,
        unsigned long Tag,
        int *Value)
)
MCgetStatus MC_Get_Next_Value_To_UInt (int MsgFileItemID,
        unsigned long Tag,
        unsigned int *Value)
)
MCgetStatus MC_Get_Next_Value_To_LongInt (int MsgFileItemID,
        unsigned long Tag,
        long int *Value)
)
MCgetStatus MC_Get_Next_Value_To_ULongInt (int MsgFileItemID,
        unsigned long Tag,
        unsigned long *Value)
)
MCgetStatus MC_Get_Next_Value_To_LongLong (int MsgFileItemID,
        unsigned long Tag,
        long long *Value)
)
MCgetStatus MC_Get_Next_Value_To_ULongLong (int MsgFileItemID,
        unsigned long Tag,
        unsigned long long *Value)
)
MCgetStatus MC_Get_Next_Value_To_String (int MC_DT DataType,
        MC_size_t BufferSize,
        void *Value,
        int *ValueSize
)
```c
int MsgFileItemID,
unsigned long Tag,
MC_size_t BufferSize,
char *Value
)
MC_STATUS MC_Get_Next_Value_To_UnicodeString(
    int MsgFileItemID,
    unsigned long Tag,
    MC_size_t BufferSize,
    int *ValueSize,
    MC_Uhar *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
- **LongLong_Type**  
  Binary 64-bit integer number
- **ULongLong_Type**  
  Binary 64-bit unsigned integer number
- **Float_Type**  
  Binary Floating point number
- **Double_Type**  
  Binary Floating point number
- **Buffer_Type**  
  Binary byte value

**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. This parameter is type size_t  
on 64-bit Windows and int on all other platforms.

### Remarks

These functions fetch the next value of a multi-valued attribute. If no value has yet been fetched from  
this attribute, these functions behave 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** and **MC_Get_Next_Value_To_UnicodeString**, 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.

For **MC_Get_Next_Value_To_UnicodeString**, it requires to call **MC_Enable_Unicode_Conversion** first. See documentation for **MC_Enable_Unicode_Conversion**.

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**.

<table>
<thead>
<tr>
<th>Function</th>
<th>May be used to retrieve values from attributes with these Value Representations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_Get_Next_Value_To_Float</strong></td>
<td>DS, FD, FL, IS, SL, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td><strong>MC_Get_Next_Value_To_Double</strong></td>
<td>DS, FD, FL, IS, SL, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td><strong>MC_Get_Next_Value_To_ShortInt</strong></td>
<td>DS, FD, FL, IS, SL, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td><strong>MC_Get_Next_Value_To_UShortInt</strong></td>
<td>DS, FD, FL, IS, SL, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td><strong>MC_Get_Next_Value_To_Int</strong></td>
<td>DS, FD, FL, IS, SL, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td><strong>MC_Get_Next_Value_To(UInt)</strong></td>
<td>DS, FD, FL, IS, SL, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td><strong>MC_Get_Next_Value_To_LongInt</strong></td>
<td>AT, DS, FD, FL, IS, SL, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td><strong>MC_Get_Next_Value_To_ULongInt</strong></td>
<td>AT, DS, FD, FL, IS, SL, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td><strong>MC_Get_Next_Value_To_LongLong</strong></td>
<td>AT, DS, FD, FL, IS, SL, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td><strong>MC_Get_Next_Value_To_UlongLong</strong></td>
<td>AT, DS, FD, FL, IS, SL, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td><strong>MC_Get_Next_Value_To_UnicodeString</strong></td>
<td>LO, LT, PN, SH, ST, UC, UT</td>
</tr>
</tbody>
</table>

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 defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_EMPTY_VALUE</td>
<td>The attribute has no value assigned to it.</td>
</tr>
<tr>
<td>MC_NULL_VALUE</td>
<td>The attribute’s value is NULL (i.e. its length is zero).</td>
</tr>
<tr>
<td>MC_NO_MORE_VALUES</td>
<td>There are no more values for this attribute.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_TAG</td>
<td>The message does not contain an attribute with an ID of <code>Tag</code>.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td><code>MsgFileItemID</code>is not a valid message object ID, file object ID or item object ID.</td>
</tr>
<tr>
<td>MC_BUFFER_TOO_SMALL</td>
<td>Buffer is not large enough to contain the value.</td>
</tr>
<tr>
<td></td>
<td><em>(MC_Get_Next_Value and MC_Get_Next_Value_To_String only)</em></td>
</tr>
<tr>
<td>MC_INCOMPATIBLE_VR</td>
<td>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.</td>
</tr>
<tr>
<td>MC_INCOMPATIBLE_VALUE</td>
<td>The attribute’s value is not consistent with its VR. (This should never happen.)</td>
</tr>
<tr>
<td>MC_VALUE_OUT_OF_RANGE</td>
<td>The attribute numeric value is too large to be accommodated by the receiving data type. (E.g., using <code>MC_Get_Value_To_Int</code> when the value is 123.45; or instances where the value’s sign (+</td>
</tr>
<tr>
<td>MC_INVALID_DATA_TYPE</td>
<td>The <code>DataType</code> parameter is invalid.</td>
</tr>
<tr>
<td>MC_CANNOT_COMPLY</td>
<td>The function fails to process the request. Consult Merge DICOM Toolkit log file for detail.</td>
</tr>
</tbody>
</table>

### See Also

- MC_Get_pValue... Functions
- MC_Get_Next_pValue... Functions
- MC_Get_Next_Value... Functions
- MC_Get_Value_To_Function
- MC_Enable_Unicode_Conversion
- MC_Get_Offset_Table_To_Function

**MC_Get_Offset_Table_To_Function**

Retrieves the basic offset table of encapsulated pixel data from a message.

**Synopsis**

```c
#include "mc3msg.h"

MC_STATUS MC_Get_Offset_Table_To_Function (  
    int    MsgFileItemID,  
    void *UserInfo,  
    void *YourGetFunction,  
)
```
**MsgFileItemID** The identifier assigned to this object by the
*MC_Open_Message, MC_Open_Empty_Message, MS_CCreate_File, MC_Create_Empty_File,* or
*MC_Open_Item* functions.

**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 offset table

The function must be prototyped as follows:

```c
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, MS_CCreate_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_Offset_Table_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_Offset_Table_To_Function* function is used to fetch a basic offset table of an
encapsulated pixel data attribute.

The (*YourGetFunction*) function is used to provide a byte array which contains an offset table data. This byte array needs to be converted to array of integers, where every integer consists of four bytes. The first integer is the offset table length and all subsequential integer values are the corresponding frame offsets.
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 the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally, and there are no more frames in the message.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>YourGetFunction is NULL</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>MsgFileItemID is not a valid message object ID, file object ID or item object ID.</td>
</tr>
<tr>
<td>MC_INVALID_TAG</td>
<td>The message does not contain an attribute with an ID of Tag.</td>
</tr>
<tr>
<td>MC_EMPTY_VALUE</td>
<td>No value has been assigned to this attribute yet.</td>
</tr>
<tr>
<td>MC_CALLBACK_CANNOT_COMPLY</td>
<td>YourGetFunction returned MC_CANNOT_COMPLY.</td>
</tr>
</tbody>
</table>

See Also

MC_Get_Encapsulated_Value_To_FUNCTION
MC_Get_Next_Encapsulated_Value_To_FUNCTION
MC_Get_Frame_To_FUNCTION
MC_Register_Compression_Callbacks
MC_Get_pAttribute_Info

Retrieves information about a given private attribute in a message.

**Synopsis**

```c
#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`, `UC`, `UR`, `UT`, `UI`, `SS`, `US`, `AT`, `SL`, `UL`, `SV`, `UV`, `FL`, `FD`, `OB`, `OW`, `OV`, `OL`, `OD`, `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 the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
</tbody>
</table>
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 Merge DICOM Toolkit’s data dictionary through use of an extended toolkit.
**Return Value**

One of the enumerated `MC_STATUS` codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MC_NORMAL_COMPLETION</code></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><code>MC_INVALID_TAG</code></td>
<td>The Merge DICOM Toolkit data dictionary does not contain an attribute identified by <code>ElementByte</code> in <code>Group</code> for <code>PrivateCode</code>.</td>
</tr>
<tr>
<td><code>MC_INVALID_GROUP</code></td>
<td><code>Group</code> was not an odd number.</td>
</tr>
<tr>
<td><code>MC_NULL_POINTER_PARM</code></td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td><code>MC_DDFILE_ERROR</code></td>
<td>An error occurred while trying to access the Merge DICOM Toolkit data dictionary. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td><code>MC_MISSING_CONFIG_PARM</code></td>
<td>This error occurs when the toolkit can not obtain the name of the data dictionary.</td>
</tr>
<tr>
<td><code>MC_BUFFER_TOO_SMALL</code></td>
<td><code>Name</code> is not large enough to contain the descriptive string.</td>
</tr>
</tbody>
</table>

**See Also**

`MC_Get_Tag_Info` `MC_Get_Attribute_Info` `MC_Get_pAttribute_Info` `MC_Get_pTags_Dict_Info`  
Retrieves all dictionary information for a given private tag.

**Synopsis**

```c
#include "mc3msg.h"

MC_STATUS MC_Get_pTags_Dict_Info (  
    char *PrivateCode,  
    unsigned short Group,  
    unsigned char ElementByte,  
    char *Name,  
    int NameSize,  
    MC_VR* ValueRep,  
    unsigned short *ValueMult_low,  
    unsigned short *ValueMult_high
)
```

- **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`.

- **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, UC, UR,
   UT, UI, SS, US, AT, SL, UL, SV, UV, FL, FD, OB, OW, OV, OL,
   OD, 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_pTags_Dict_Info retrieves information about a private attribute identified by ElementByte for
the PrivateCode within the given Group. The attribute’s name is returned at Name, the value
representation code is returned at ValueRep, and the value multiplicities are returned in
ValueMult_low and ValueMult_high.

This function will only work with private attributes added to the Merge DICOM Toolkit’s data dictionary
through use of an extended toolkit.

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>Name, ValueRep, ValueMult_low, ValueMult_High, and/or PrivateCode invalid.</td>
</tr>
<tr>
<td>MC_BUFFER_TOO_SMALL</td>
<td>NameSize is not large enough to contain Name.</td>
</tr>
<tr>
<td>MC_MISSING_CONFIG_PARM</td>
<td>This error occurs when the toolkit can not obtain the name of the data dictionary.</td>
</tr>
<tr>
<td>MC_DDFILE_ERROR</td>
<td>An error occurred while trying to access the Merge DICOM Toolkit data dictionary. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_INVALID_TAG</td>
<td>The Merge DICOM Toolkit data dictionary does not contain an attribute identified by ElementByte in Group for PrivateCode.</td>
</tr>
</tbody>
</table>

See Also

MC_Get_pTag_Info    MC_Get_Tags_Dict_Info

MC_Get_pValue... Functions

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

Synopsis

```c
#include “mc3msg.h”
MC_STATUS MC_Get_pValue (  
   int MsgFileItemID,
   char *PrivateCode,
```
unsigned short Group,
unsigned char ElementByte,
MC_DT DataType,
MC_size_t BufferSize,
void *Value,
int *ValueSize)

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_Double {
    int MsgFileItemID,
    char *PrivateCode,
    unsigned short Group,
    unsigned char ElementByte,
    double *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_USHORTInt {
    int MsgFileItemID,
    char *PrivateCode,
    unsigned short Group,
    unsigned char ElementByte,
    unsigned short *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_UINT {
    int MsgFileItemID,
    char *PrivateCode,
    unsigned short Group,
    unsigned char ElementByte,
    unsigned 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_ULongInt (  
    int MsgFileItemID,  
    char *PrivateCode,  
    unsigned short Group,  
    unsigned char ElementByte,  
    unsigned long *Value
)

MC_STATUS MC_Get_pValue_To_LongLong (  
    int MsgFileItemID,  
    char *PrivateCode,  
    unsigned short Group,  
    unsigned char ElementByte,  
    long long *Value
)

MC_STATUS MC_Get_pValue_To_ULongLong (  
    int MsgFileItemID,  
    char *PrivateCode,  
    unsigned short Group,  
    unsigned char ElementByte,  
    unsigned long long *Value
)

MC_STATUS MC_Get_pValue_To_String (  
    int MsgFileItemID,  
    char *PrivateCode,  
    unsigned short Group,  
    unsigned char ElementByte,  
    MC_size_t BufferSize,  
    char *Value
)

MC_STATUS MC_Get_pValue_To_UnicodeString (  
    int MsgFileItemID,  
    char *PrivateCode,  
    unsigned short Group,  
    unsigned char ElementByte,  
    MC_size_t BufferSize,  
    int *ValueSize,  
    MC_Uhar *Value
)

MC_STATUS MC_Get_pValue_To_Buffer (  
    int MsgFileItemID,  
    char *PrivateCode,  
    unsigned short Group,  
    unsigned char ElementByte,  
    MC_size_t BufferSize,  
    void *Value,  
    int *ValueSize
)

MsgFileItemID: The identifier assigned to this object by the  
MD 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

- **LongLong_Type**  
  Binary 64-bit integer number

- **UlongLong_Type**  
  Binary 64-bit unsigned integer number

- **Float_Type**  
  Binary Floating point number

- **Double_Type**  
  Binary Floating point number

- **Buffer_Type**  
  Binary byte value

**Value**  
The attribute’s value will be returned here.

**BufferSize**  
The size of the Value buffer in bytes. This parameter is type size_t on 64-bit Windows and int on all other platforms.

**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, MC_Get_pValue_To_UnicodeString 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.

For MC_Get_pValue_To_UnicodeString, it requires to call MC_Enable_Unicode_Conversion first. See documentation for MC_Enable_Unicode_Conversion.

MC_Get_pValue_To_Buffer is used to retrieve the value of a private attribute of a binary value representation of UN, OB, OW, OV, OL, OD or OF, or a text value representation of UR or UT, or 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_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.

<table>
<thead>
<tr>
<th>Function</th>
<th>May be used to retrieve values from attributes with these Value Representations</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_Get_pValue_To_Float</td>
<td>DS, FD, FL, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Get_pValue_To_Double</td>
<td>DS, FD, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Get_pValue_To_ShortInt</td>
<td>DS, FD, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Get_pValue_To_USHORTInt</td>
<td>DS, FD, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Get_pValue_To_Int</td>
<td>DS, FD, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Get_pValue_To(UInt)</td>
<td>DS, FD, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Get_pValue_To_LongInt</td>
<td>AT, DS, FD, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Get_pValue_ToULONGInt</td>
<td>AT, DS, FD, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Get_pValue_To_LongLong</td>
<td>AT, DS, FD, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Get_pValue_To_ULongLong</td>
<td>AT, DS, FD, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Get_pValue_To_UnicodeString</td>
<td>LO, LT, PN, SH, ST, UC, UT</td>
</tr>
<tr>
<td>MC_Get_pValue_To_Buffer</td>
<td>UNKNOWN VR, OB, OW, OL, OD, OF, UR, UT</td>
</tr>
</tbody>
</table>

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 defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_TAG</td>
<td>The message does not contain an attribute identified by ElementByte in Group for PrivateCode.</td>
</tr>
<tr>
<td>MC_INVALID_PRIVATE_CODE</td>
<td>The message does not contain any attributes in Group for PrivateCode.</td>
</tr>
<tr>
<td>MC_INVALID_GROUP</td>
<td>Group was not an odd number.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>MsgFileItemID is not a valid message object ID, file object ID or item object ID.</td>
</tr>
<tr>
<td>MC_EMPTY_VALUE</td>
<td>The attribute has no value assigned to it.</td>
</tr>
<tr>
<td>MC_NULL_VALUE</td>
<td>The attribute’s value is NULL (i.e. its length is zero).</td>
</tr>
<tr>
<td>MC_BUFFER_TOO_SMALL</td>
<td>Buffer is not large enough to contain the value. (MC_Get_pValue, MC_Get_pValue_To_Buffer and MC_Get_pValue_To_String only)</td>
</tr>
<tr>
<td>MC_INCOMPATIBLE_VR</td>
<td>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.</td>
</tr>
<tr>
<td>MC_INCOMPATIBLE_VALUE</td>
<td>The attribute’s value is not consistent with its VR. (This should never happen.)</td>
</tr>
<tr>
<td>MC_VALUE_OUT_OF_RANGE</td>
<td>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(+</td>
</tr>
<tr>
<td>MC_INVALID_DATA_TYPE</td>
<td>The DataType parameter is invalid.</td>
</tr>
</tbody>
</table>

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,
```c
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 defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>CountPtr was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_TAG</td>
<td>The message does not contain an attribute identified by ElementByte in Group for PrivateCode.</td>
</tr>
<tr>
<td>MC_INVALID_PRIVATE_CODE</td>
<td>The message does not contain any attributes in Group for PrivateCode.</td>
</tr>
<tr>
<td>MC_INVALID_GROUP</td>
<td>Group was not an odd number.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>MsgFileItemID is not a valid message object ID, file object ID or item object ID.</td>
</tr>
<tr>
<td>MC_NULL_VALUE</td>
<td>The attribute contains a NULL value (i.e. its value is of length zero).</td>
</tr>
<tr>
<td>MC_EMPTY_VALUE</td>
<td>No value has been assigned to the attribute.</td>
</tr>
</tbody>
</table>

### See Also

MC_Get_Value_Count
**MC_Get_pValue_Length**

Returns the length of a private attribute value.

**Synopsis**

```c
#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 defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_INVALID_TAG</strong></td>
<td>The message does not contain an attribute identified by <code>ElementByte</code> in <code>Group</code> for <code>PrivateCode</code>.</td>
</tr>
<tr>
<td><strong>MC_NULL_POINTER_PARM</strong></td>
<td><code>Length</code> or <code>PrivateCode</code> was NULL.</td>
</tr>
</tbody>
</table>
MC_INVALID_PRIVATE_CODE  The message does not contain any 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 Merge DICOM Toolkit 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 binary value representation of OB, OW, OV, OL, OD or OF, or a numeric value representation of AT, SS, US, SL, UL, SV, UV, FL or FD, or a text value representation of UR or UT, 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 

**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 

**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 UN, OB, OW, OV, OL, OD, OF, AT, SS, US, SL, UL, SV, UV, FL, FD, UR or UT. Such attributes tend to have values of great length. To accommodate this, one uses the MC_Get_pValue_To_Function function to specify the name of a function (YourGetFunction) which Merge DICOM Toolkit, 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 defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_TAG</td>
<td>The message does not contain an attribute identified by ElementByte in Group for PrivateCode.</td>
</tr>
<tr>
<td>MC_INVALID_PRIVATE_CODE</td>
<td>The message does not contain any attributes in Group for PrivateCode.</td>
</tr>
<tr>
<td>MC_INVALID_GROUP</td>
<td>Group was not an odd number.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>MsgFileItemID is not a valid message object ID, file object ID or item object ID.</td>
</tr>
<tr>
<td>MC_INCOMPATIBLE_VR</td>
<td>The attribute’s value representation was not OB, OW, or OF.</td>
</tr>
<tr>
<td>MC_EMPTY_VALUE</td>
<td>No value has been assigned to this attribute yet.</td>
</tr>
<tr>
<td>MC_CALLBACK_CANNOT_COMPLY</td>
<td>YourGetFunction returned with MC_CANNOT_COMPLY.</td>
</tr>
</tbody>
</table>
See Also

<table>
<thead>
<tr>
<th>Function</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_Get_Value</td>
<td>MC_Get_Value_To_Function</td>
</tr>
<tr>
<td>MC_Get_Value_To_Float</td>
<td>MC_Get_Value_To_Double</td>
</tr>
<tr>
<td>MC_Get_Value_To_ShortInt</td>
<td>MC_Get_Value_To_UShortInt</td>
</tr>
<tr>
<td>MC_Get_Value_To_LongInt</td>
<td>MC_Get_Value_To_Int</td>
</tr>
<tr>
<td>MC_Get_Value_To_ULONGInt</td>
<td>MC_Get_Value_To_UInt</td>
</tr>
<tr>
<td>MC_Get_Value_To_NULL</td>
<td>MC_Get_Value_To_Empty</td>
</tr>
<tr>
<td>MC_Get_pValue</td>
<td>MC_Get_pValue_To_Double</td>
</tr>
<tr>
<td>MC_Get_pValue_To_Float</td>
<td>MC_Get_pValue_To_UShortInt</td>
</tr>
<tr>
<td>MC_Get_pValue_To_ShortInt</td>
<td>MC_Get_pValue_To_Int</td>
</tr>
<tr>
<td>MC_Get_pValue_To_LONGInt</td>
<td>MC_Get_pValue_To_UInt</td>
</tr>
<tr>
<td>MC_Get_pValue_To_ULONGInt</td>
<td>MC_Get_pValue_To_UInt</td>
</tr>
<tr>
<td>MC_Get_pValue_To_NULL</td>
<td>MC_Get_pValue_To_Empty</td>
</tr>
<tr>
<td>MC_Get_Next_Value</td>
<td>MC_Get_Next_Value_To_Float</td>
</tr>
<tr>
<td>MC_Get_Next_Value_To_Float</td>
<td>MC_Get_Next_Value_To_Double</td>
</tr>
<tr>
<td>MC_Get_Next_Value_To_ShortInt</td>
<td>MC_Get_Next_Value_To_UShortInt</td>
</tr>
<tr>
<td>MC_Get_Next_Value_To_LONGInt</td>
<td>MC_Get_Next_Value_To_Int</td>
</tr>
<tr>
<td>MC_Get_Next_Value_To_ULONGInt</td>
<td>MC_Get_Next_Value_To_UInt</td>
</tr>
<tr>
<td>MC_Get_Next_pValue</td>
<td>MC_Get_Next_pValue_To_Double</td>
</tr>
<tr>
<td>MC_Get_Next_pValue_To_Float</td>
<td>MC_Get_Next_pValue_To_UShortInt</td>
</tr>
<tr>
<td>MC_Get_Next_pValue_To_ShortInt</td>
<td>MC_Get_Next_pValue_To_Int</td>
</tr>
<tr>
<td>MC_Get_Next_pValue_To_LONGInt</td>
<td>MC_Get_Next_pValue_To(UInt)</td>
</tr>
<tr>
<td>MC_Get_Next_pValue_To_ULONGInt</td>
<td>MC_Get_Next_pValue_To(UInt)</td>
</tr>
</tbody>
</table>

MC_Get_Stream_Length

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

Synopsis

```c
#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 Merge DICOM Toolkit 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
- ENCAPSULATED_UNCOMPRESSED_ELEMENT
- 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
- HEVC_H265_M10P_LEVEL_5_1
- HEVC_H265_MP_LEVEL_5_1
- JPIP_REFERENCED
- JPIP_REFERENCED_DEFLATE
- MPEG2_MPML
- MPEG2_MPHL
- MPEG4_AVC_H264_HP_LEVEL_4_1
- MPEG4_AVC_H264_BDC_HP_LEVEL_4_1
- MPEG4_AVC_H264_HP_LEVEL_4_2_2D
- MPEG4_AVC_H264_HP_LEVEL_4_2_3D
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 defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>The MessageID value is not a valid message object ID.</td>
</tr>
<tr>
<td>MC_INVALID_TRANSFER_SYNTAX</td>
<td>SyntaxType is not one of the TRANSFER_SYNTAX codes defined in “mc3msg.h”.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>LengthPtr was NULL.</td>
</tr>
</tbody>
</table>

See Also

MC_Message_To_Stream

MC_Get_String_Config_Value

Used to get the value of a character string toolkit 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:
MERGECOM_3_PROFILE
MERGECOM_3_SERVICES
MERGECOM_3_APPLICATIONS
CAPTURE_FILE
COMPRESSION_RGB_TRANSFORM_FORMAT
DECODER_TAG_FILTER
DEFLATED_EXPLICIT_LITTLE_ENDIAN_SYNTAX
DICTIONARY_ACCESS
DICTIONARY_FILE
ENCAPSULATED_UNCOMPRESSED_ELE_SYNTAX
EXPLICIT_BIG_ENDIAN_SYNTAX
EXPLICIT_LITTLE_ENDIAN_SYNTAX
HEVC_H265_M10P_LEVEL_5_1_SYNTAX
HEVC_H265_MP_LEVEL_5_1_SYNTAX
IMPLEMENTATION_CLASS_UID
IMPLEMENTATION_VERSION
IMPLICIT_BIG_ENDIAN_SYNTAX
IMPLICIT_LITTLE_ENDIAN_SYNTAX
INITIATOR_NAME
IP_TYPE
JPEG_2000LOSSLESS_ONLY_SYNTAX
JPEG_2000_MC_LOSSLESS_ONLY_SYNTAX
JPEG_2000_MC_SYNTAX
JPEG_2000_SYNTAX
JPEG_BASELINE_SYNTAX
JPEG_EXTENDED_2_4_SYNTAX
JPEG_EXTENDED_3_5_SYNTAX
JPEG_EXTENDED_HIER_16_18_SYNTAX
JPEG_EXTENDED_HIER_17_19_SYNTAX
JPEG_FULL_PROG_HIER_24_26_SYNTAX
JPEG_FULL_PROG_HIER_25_27_SYNTAX
JPEG_FULL_PROG_NON_HIER_10_12_SYNTAX
JPEG_FULL_PROG_NON_HIER_11_13_SYNTAX
JPEG_LOSSLESS_HIER_14_SYNTAX
JPEG_LOSSLESS_HIER_28_SYNTAX
JPEG_LOSSLESS_HIER_29_SYNTAX
JPEG_LOSSLESS_NON_HIER_14_SYNTAX
JPEG_LOSSLESS_NON_HIER_15_SYNTAX
JPEG_LS_LOSSLESS_SYNTAX
JPEG_LS_LOSSY_SYNTAX
JPEG_SPEC_HIER_20_22_SYNTAX
JPEG_SPEC_HIER_21_23_SYNTAX
JPEG_SPEC_NON_HIER_6_8_SYNTAX
JPEG_SPEC_NON_HIER_7_9_SYNTAX
JPIP_REFERENCED_DEFLECT_SYNTAX
JPIP_REFERENCED_SYNTAX
LARGE_DATA_STORE
LICENSE
LOCAL_APPL_CONTEXT_NAME
LOG_FILE
MPEG2_MPHL_SYNTAX
MPEG2_MPML_SYNTAX
MPEG4_AVC_H264_BDC_HP_LEVEL_4_1_SYNTAX
MPEG4_AVC_H264_HP_LEVEL_4_1_SYNTAX
MPEG4_AVC_H264_HP_LEVEL_4_2_2D_SYNTAX
MPEG4_AVC_H264_HP_LEVEL_4_2_3D_SYNTAX
MPEG4_AVC_H264_STEREO_HP_LEVEL_4_2_SYNTAX
MSG_INFO_FILE
NULL_TYPE3_VALIDATION
PEGASUS_DISP_REG_NAME
PEGASUS_DISP_REGISTRATION
PEGASUS_OP_D2SEPLUS_NAME
PEGASUS_OP_D2SEPLUS_REGISTRATION
PEGASUS_OP_J2KE_NAME
PEGASUS_OP_J2KE_REGISTRATION
PEGASUS_OP_J2KP_NAME
PEGASUS_OP_J2KP_REGISTRATION
PEGASUS_OP_JLSE_NAME
PEGASUS_OP_JLSE_REGISTRATION
PEGASUS_OP_JLSP_NAME
PEGASUS_OP_JLSP_REGISTRATION
PEGASUS_OP_LIE3PLUS_NAME
PEGASUS_OP_LIE3PLUS_REGISTRATION
PEGASUS_OP_LIP3PLUS_NAME
PEGASUS_OP_LIP3PLUS_REGISTRATION
PEGASUS_OP_SE2PLUS_NAME
PEGASUS_OP_SE2PLUS_REGISTRATION
PEGASUS_OPCODE_PATH
PRIVATE_SYNTAX_1_SYNTAX
PRIVATE_SYNTAX_2_SYNTAX
RECEIVER_NAME
RLE_SYNTAX
SMPTE_ST_2110_20_UNCOMPRESSED_PROGRESSIVE_ACTIVE_VIDEO_SYNTAX
SMPTE_ST_2110_20_UNCOMPRESSED_INTERLACED_ACTIVE_VIDEO_SYNTAX
SMPTE_ST_2110_30_PCM_DIGITAL_AUDIO_SYNTAX
TEMP_FILE_DIRECTORY
UNKNOWN_VR_CODE

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

**ValueLength**  The length of **Value**.

**Value**  A character string to copy the value of **Parm** into.

### Remarks

The Merge DICOM Toolkit 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 toolkit configuration elsewhere in this manual.

### Return Value

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

<table>
<thead>
<tr>
<th><strong>Value</strong></th>
<th><strong>Meaning</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_INVALID_PARAMETER_NAME</strong></td>
<td>Parameter specified is not one of the above listed parameters.</td>
</tr>
</tbody>
</table>
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_Set_Int_Config_Value
MC_Get_Log_Destination  MC_Set_Log_Destination
MC_Get_Long_Config_Value  MC_Set_Long_Config_Value
MC_Set_String_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  
A tag identifying a message attribute.

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 the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_TAG</td>
<td>The attribute identified by Tag is not a valid DICOM attribute.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_DDFILE_ERROR</td>
<td>An error occurred while trying to access the Merge DICOM Toolkit data dictionary. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_MISSING_CONFIG_PARM</td>
<td>This error occurs when the toolkit can not obtain the name of the data dictionary.</td>
</tr>
<tr>
<td>MC_BUFFER_TOO_SMALL</td>
<td>Name is not large enough to contain the descriptive string.</td>
</tr>
</tbody>
</table>
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  
)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>A tag identifying a message attribute.</td>
</tr>
<tr>
<td>Name</td>
<td>A buffer to contain the tag's name.</td>
</tr>
<tr>
<td>NameSize</td>
<td>The size, in bytes, of Name</td>
</tr>
<tr>
<td>ValueRep</td>
<td>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, UC, UR, UT, UI, SS, US, AT, SL, UL, SV, UV, FL, FD, OB, OW, OV, OL, OD, OF, SQ, UNKNOWN_VR</td>
</tr>
<tr>
<td>ValueMult_low</td>
<td>The minimum multiplicity will be returned here</td>
</tr>
<tr>
<td>ValueMult_high</td>
<td>The maximum multiplicity will be returned here</td>
</tr>
</tbody>
</table>

Remarks

`MC_Get_Tags_Dict_Info` retrieves information about an attribute identified by `Tag`. The attribute’s Name is returned at `Name`, value 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 the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>`Name, ValueRep, ValueMult_low, ValueMult_High, and/or Tag are invalid.</td>
</tr>
<tr>
<td>MC_BUFFER_TOO_SMALL</td>
<td><code>NameSize</code> is not large enough to contain <code>Name</code>.</td>
</tr>
<tr>
<td>MC_MISSING_CONFIG_PARM</td>
<td>DICTIONARY_FILE is invalid</td>
</tr>
<tr>
<td>MC_DDFILE_ERROR</td>
<td>Data Dictionary problem</td>
</tr>
</tbody>
</table>
MC_INVALID_TAG  Tag is not found

See Also
MC_Get_Attribute_Info

MC_Get_Tag_Keyword
Generates attribute’s keyword for a given tag.

Synopsis
#include “mc3msg.h”
MC_STATUS MC_Get_Tag_Keyword ( 
    unsigned long Tag,
    char *Keyword,
    int Length
)

Tag  A tag identifying a message attribute.
Keyword  A buffer to contain the tag’s keyword.
Length  The size, in bytes, of Keyword

Remarks
MC_Get_Tag_Keyword generates a keyword for standard attributes based on the name of the data element as described by section 6 part 6 in the DICOM standard. The attribute’s Keyword is returned at Keyword and the length of the keyword buffer is returned at Length.

Return Value
One of the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_TAG</td>
<td>Tag is not found or private tag</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>Keyword, Length and/or Tag are invalid.</td>
</tr>
<tr>
<td>MC_BUFFER_TOO_SMALL</td>
<td>Keyword Size is not large enough to contain Keyword.</td>
</tr>
<tr>
<td>MC_MISSING_CONFIG_PARM</td>
<td>DICTIONARY_FILE is invalid</td>
</tr>
<tr>
<td>MC_DDFILE_ERROR</td>
<td>Data Dictionary problem</td>
</tr>
<tr>
<td>MC_INVALID_TAG</td>
<td>Tag is not found</td>
</tr>
</tbody>
</table>

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  
ENCAPSULATED_UNCOMPRESSED_ELE  
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  
HEVC_H265_M10P_LEVEL_5_1  
HEVC_H265_MP_LEVEL_5_1  
JPPIP_REFERENCED  
JPPIP_REFERENCED_DEFLATE  
MPEG2_MPML  
MPEG2_MPHL  
MPEG4_AVC_H264_HP_LEVEL_4_1  
MPEG4_AVC_H264_BDC_HP_LEVEL_4_1  
MPEG4_AVC_H264_HP_LEVEL_4_2_2D
MPEG4_AVC_H264_HP_LEVEL_4_2_3D
MPEG4_AVC_H264_STEREO_HP_LEVEL_4_2,
JPEG_2000_MC
SMPT_E ST_2110_20 UNCOMPRESSED PROGRESSIVE_ACTIVE_VIDEO
SMPT_E ST_2110_20 UNCOMPRESSED INTERLACED_ACTIVE_VIDEO
SMPT E ST_2110_30_PCM_DIGITAL_AUDIO
PRIVATE_SYNTAX_1
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 Merge DICOM Toolkit 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 Merge DICOM Toolkit’s configuration files. Please see the detailed description of toolkit configuration elsewhere in this manual.

Return Value
One of the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_TRANSFER_SYNTAX</td>
<td>SyntaxName is not registered in the Merge DICOM Toolkit configuration files.</td>
</tr>
<tr>
<td>MC_BUFFER_TOO_SMALL</td>
<td>Uid is not large enough to contain the unique identifier.</td>
</tr>
</tbody>
</table>

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 Merge DICOM Toolkit 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 buffer.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_UNKNOWN_SERVICE</td>
<td><em>ServiceName</em> was not registered in the Merge DICOM Toolkit configuration files.</td>
</tr>
<tr>
<td>MC_BUFFER_TOO_SMALL</td>
<td><em>Uid</em> is not large enough to contain the unique identifier.</td>
</tr>
</tbody>
</table>

**See Also**

MC_Get_MergeCOM_Service
MC_Get_MergeCOM_Service_From_UID
MC_Get_Enum_From_Transfer_Syntax
MC_Get_Transfer_Syntax_From_Enum

**MC_Get_User_Identity_Info**

Returns the contents of a user identity association negotiation related field.

**Synopsis**

```c
#include "mergecom.h"

typedef enum {
    PRIMARY_FIELD,
    SECONDARY_Field,
    SERVER_RESPONSE_FIELD
} USER_IDENTITY_FIELD_TYPE;

MC_STATUS MC_Get_User_Identity_Info (  
    int AssocID,
    USER_IDENTITY_FIELD_TYPE FieldType,
    void *Field,
    unsigned short FieldLength
)
```

**AssocID** The identifier assigned to this object by the
MC_Open_Association, MC_Open_Secure_Association,
MC_Process_Association_Request,
MC_Process_Secure_Association_Request,
MC_Wait_For_Association or
MC_Wait_For_Secure_Association function.
**FieldType**
An enumerated value specifying the User Identity related field to retrieve.

**Field**
A buffer into which the user identity field specified by FieldType is copied into. Note that if Field is a text field, a NULL termination character will not be included in the value returned in this parameter.

**FieldLength**
The length of Field in bytes.

**Remarks**

MC_Get_User_Identity_Info returns the value of a field used in user identity association negotiation as defined in DICOM Supplement 99. It is used by an application accepting associations to determine the user identity related fields in an association request (the PRIMARY_FIELD and SECONDARY_FIELD), and it is used by applications opening associations to retrieve the user identity server response, if applicable (the SERVER_RESPONSE_FIELD).

The structure returned by the MC_Get_Association_Info routine contains general information about the user identity negotiation such as the type of user identity information contained in the association negotiation, and if a server response has been requested in the association negotiation. For user identity types that contain additional fields, it is recommended that the MC_Get_User_Identity_Length routine be first used to determine the length of a specific user identity field. The MC_Get_User_Identity_Info routine can then be used to retrieve the actual field's value.

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, MC_Send_Request, MC_Send_Response_Message or MC_Send_Response calls.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>Field was NULL.</td>
</tr>
<tr>
<td>MC_BUFFER_TOO_SMALL</td>
<td>Field was not large enough to contain the user identity field requested.</td>
</tr>
<tr>
<td>MC_INVALID_ASSOC_ID</td>
<td>AssocID is not a valid association object ID.</td>
</tr>
<tr>
<td>MC_STATE_VIOLATION</td>
<td>The call was made after messages have been exchanged over the association.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>
See Also

MC_Open_Association_With_Identity
MC_Accept_Association_With_Identity
MC_Get_User_Identity_Length  MC_Get_Association_Info

**MC_Get_User_Identity_Length**

Returns the length of a user identity association negotiation related field.

**Synopsis**

```cpp
#include "mergecom.h"

typedef enum {
    PRIMARY_FIELD,
    SECONDARY_FIELD,
    SERVER_RESPONSE_FIELD
} USER_IDENTITY_FIELD_TYPE;

MC_STATUS MC_Get_User_Identity_Length (
    int AssocID,
    USER_IDENTITY_FIELD_TYPE FieldType,
    unsigned short *FieldLength
)
```

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

**FieldType**  An enumerated value specifying the User Identity related field
to retrieve.

**FieldLength**  The length of FieldType in bytes is returned here.

**Remarks**

**MC_Get_User_Identity_Length** returns the length of a field used in user identity association
negotiation as defined in DICOM Supplement 99. It is used by an application accepting associations to
determine the length of the user identity related fields in an association request (the PRIMARY_FIELD
and SECONDARY_FIELD), and it is used by applications opening associations to determine the length
of the user identity server response, if applicable (the SERVER_RESPONSE_FIELD).

The structure returned by the **MC_Get_Association_Info** routine contains general information about
the user identity negotiation such as the type of user identity information contained in the association
negotiation, and if a server response has been requested in the association negotiation. For user
identity types that contain additional fields, it is recommended that the
**MC_Get_User_Identity_Length** routine be first used to determine the length of a specific user identity
field. The **MC_Get_User_Identity_Info** routine can then be used to retrieve the actual field’s value.
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`, `MC_Send_Request`, `MC_Send_Response_Message` or `MC_Send_Response` calls.

**Return Value**

One of the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MC_NORMAL_COMPLETION</code></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><code>MC_NULL_POINTER_PARM</code></td>
<td><code>FieldLength</code> was NULL.</td>
</tr>
<tr>
<td><code>MC_INVALID_ASSOC_ID</code></td>
<td><code>AssocID</code> is not a valid association object ID.</td>
</tr>
<tr>
<td><code>MC_STATE_VIOLATION</code></td>
<td>The call was made after messages have been exchanged over the association.</td>
</tr>
<tr>
<td><code>MC_SYSTEM_ERROR</code></td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**See Also**

- `MC_Open_Association_With_Identity`
- `MC_Accept_Association_With_Identity`
- `MC_Get_User_Identity_Info`
- `MC_Get_Association_Info`

**MC_Get_Value... Functions**

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

**Synopsis**

```c
#include "mc3msg.h"

MC_STATUS MC_Get_Value (  
    int MsgFileItemID,
    unsigned long Tag,
    MC_DT DataType,
    MC_size_t BufferSize,
    void *Value,
    int *ValueSize
)

MC_STATUS MC_Get_Value_To_Float (  
    int MsgFileItemID,
    unsigned long Tag,
    float *Value
)

MC_STATUS MC_Get_Value_To_Double (  
    int MsgFileItemID,
    unsigned long Tag,
    double *Value
)
```
MC_STATUS MC_Get_Value_To_ShortInt (int MsgFileItemID,
 unsigned long Tag,
 short int *Value
)

MC_STATUS MC_Get_Value_To_USHORTInt (int MsgFileItemID,
 unsigned long Tag,
 unsigned short *Value
)

MC_STATUS MC_Get_Value_To_Int (int MsgFileItemID,
 unsigned long Tag,
 int *Value
)

MC_STATUS MC_Get_Value_To_UINT (int MsgFileItemID,
 unsigned long Tag,
 unsigned int *Value
)

MC_STATUS MC_Get_Value_To_LongInt (int MsgFileItemID,
 unsigned long Tag,
 long int *Value
)

MC_STATUS MC_Get_Value_ToONGLONG (int MsgFileItemID,
 unsigned long Tag,
 unsigned long long *Value
)

MC_STATUS MC_Get_Value_ToULONGLong (int MsgFileItemID,
 unsigned long Tag,
 unsigned long long *Value
)

MC_STATUS MC_Get_Value_To_String (int MsgFileItemID,
 unsigned long Tag,
 MC_size_t BufferSize,
 char *Value
)

MC_STATUS MC_Get_Value_To_UnicodeString (int MsgFileItemID,
 unsigned long Tag,
 MC_size_t BufferSize,
 int *ValueSize,
 MC_Uhar *Value
)

MC_STATUS MC_Get_Value_To_Buffer (int MsgFileItemID,
 unsigned long Tag,
MC_size_t 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
- LongLong_Type: Binary 64-bit integer number
- UlongLong_Type: Binary 64-bit unsigned integer number
- Float_Type: Binary Floating point number
- Double_Type: Binary Floating point number
- Buffer_Type: Binary byte value

Value The attribute’s value will be returned here.

BufferSize The size of the Value buffer in bytes. This parameter is type
size_t on 64-bit Windows and int on all other platforms.

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,
MC_Get_Value_To_UnicodeString 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.

For MC_Get_Value_To_UnicodeString, it requires to call MC_Enable_Unicode_Conversion first. See
documentation for MC_Enable_Unicode_Conversion.
**MC_Get_Value_To_Buffer** is used to retrieve the value of a standard attribute of a binary value representation of UN, OB, OW, OV, OL, OD or OF, or a text value representation of UR or UT, or an attribute whose value representation is unknown. Using **MC_Get_Value_To_Buffer** for attributes with other Value Representations 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**.

<table>
<thead>
<tr>
<th>Function</th>
<th>May be used to retrieve values from attributes with these Value Representations</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_Get_Value_To_Float</td>
<td>DS, FD, FL, IS, SL, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Get_Value_To_Double</td>
<td>DS, FD, FL, IS, SL, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Get_Value_To_ShortInt</td>
<td>DS, FD, FL, IS, SL, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Get_Value_To_Int</td>
<td>DS, FD, FL, IS, SL, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Get_Value_To_UInt</td>
<td>DS, FD, FL, IS, SL, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Get_Value_To_LongInt</td>
<td>AT, DS, FD, FL, IS, SL, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Get_Value_To_UlongInt</td>
<td>AT, DS, FD, FL, IS, SL, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Get_Value_To_UlongLong</td>
<td>AT, DS, FD, FL, IS, SL, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Get_Value_To_UnicodeString</td>
<td>LO, LT, PN, SH, ST, UC, UT</td>
</tr>
<tr>
<td>MC_Get_Value_To_Buffer</td>
<td>UNKNOWN_VR, OB, OW, OV, OL, OD, OF, UR, UT</td>
</tr>
</tbody>
</table>

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 defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MC_NORMAL_COMPLETION</code></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><code>MC_NULL_POINTER_PARM</code></td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td><code>MC_INVALID_TAG</code></td>
<td>The message does not contain an attribute with an ID of <code>Tag</code>.</td>
</tr>
<tr>
<td><code>MC_INVALID_MESSAGE_ID</code></td>
<td><code>MsgFileItemID</code> is not a valid message object ID, file object ID or item object ID.</td>
</tr>
<tr>
<td><code>MC_EMPTY_VALUE</code></td>
<td>The attribute has no value assigned to it.</td>
</tr>
<tr>
<td><code>MC_NULL_VALUE</code></td>
<td>The attribute’s value is NULL (i.e. it’s length is zero).</td>
</tr>
<tr>
<td><code>MC_BUFFER_TOO_SMALL</code></td>
<td>Buffer is not large enough to contain the value.</td>
</tr>
<tr>
<td></td>
<td>(<code>MC_Get_Value</code>, <code>MC_Get_Value_To_Buffer</code> and <code>MC_Get_Value_To_String</code> only)</td>
</tr>
<tr>
<td><code>MC_INCOMPATIBLE_VR</code></td>
<td>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.</td>
</tr>
<tr>
<td><code>MC_INCOMPATIBLE_VALUE</code></td>
<td>The attribute’s value is not consistent with its VR. (This should never happen.)</td>
</tr>
<tr>
<td><code>MC_VALUE_OUT_OF_RANGE</code></td>
<td>The attribute numeric value is too large to be accommodated by the receiving data type. (E.g. Using <code>MC_Get_Value_To_Int</code> when the value is 123.45; or instances where the value’s sign(+-) would be affected)</td>
</tr>
<tr>
<td><code>MC_INVALID_DATA_TYPE</code></td>
<td>The <code>DataType</code> parameter is invalid.</td>
</tr>
<tr>
<td><code>MC_CANNOT_COMPLY</code></td>
<td>The function fails to process the request. Consult Merge DICOM Toolkit log file for detail.</td>
</tr>
</tbody>
</table>

See Also

- `MC_Get_pValue... Functions`
- `MC_Get_Next_pValue... Functions`
- `MC_Get_Next_Value... Functions`
- `MC_Enable_Unicode_Conversion`

**MC_Get_Value_Count**

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

**Synopsis**

```c
#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 defined in “mcstatus.h”:

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

See Also

MC_Get_pValue_Count

MC_Get_Value_Length

Returns the length of an attribute value.

Synopsis

```c
#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.

For pixel data tag which has a default value representation of OW, the returned length can be odd or even depending on the actual size of pixel data and the effective VR. For example, if pixel data is generated by uncompressing encapsulated data with 8 bits allocation per pixel, the effective VR is set to OB. If actual uncompressed data size is odd, the function returns an odd size length (No padding has been applied). See **MC_Get_Attribute_Info** for retrieving the effective VR of an attribute.

However, if pixel data is retrieved from a file or stream, the length is always even size due to padding.

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 defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_NULL_POINTER_PARM</strong></td>
<td><code>Length</code> was NULL.</td>
</tr>
<tr>
<td><strong>MC_INVALID_TAG</strong></td>
<td>The message does not contain an attribute with an ID of Tag.</td>
</tr>
<tr>
<td><strong>MC_INVALID_MESSAGE_ID</strong></td>
<td><code>MsgFileItemID</code> is not a valid message object ID, file object ID or item object ID.</td>
</tr>
<tr>
<td><strong>MC_NULL_VALUE</strong></td>
<td>The attribute contains a NULL value (i.e. its value is of length zero).</td>
</tr>
<tr>
<td><strong>MC_EMPTY_VALUE</strong></td>
<td>No value has been assigned to the attribute.</td>
</tr>
<tr>
<td><strong>MC_INVALID_VALUE_NUMBER</strong></td>
<td><code>ValueNumber</code> was negative, or the attribute does not have that many values.</td>
</tr>
<tr>
<td><strong>MC_SYSTEM_ERROR</strong></td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>
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 binary value representation of OB, OW, OV, OL, OD or OF, or a numeric value representation of AT, SS, US, SL, UL, SV, UV, FL or FD, or a text value representation of UR or UT, 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.
The `MC_Get_Value_To_Function` function is used to fetch the value of an attribute which has a value representation of UN, OB, OW, OV, OL, OD, OF, AT, SS, US, SL, UL, SV, UV, FL, FD, UR or UT. Such attributes tend to have values of great length. This function can also be used to fetch the value of an attribute which has a numeric value representation (SL, SS, SV, UL, US, UV, AT, FL, FD).

To accommodate this, one uses the `MC_Get_Value_To_Function` function to specify the name of a function (YourGetFunction) which Merge DICOM Toolkit, in turn, will call. 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 defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_TAG</td>
<td>The message does not contain an attribute with an ID of Tag.</td>
</tr>
</tbody>
</table>
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, OF, SL, SS, UL, US, AT, FL or FD.

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 toolkit version.

Synopsis

```c
#include "mc3msg.h"

MC_STATUS MC_Get_Version_String (  
    int VersionLength,
    char *Version
);
```

Parameters:

- `VersionLength`: The length in bytes of `Version`.
- `Version`: The Merge DICOM Toolkit version is returned here.

Remarks

MC_Get_Version_String retrieves a text string containing Merge DICOM Toolkit's version.

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td><code>Version</code> was NULL.</td>
</tr>
<tr>
<td>MC_BUFFER_TOO_SMALL</td>
<td><code>Version</code> is not large enough to contain the descriptive string.</td>
</tr>
</tbody>
</table>

See Also

MC_Library_Initialization

MC_Json_To_Message

Reads attribute values from a DICOM JSON Model string into a message, file or item object.
**Synopsis**

```c
#include "mc3msg.h"

MC_STATUS MC_Json_To_Message (  
    int MessageID,
    void *UserInfo,
    MC_STATUS (*YourGetJsonFunction) ()
)
```

- **MessageID**
  - The identifier of a message, file or item object.

- **UserInfo**
  - Address of data which will be passed on to `YourGetJsonFunction` each time it is called.
  - This may be NULL.

- **YourGetJsonFunction**
  - Name of a function which will be called repeatedly to get the JSON string.

The function must be prototyped as follows:

```c
MC_STATUS YourGetJsonFunction (  
    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_Json_To_Message` function. This may be NULL.

- **CBdataSize**
  - Set `*CBdataSize` to the number of bytes you are providing.

- **CBdataBuffer**
  - Set `*CBdataBuffer` to the address of the data you are providing.

- **CBisFirst**
  - Set to TRUE (not zero) by the toolkit on the first call.

- **CBisLast**
  - Set `*CBisLast` to TRUE (not zero) when you are returning with the last block of JSON data.

**Remarks**

`MC_Json_To_Message` requests that the DICOM JSON Model string buffer be converted into a DICOM message. The JSON data is requested from the user in blocks by calling `YourGetJsonFunction` repeatedly until the entire JSON content has been received by the Merge DICOM Toolkit.

`YourGetJsonFunction` will be called repeatedly to get blocks of data from it. Merge DICOM Toolkit sets `CBisFirst` to TRUE(non-zero) the first time it calls `YourGetJsonFunction` for this message and `YourGetJsonFunction` should set `*CBisLast` to TRUE(non-zero) when it gives Merge DICOM Toolkit the final block of data to be converted.
**CBdataBuffer** is set to the address of a buffer containing the data block to be converted, and **CBdataSize** is set to the number of bytes to be placed at **CBdataBuffer**.

### Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>YourGetJsonFunction parameter was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>The MessageID value is not a valid message, file or item object ID.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_CALLBACK_CANNOT_COMPLY</td>
<td>YourGetJsonFunction returned a value other than MC_NORMAL_COMPLETION.</td>
</tr>
<tr>
<td>MC_LIBRARY_NOT_INITIALIZED</td>
<td>This call was made prior to the initialization of the library.</td>
</tr>
</tbody>
</table>

### See Also

MC_Message_To_Json

### MC_Library_Initialization

Prepares the Merge DICOM Toolkit library and provides optional information to the library.

### Synopsis

```c
#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 defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MC_NORMAL_COMPLETION</code></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><code>MC_CONFIG_INFO_ERROR</code></td>
<td>An error has occurred while trying to initialize configuration information. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td><code>MC_DDFILE_ERROR</code></td>
<td>An error has occurred while trying to access the Merge DICOM Toolkit data dictionary. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td><code>MC_LIBRARY_ALREADY_INITIALIZED</code></td>
<td>The <code>MC_Library_Initialization</code> has already been called and the library is already initialized.</td>
</tr>
<tr>
<td><code>MC_MISSING_CONFIG_PARM</code></td>
<td>This error occurs when the toolkit can not obtain the name of the data dictionary.</td>
</tr>
<tr>
<td><code>MC_NO_FILE_SYSTEM</code></td>
<td>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.</td>
</tr>
<tr>
<td><code>MC_NO_MERGE_INI</code></td>
<td>This error occurs when the toolkit can not open the toolkit initialization file. See the section titled “Configuration” in this manual and the <em>Merge DICOM Toolkit Users Manual</em> for a detailed discussion of the initialization file(s).</td>
</tr>
<tr>
<td><code>MC_SYSTEM_ERROR</code></td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

### See Also

- `MC_Library_Reset`
- `MC_Set_MergeINI`
- `MC_Library_Release`

## MC_Library_Release

Releases all resources used by the Merge DICOM Toolkit 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 defined in "mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_LIBRARY_NOT_INITIALIZED</td>
<td>The library has not been initialized by a call to <strong>MC_Library_Initialization</strong>.</td>
</tr>
</tbody>
</table>

**See Also**

- **MC_Library_Initialization**
- **MC_Set_MergeINI**
- **MC_Library_Reset**

**MC_Library_Reset**

Sets the Merge DICOM Toolkit library back to its initial state.

**Synopsis**

```c
#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 defined in "mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_LIBRARY_NOT_INITIALIZED</td>
<td>The library has not been initialized by a call to <strong>MC_Library_Initialization</strong>.</td>
</tr>
<tr>
<td>MC_CONFIG_INFO_ERROR</td>
<td>An error occurred while trying to initialize configuration information. A message describing the error is written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_DDFILE_ERROR</td>
<td>An error occurred while trying to access the Merge DICOM Toolkit data dictionary. A message describing the error is written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_MISSING_CONFIG_PARM</td>
<td>This error occurs when the toolkit cannot obtain the name of the data dictionary.</td>
</tr>
<tr>
<td>MC_NO_FILE_SYSTEM</td>
<td>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.</td>
</tr>
</tbody>
</table>

© Copyright Merge Healthcare Solutions Inc. 2023
MC_NO_MERGEINI

This error occurs when the toolkit cannot open the toolkit initialization file. Refer to Configuration in this manual and the Merge DICOM Toolkit 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 Merge DICOM Toolkit 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

```c
#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

```c
#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)

Creates a list of the current contents of an item object.

Synopsis

```c
#include “mc3msg.h”
void MC_List_Item (  
    int ItemID,
)
```
MC_List_Item

Prepares a report listing the contents of the current contents of an item object identified by ItemID. The report will be written to the stream identified by StreamHandle. Use NULL to direct the report 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.

```
#include "mc3msg.h"

void MC_List_Item (  
    int ItemID,  
    char *Afilename  
)

```

**Synopsis**

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**

```c
#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_MemoryLog_To_Function

Retrieves the messages that have been placed in the memory log buffer.

**Synopsis**

```c
#include "mc3msg.h"

MC_STATUS MC_MemoryLog_To_Function (  
    int AclearLog,  
    MC_STATUS (*UserFunction)()
)
```
**AclearLog** Memory log will be cleared if this is true.

**UserFunction** The name of a function that will be called once for each message (line) in the memory buffer.

**UserFunction** must be prototyped as follows:

```c
MC_STATUS UserFunction (  
    char *LogLine,  
    int CBisFirst,  
    int CBisLast  
)
```

*LogLine* Address of the log message passed to the user function.

*CBisFirst* This is TRUE (not zero) the first time *UserFunction* is being called.

*CBisLast* This is TRUE (not zero) the last time *UserFunction* is being called.

**Remarks**

The **MC_MemoryLog_To_Function** function is used to fetch the contents of the memory log. The user specifies the name of a function (**UserFunction**) which Merge DICOM Toolkit, in turn, calls. This “callback” function is called repeatedly to provide lines from the memory log.

**UserFunction**

Each message (or line) in the memory log is returned in *LogLine*.

*CBisFirst* is set to TRUE the first time *UserFunction* is called. This provides a clear mechanism for the function to know it is being called the first time. Users often open a file at this time.

*CBisLast* is set to TRUE the last time *UserFunction* is called. This provides a clear mechanism for the function to know it is being called the last time. Users often close a file at this time.

**UserFunction** 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”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td><em>UserFunction</em> parameter was NULL.</td>
</tr>
<tr>
<td>MC_LOG_EMPTY</td>
<td>The memory log buffer has not been created yet or is empty.</td>
</tr>
<tr>
<td>MC_CALLBACK_CANNOT_COMPLY</td>
<td><em>UserFunction</em> returned with <strong>MC_CANNOT_COMPLY</strong>.</td>
</tr>
</tbody>
</table>

**See Also**

MC_Set_Log_Destination  
MC_Register_MemoryLog_Function  
MC_Register_Enhanced_MemoryLog_Function
MC_Message_To_File

Changes a message object into a file object

Synopsis

```c
#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 the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>The MessageID value is not a valid message object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>Filename has a value of NULL.</td>
</tr>
</tbody>
</table>

See Also

MC_File_To_Message

MC_Message_To_Json

Converts a DICOM message, file or item into a JSON string based on DICOM JSON Model and passes
the converted data to the user.

Synopsis

```c
#include "mc3msg.h"

MC_STATUS MC_Message_To_Json ( 
    int MessageID, 
    void *UserInfo, 
    JSON_OPTIONS JsonOptions,
```
MC_STATUS (*YourReceiveJsonFunction)()
)

MessageID
The identifier of a message, file or item object.

UserInfo
Address of data which will be passed on to YourReceiveJsonFunction each time it is called.
This may be NULL.

JsonOptions
Combination of the JSON configuration flags that are defined in the JSON_OPTIONS enumerator list:
JsonOptDefault
JsonOptIncludeBulks
JsonOptIncludePixelData
JsonOptExcludeSequences
JsonOptExcludePrivateAttributes
JsonOptIndentation

YourReceiveJsonFunction
Pointer to a function which will be called repeatedly by the toolkit to receive the JSON string.
The function must be prototyped as follows:
MC_STATUS YourReceiveJsonFunction (  
    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_Json function. This may be NULL.

CBdataSize
The number of bytes of JSON data being provided to you in CBdataBuffer.

CBdataBuffer
The buffer containing JSON data from the message object.

CBisFirst
Is TRUE (not zero) when Merge DICOM Toolkit is providing the first block of JSON data.

CBisLast
Is TRUE (not zero) when Merge DICOM Toolkit is providing the last block of JSON data.

Remarks
MC_Message_To_Json requests that the contents of the message identified by MessageID be converted into a JSON string. The converted message is passed to the user in blocks by calling YourReceiveJsonFunction repeatedly until the entire converted message has been transferred.
MC_Message_To_Json can pass data to YourReceiveJsonFunction by specifying the data’s address in UserInfo. Merge DICOM Toolkit passes the address to YourReceiveJsonFunction in CBuserInfo each time it is called. UserInfo may be NULL.

JsonOptions specify the configuration flags for the DICOM to JSON conversion. The enumerated JSON_OPTIONS flags are defined in “mc3msg.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>JsonOptDefault</td>
<td>Use the default JSON conversion settings.</td>
</tr>
<tr>
<td>JsonOptIncludeBulks</td>
<td>Store bulk attributes (VR is OB or OW) in the JSON.</td>
</tr>
<tr>
<td>JsonOptIncludePixelData</td>
<td>Store Pixel Data buffer in the JSON.</td>
</tr>
<tr>
<td>JsonOptExcludeSequences</td>
<td>Do not store Sequence attributes in the JSON.</td>
</tr>
<tr>
<td>JsonOptExcludePrivateAttributes</td>
<td>Do not store Private attributes in the JSON.</td>
</tr>
<tr>
<td>JsonOptIndentation</td>
<td>Use default indentation in the generated JSON.</td>
</tr>
</tbody>
</table>

YourReceiveJsonFunction

YourReceiveJsonFunction will be called repeatedly to pass blocks of data to it. Merge DICOM Toolkit sets CBisFirst to TRUE(non-zero) the first time it calls YourReceiveJsonFunction for this message and it sets CBisLast to TRUE(non-zero) when it calls YourReceiveJsonFunction with the final block of converted data.

CBdataBuffer is set to the address of a buffer containing the converted data block, and CBdataSize is set to the number of bytes placed at CBdataBuffer. The data buffer is valid only for the duration of the call to YourReceiveJsonFunction.

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>The YourReceiveJsonFunction parameter was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>The MessageID value is not a valid message, file or item object ID.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_MESSAGE_EMPTY</td>
<td>The message has no attributes in it.</td>
</tr>
<tr>
<td>MC_CALLBACK_CANNOT_COMPLY</td>
<td>YourReceiveJsonFunction returned a value other than MC_NORMAL_COMPLETION.</td>
</tr>
<tr>
<td>MC_LIBRARY_NOT_INITIALIZED</td>
<td>This call was made prior to the initialization of the library.</td>
</tr>
</tbody>
</table>

See Also

MC_Json_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

```c
#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 the enumerated `MC_STATUS` codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>The <code>MsgID</code> value is not a valid object ID.</td>
</tr>
<tr>
<td>MC_MSGFILE_ERROR</td>
<td>An error occurred attempting to access the configuration data file for the directory record.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

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

```c
#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
- ENCAPSULATED_UNCOMPRESSED_ELE
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
HEVC_H265_M10P_LEVEL_5_1
HEVC_H265_MP_LEVEL_5_1
JPIP_REFERENCED
JPIP_REFERENCED_DEFLATE
MPEG2_MPML
MPEG2_MPML
MPEG4_AVC_H264_HP_LEVEL_4_1
MPEG4_AVC_H264_BDC_HP_LEVEL_4_1
MPEG4_AVC_H264_HP_LEVEL_4_2_2D
MPEG4_AVC_H264_HP_LEVEL_4_2_3D
MPEG4_AVC_H264_STEREO_HP_LEVEL_4_2
JPEG_2000_MC_LOSSLESS_ONLY
JPEG_2000_MC
SMPTE_ST_2110_20_UNCOMPRESSED_PROG
RESSIVE_ACTIVE_VIDEO
SMPTE_ST_2110_20_UNCOMPRESSED_INTER
LACED_ACTIVE_VIDEO
SMPTE_ST_2110_30_PCM_DIGITAL_AUDIO
PRIVATE_SYNTAX_1
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:

```c
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 Merge DICOM Toolkit is providing the first block of stream data.
- **CBisLast** Is TRUE (not zero) when Merge DICOM Toolkit 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`. Merge DICOM Toolkit 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. Merge DICOM Toolkit 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 set to the number of bytes placed at `*CBdataBuffer`.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>The <code>MessageID</code> value is not a valid message object ID.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_MESSAGE_EMPTY</td>
<td>The message has no attributes in it.</td>
</tr>
<tr>
<td>MC_CALLBACK_CANNOT_COMPLY</td>
<td><code>YourReceiveStreamFunction</code> returned a value other than <code>MC_NORMAL_COMPLETION</code>.</td>
</tr>
<tr>
<td>MC_INVALID_TRANSFER_SYNTAX</td>
<td>An invalid code was used for the <code>SyntaxType</code> parameter.</td>
</tr>
<tr>
<td>MC_TEMPFILE_ERROR</td>
<td>A value for an attribute is stored temporarily on file and an I/O error occurred.</td>
</tr>
</tbody>
</table>

**See Also**

`MC_Stream_To_Message`  
`MC_Get_Stream_Length`

**MC_Message_To_XML**

Converts a DICOM message, file or item into an XML string based on Merge DICOM Model and passes the converted data to the user.

**Synopsis**

```c
#include "mc3msg.h"

MC_STATUS MC_Message_To_XML (  
    int MessageID,
```
```c
void *UserInfo,
XML_OPTIONS XmlOptions,
MC_STATUS (*YourReceiveXMLFunction)()
)

MessageID  The identifier of a message, file or item object.
UserInfo  Address of data which will be passed on to
YourReceiveXMLFunction each time it is called.
This may be NULL.
XmlOptions  Combination of the XML configuration flags that
are defined in the XML_OPTIONS enumerator list:
XmlOptDefault
XmlOptIncludeBulks
XmlOptIncludePixelData
XmlOptExcludeSequences
XmlOptExcludePrivateAttributes
XmlOptBase64Binary
YourReceiveXMLFunction  Pointer to a function which will be called
repeatedly by the toolkit to receive the XML
string.
The function must be prototyped as follows:
MC_STATUS YourReceiveXMLFunction ( 
    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_XML function. This may be NULL.
CBdataSize  The number of bytes of XML data being provided to you in
CBdataBuffer.
CBdataBuffer  The buffer containing XML data from the message object.
CBisFirst  Is TRUE (not zero) when Merge DICOM Toolkit is providing
the first block of XML data.
CBisLast  Is TRUE (not zero) when Merge DICOM Toolkit is providing
the last block of XML data.
**Remarks**

**MC_Message_To_XML** requests that the contents of the message identified by `MessageID` be converted into an XML string. The converted message is passed to the user in blocks by calling `YourReceiveXMLFunction` repeatedly until the entire converted message has been transferred.

**MC_Message_To_XML** can pass data to `YourReceiveXMLFunction` by specifying the data’s address in `UserInfo`. Merge DICOM Toolkit passes the address to `YourReceiveXMLFunction` in `CBuserInfo` each time it is called. `UserInfo` may be NULL.

`XmlOptions` specify the configuration flags for the DICOM to XML conversion. The enumerated `XML_OPTIONS` flags are defined in “mc3msg.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>XmlOptDefault</td>
<td>Use the default XML conversion settings.</td>
</tr>
<tr>
<td>XmlOptIncludeBulsks</td>
<td>Store bulk attributes (VR is OB or OW) in the XML.</td>
</tr>
<tr>
<td>XmlOptIncludePixelData</td>
<td>Store Pixel Data buffer in the XML.</td>
</tr>
<tr>
<td>XmlOptExcludeSequences</td>
<td>Do not store Sequence attributes in the XML.</td>
</tr>
<tr>
<td>XmlOptExcludePrivateAttributes</td>
<td>Do not store Private attributes in the XML.</td>
</tr>
<tr>
<td>XmlOptBase64Binary</td>
<td>Use Base64 encoding for bulks and UN VR attributes.</td>
</tr>
</tbody>
</table>

**YourReceiveXMLFunction**

`YourReceiveXMLFunction` will be called repeatedly to pass blocks of data to it. Merge DICOM Toolkit sets `CBisFirst` to TRUE (non-zero) the first time it calls `YourReceiveXMLFunction` for this message and it sets `CBisLast` to TRUE (non-zero) when it calls `YourReceiveXMLFunction` with the final block of converted data.

`CBdataBuffer` is set to the address of a buffer containing the converted data block, and `CBdataSize` is set to the number of bytes placed at `CBdataBuffer`. The data buffer is valid only for the duration of the call to `YourReceiveXMLFunction`.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>The <code>YourReceiveXMLFunction</code> parameter was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>The <code>MessageID</code> value is not a valid message, file or item object ID.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_MESSAGE_EMPTY</td>
<td>The message has no attributes in it.</td>
</tr>
<tr>
<td>MC_CALLBACK_CANNOT_COMPLY</td>
<td><code>YourReceiveXMLFunction</code> returned a value other than <code>MC_NORMAL_COMPLETION</code>.</td>
</tr>
</tbody>
</table>
MC_LIBRARY_NOT_INITIALIZED
This call was made prior to the initialization of the library.

See Also
MC_XML_To_Message

**MC_Message_To_XML_Native**

Converts a DICOM message, file or item into an XML string based on Native DICOM Model and passes the converted data to the user.

**Synopsis**

```c
#include "mc3msg.h"

MC_STATUS MC_Message_To_XML_Native
    (int MessageID,
    void *UserInfo,
    XML_OPTIONS XmlOptions,
    MC_STATUS (*YourReceiveXMLFunction)())
```

- **MessageID**
  The identifier of a message, file or item object.

- **UserInfo**
  Address of data which will be passed on to `YourReceiveXMLFunction` each time it is called. This may be NULL.

- **XmlOptions**
  Combination of the XML configuration flags that are defined in the XML_OPTIONS enumerator list:
  - XmlOptDefault
  - XmlOptIncludeBulks
  - XmlOptIncludePixelData
  - XmlOptExcludeSequences
  - XmlOptExcludePrivateAttributes
  - XmlOptBase64Binary
  - XmlOptIndentation

- **YourReceiveXMLFunction**
  Pointer to a function which will be called repeatedly by the toolkit to receive the XML string.

The function must be prototyped as follows:

```c
MC_STATUS YourReceiveXMLFunction
    (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_XML_Native function. This may be NULL.

CBdataSize: The number of bytes of XML data being provided to you in CBdataBuffer.

CBdataBuffer: The buffer containing XML data from the message object.

CBisFirst: Is TRUE (not zero) when Merge DICOM Toolkit is providing the first block of XML data.

CBisLast: Is TRUE (not zero) when Merge DICOM Toolkit is providing the last block of XML data.

Remarks

MC_Message_To_XML_Native requests that the contents of the message identified by MessageID be converted into an XML string based on Native DICOM Model. The converted message is passed to the user in blocks by calling YourReceiveXMLFunction repeatedly until the entire converted message has been transferred.

MC_Message_To_XML_Native can pass data to YourReceiveXMLFunction by specifying the data’s address in UserInfo. Merge DICOM Toolkit passes the address to YourReceiveXMLFunction in CBuserInfo each time it is called. UserInfo may be NULL.

XmlOptions specify the configuration flags for the DICOM to Native DICOM XML conversion. The enumerated XML_OPTIONS flags are defined in “mc3msg.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>XmlOptDefault</td>
<td>Use the default XML conversion settings.</td>
</tr>
<tr>
<td>XmlOptIncludeBulks</td>
<td>Store bulk attributes (VR is OB or OW) in the XML.</td>
</tr>
<tr>
<td>XmlOptIncludePixelData</td>
<td>Store Pixel Data buffer in the XML.</td>
</tr>
<tr>
<td>XmlOptExcludeSequences</td>
<td>Do not store Sequence attributes in the XML.</td>
</tr>
<tr>
<td>XmlOptExcludePrivateAttributes</td>
<td>Do not store Private attributes in the XML.</td>
</tr>
<tr>
<td>XmlOptBase64Binary</td>
<td>Use Base64 encoding for bulks and UN VR attributes.</td>
</tr>
<tr>
<td>XmlOptIndentation</td>
<td>Use default indentation in the generated XML.</td>
</tr>
</tbody>
</table>

YourReceiveXMLFunction

YourReceiveXMLFunction will be called repeatedly to pass blocks of data to it. Merge DICOM Toolkit sets CBisFirst to TRUE (non-zero) the first time it calls YourReceiveXMLFunction for this message and it sets CBisLast to TRUE (non-zero) when it calls YourReceiveXMLFunction with the final block of converted data.

CBdataBuffer is set to the address of a buffer containing the converted data block, and CBdataSize is set to the number of bytes placed at CBdataBuffer. The data buffer is valid only for the duration of the call to YourReceiveXMLFunction.
**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MC_NORMAL_COMPLETION</code></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><code>MC_NULL_POINTER_PARM</code></td>
<td>The YourReceiveXMLFunction parameter was NULL.</td>
</tr>
<tr>
<td><code>MC_INVALID_MESSAGE_ID</code></td>
<td>The MessageID value is not a valid message, file or item object ID.</td>
</tr>
<tr>
<td><code>MC_SYSTEM_ERROR</code></td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td><code>MC_MESSAGE_EMPTY</code></td>
<td>The message has no attributes in it.</td>
</tr>
<tr>
<td><code>MC_CALLBACK_CANNOT_COMPLY</code></td>
<td><code>YourReceiveXMLFunction</code> returned a value other than <code>MC_NORMAL_COMPLETION</code>.</td>
</tr>
<tr>
<td><code>MC_LIBRARY_NOT_INITIALIZED</code></td>
<td>This call was made prior to the initialization of the library.</td>
</tr>
</tbody>
</table>

**See Also**

`MC_XML_Native_To_Message`

### MC_NewProposedServiceList

**MC_NewProposedServiceListAsync**

Creates a new service list for use in association negotiation.

**Synopsis**

```c
#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 requestor. 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 Operations Invoked during association negotiation. 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 the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MC_NORMAL_COMPLETION</code></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><code>MC_SYSTEM_ERROR</code></td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td><code>MC_DUPLICATE_NAME</code></td>
<td>A Duplicate Name is present in the <code>ServiceNameArray</code> or the request service list name is already in use.</td>
</tr>
</tbody>
</table>

**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**

```c
#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_SYNTAX-es 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_SYNTAX-es are an enumerated type found in mc3msg.h.

Service names are generated using MC_NewServiceFromName or MC_NewServiceFromUID.

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_DUPLICATE_NAME</td>
<td>The requested syntax list name is already in use.</td>
</tr>
<tr>
<td>MC_DUPLICATE_SYNTAX</td>
<td>A syntax id is duplicated in the array supplied.</td>
</tr>
</tbody>
</table>

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

```c
#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 utilized 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.

© Copyright Merge Healthcare Solutions Inc. 2023
When creating a new service, it can be identified by its Merge DICOM Toolkit 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 negotiation information for the service.

The application assigned name is utilized by `MC_NewProposedServiceList`.

### Return Value

One of the enumerated `MC_STATUS` codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_INVALID_SOP_CLASS_UID</td>
<td>The SOPClassName or SOPClassUID do not reference a defined SOP in <code>mergecom.srv</code></td>
</tr>
</tbody>
</table>

### See Also

- `MC_NewSyntaxList`
- `MC_NewProposedServiceList`
- `MC_FreeService`
- `MC_Set_Negotiation_Information_For_Association`
- `MC_Open_Association`
- `MC_Open_Association_With_Connect_Timeout`
- `MC_Open_Association_With_Identity`
- `MC_Open_Association_With_Identity_With_Reject_Info`
- `MC_Open_Association_With_Identity_With_Reject_Info_With_Connect_Timeout`
- `MC_Open_Association_With_Reject_Info`
- `MC_Open_Association_With_Callback`
- `MC_Open_Association_With_AllOptional_Parameters`

Establishes a connection with a remote DICOM application.

### Synopsis

```c
#include "mergecom.h"

MC_STATUS MC_Open_Association (  
    int ApplicationID,  
    int *AssociationID,  
    const char *RemoteApplicationTitle,  
    int *RemoteHostPortNumber,  
    char *RemoteHostTCPName,  
)```

© Copyright Merge Healthcare Solutions Inc. 2023
typedef enum {
    NO_USER_IDENTITY = 0,
    USERNAME = 1,
    USERNAME_AND_PASSCODE = 2,
    KERBEROS_SERVICE_TICKET = 3,
    SAML_ASSERTION = 4
} USER_IDENTITY_TYPE;

MC_STATUS MC_Open_Association_With_Identity (  
    int ApplicationID,
    int *AssociationID,
    const char *RemoteApplicationTitle,
    int *RemoteHostPortNumber,
    char *RemoteHostTCPIPName,
    char *ServiceList,
    SecureSocketFunctions *SecureFunctions,
    void *SecureContext,
    USER_IDENTITY_TYPE IdentityType,
    unsigned char ResponseRequested,
    void *PrimaryField,
    unsigned short PrimaryFieldLength,
    void *SecondaryField,
    unsigned short SecondaryFieldLength
)

typedef struct {  
    unsigned short Result;
    unsigned short Source;
    unsigned short Reason;
} AssocRejectInfo;

MC_STATUS MC_Open_Association_With_Identity_With_Reject_Info (  
    int ApplicationID,
    int *AssociationID,
    const char *RemoteApplicationTitle,
    int *RemoteHostPortNumber,
    char *RemoteHostTCPIPName,
    char *ServiceList,
    SecureSocketFunctions *SecureFunctions,
    void *SecureContext,
    USER_IDENTITY_TYPE IdentityType,
unsigned char ResponseRequested,
void *PrimaryField,
unsigned short PrimaryFieldLength,
void *SecondaryField,
unsigned short SecondaryFieldLength,
AssocRejectInfo RejectInfo
)
MC_STATUS
MC_Open_Association_With_Identity_With_Reject_Info_With_Connect_Timeout (  
int ApplicationID,
int *AssociationID,
const char *RemoteApplicationTitle,
int *RemoteHostPortNumber,
char *RemoteHostTCPIPName,
char *ServiceList,
SecureSocketFunctions *SecureFunctions,
void *SecureContext,
USER_IDENTITY_TYPE IdentityType,
unsigned char ResponseRequested,
void *PrimaryField,
unsigned short PrimaryFieldLength,
void *SecondaryField,
unsigned short SecondaryFieldLength,
AssocRejectInfo RejectInfo,
int Timeout
)

MC_STATUS MC_Open_Association_With_Reject_Info (  
int ApplicationID,
int *AssociationID,
const char *RemoteApplicationTitle,
int *RemoteHostPortNumber,
char *RemoteHostTCPIPName,
char *ServiceList
AssocRejectInfo *RejectInfo
)

MC_STATUS MC_Open_Association_With_Callback (  
int ApplicationID,
int *AssociationID,
const char *RemoteApplicationTitle,
int *RemoteHostPortNumber,
char *RemoteHostTCPIPName,
char *ServiceList,
MC_STATUS (*YourCallback) ()
)

MC_STATUS MC_Open_Association_With_AllOptional_Parameters (  
int ApplicationID,
int *AssociationID,
const char *RemoteApplicationTitle,
int *RemoteHostPortNumber,
char *RemoteHostTCPIPName,
char *ServiceList,
SecureSocketFunctions *SecureFunctions,
void *SecureContext,
USER_IDENTITY_TYPE IdentityType,
unsigned char ResponseRequested,
void *PrimaryField,
unsigned short PrimaryFieldLength,
void *SecondaryField,
unsigned short SecondaryFieldLength,
AssocRejectInfo RejectInfo,
int Timeout,
MC_STATUS (*YourCallback) ()
)

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 parameters 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. This parameter
can be a hostname, an IPv4 address, or an IPv6 address.

ServiceList Name of a service list in the Merge DICOM Toolkit
configuration file.

SecureFunctions An optional pointer to a structure containing
functions that will be called by Merge DICOM
Toolkit while processing network I/O over the
secure connection. See
MC_Open_Secure_Association for details.

SecureContext An optional pointer to application-specific data that
Merge DICOM Toolkit passes to the functions
declared in SecureFunctions. See
MC_Open_Secure_Association for details.

IdentityType The type of User Identity negotiation to perform as
specified in DICOM PS3.7, Section D.3.3.7.

ResponseRequested If response from the server is required, this field
should be set to
POSITIVE_RESPONSE_REQUESTED, if a response is
not required, this field should be set to
NO_RESPONSE_REQUESTED.

PrimaryField A buffer pointing to the primary field used in User
Identity Negotiation.
PrimaryFieldLength
The length of PrimaryField. Note that when passing a text string in PrimaryField, the NULL terminator should not be included in the length.

SecondaryField
A buffer pointing to the secondary field used in User Identity Negotiation. If the secondary field is not used for the type of User Identity specified by IdentityType, this field can be set to NULL.

SecondaryFieldLength
The length of SecondaryField. Note that when passing a text string in SecondaryField, the NULL terminator should not be included in the length.

RejectInfo
If the association is rejected, this return argument will contain the result/source/reason codes explaining the reason for rejection (See DICOM PS3.8, Section 9.3.4).

Timeout
The number of seconds to wait for the network connect to be accepted.

If greater than zero, the parameter supersedes the application-wide CONNECT_TIMEOUT configuration setting.

If negative or zero, the CONNECT_TIMEOUT configuration setting takes effect.

YourCallback
Name of a user function which will be called with the association ID as a parameter

The callback function must be prototyped as follows:

```c
MC_STATUS YourCallback ( 
  int AssociationID 
)
```

AssociationID
The ID assigned to the association object as soon as it is created. This ID can be used to cancel the association request before the connect timeout expires.

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 RemoteHostTCP/IPName.

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 Merge DICOM Toolkit 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_Open_Association_With_Identity is used when User Identity information is to be negotiated for the association as defined in DICOM Supplement 99. MC_Open_Association_With_Identity can be used to open both secure and unsecure associations. See MC_Open_Secure_Association for details on opening secure associations.

When negotiating user identity information, the IdentityType parameter is set to a specific kind of user identity. The PrimaryField and SecondaryField parameters are set as defined in DICOM Part 8. For instance, when USERNAME_AND_PASSCODE is specified, the PrimaryField has the username and SecondaryField has the passcode. Due to the fact that older DICOM applications may not be looking for the user identity information, it is possible to ask for a response from the server so the application can determine if the user identity information was accepted or ignored. The ResponseRequested field is used to request this response. After the association has been negotiated, an application can use the MC_Get_Association_Information to retrieve the information on if the server has responded to the user identity information in the request. The MC_Get_User_Identity_Length and MC_Get_User_Identity_Info routines can also be used if information was returned from the remote system.

**NOTE:** MC_Open_Association_With_Identity is identical to MC_Open_Association when IdentityType is set to NO_USER_IDENTITY, and the SecureFunctions, PrimaryField, and SecondaryField parameters are set to NULL.

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) for `MC_Open_Association` 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 Merge DICOM Toolkit Application Configuration file.

**Return Value**

One of the enumerated `MC_STATUS` codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally. AssociationID has been set to the association's identification number.</td>
</tr>
<tr>
<td>MC_ASSOCIATION_REJECTED</td>
<td>The remote system rejected the open association request.</td>
</tr>
<tr>
<td>MC_NEGOTIATION_ABORTED</td>
<td>The association was aborted during negotiation</td>
</tr>
<tr>
<td>MC_CONFIG_INFO_MISSING</td>
<td>RemoteApplicationTitle or ServiceList could not be found in the mergecom.app configuration file.</td>
</tr>
<tr>
<td>MC_CONFIG_INFO_ERROR</td>
<td>The service list contained too many services. DICOM has a limit of 128 presentation contexts/services when negotiating an association.</td>
</tr>
<tr>
<td>MC_CONNECTION_FAILED</td>
<td>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.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>AssociationID and/or RemoteApplicationTitle was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_APPLICATION_TITLE</td>
<td>RemoteApplicationTitle was not 1-16 bytes long.</td>
</tr>
<tr>
<td>MC_INVALID_APPLICATION_ID</td>
<td>ApplicationID does not identify a valid Merge DICOM Toolkit application.</td>
</tr>
<tr>
<td>MC_INVALID_PORT_NUMBER</td>
<td>RemoteHostPortNumber was not NULL and pointed at a negative number.</td>
</tr>
<tr>
<td>MC_INVALID_SERVICE_LIST_NAME</td>
<td>ServiceList was not NULL and pointed at an empty string.</td>
</tr>
<tr>
<td>MC_INVALID_HOST_NAME</td>
<td>RemoteHostTCPIName was not NULL and it was not 1-39 bytes long.</td>
</tr>
<tr>
<td>MC_UNKNOWN_HOST_NAME</td>
<td>The RemoteHostTCPIName is unknown to the system.</td>
</tr>
<tr>
<td>MC_TIMEOUT</td>
<td>Timeout attempting the association.</td>
</tr>
<tr>
<td>MC_SYSTEM_CALL_INTERRUPTED</td>
<td>The operating system interrupted the network call. Retry the connection.</td>
</tr>
<tr>
<td>MC_NO_APPLICATIONS_REGISTERED</td>
<td>An application title for this application has not yet be registered.</td>
</tr>
<tr>
<td>MC_LIBRARY_NOT_INITIALIZED</td>
<td>This call was made prior to the initialization of the library</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message</td>
</tr>
</tbody>
</table>
See Also

MC_Register_Application  MC_Open_Secure_Association
MC_Wait_For_Association  MC_Wait_For_Secure_Association
MC_Get_Association_Info  MC_Set_Negotiation_Info
MC_Get_Negotiation_Info  MC_Get_User_Identity_Length
MC_Get_User_Identity_Info  MC_Clear_Negotiation_Info
MC_Close_Association  MC_Abort_Association

MC_Open_Empty_Item

Creates a new empty Merge DICOM Toolkit item object.

Synopsis

#include "mc3msg.h"

MC_STATUS MC_Open_Empty_Item (  
  int *ItemIDPtr
)

ItemIDPtr Upon successful completion, the item object identifier is returned here.

Remarks

The MC_Open_Empty_Item function creates a empty “item object” which contains no attributes. 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.

If an item is opened using MC_Open_Empty_Item, it is not necessary to add attributes to the item 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 item object before the value is set. (Note that this is NOT THE CASE if an item object is opened using MC_Open_Item. In that case the item IS associated with a given item name and attributes other than those associated with that item name must be explicitly added to the item before setting values for the added attributes.)

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>*ItemIDPtr was NULL.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>
See Also
MC_Open_Item

MC_Open_Empty_Message
Creates a new “empty” Merge DICOM Toolkit 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” for more information about these commands.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>MessageIDPtr was NULL.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the</td>
</tr>
<tr>
<td></td>
<td>operating environment. A message describing the error has been written</td>
</tr>
<tr>
<td></td>
<td>to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>
See Also

MC_Open_Message MC_Create_File
MC_Free_Message MC_Free_Item

MC_Open_File
MC_Open_File_Bypass_OBOW
MC_Open_File_Upto_Tag
MC_Open_File_Upto_Tag_Bypass_Value

Requests that the values of a file object be retrieved from a DICOM file

Synopsis

```c
#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())
)

MC_STATUS MC_Open_File_Upto_Tag_Bypass_Value (  
    int ApplicationID,
    int FileID,
    void *UserInfo,
    unsigned long Tag,
    MTI_BOOLEAN AbypassTagValue,
    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_CreateEmpty_File` functions.

**UserInfo**
Address of data which will be passed on to `YourFromMediaFunction` each time it is called. This may be NULL.

**AbypassTagValue**
Bypass an attribute value if TRUE.

**Note:**
For `MC_Open_File_Upto_Tag_Bypass_Value` this argument is unused, as the function will *by definition* bypass the attribute value. Retained for backward compatibility only.

**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 of:
- the first attribute greater than Tag (for `MC_Open_File_Upto_Tag`)
- the data of the attribute at Tag (for `MC_Open_File_Upto_Tag_Bypass_Value`) 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:

```c
MC_STATUS YourFromMediaFunction ( 
    char *CBfilename, 
    void *CBuserInfo, 
    int *CBdataSizePtr, 
    void **CBdataBufferPtr, 
    int CBisFirst, 
    int *CBisLastPtr 
)
```

- **CBfilename**
  String filename associated with the 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 Merge DICOM Toolkit calls `YourFromMediaFunction` to request data blocks.
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**, **MC_Open_File_Upto_Tag_Bypass_Value** request that the contents of a DICOM file be converted and placed into the file object `FileID`. The file is passed to Merge DICOM Toolkit by **YourFromMediaFunction**. Merge DICOM Toolkit 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`. Merge DICOM Toolkit 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, OF or OD 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, OF or OD 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.

**MC_Open_File_Bypass_OBOW** can be used to increase performance for handling attributes of type OB, OW, OF or OD. When a callback function is registered with **MC_Register_Callback_Function** for an attribute of type OB, OW, OF or OD, **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 Merge DICOM Toolkit, 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, OF or OD 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.

**MC_Open_File_Upto_Tag** can be used to increase performance for handling attributes of type OB, OW, OF or OD. **MC_Open_File_Upto_Tag** will stop reading the file from media when it reaches the first attribute greater than `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.

**MC_Open_File_Upto_Tag_Bypass_Value** has the same functionality as **MC_Open_File_Upto_Tag** except that the attribute values with given `Tag` will be read, but left on media. It increases performance for handling large size attributes, especially pixel data (\(^{7fe0,0010}\)). Instead of reading the attribute values the above described callback mechanism should be used with **MC_Register_Callback_Function** for that attribute to be retrieved from media when needed. If the file contains any attributes with a value representation of SQ (i.e. the file contains one or more sequences 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 the item in the sequence attribute. Later, the `MC_Get_Value...` functions may be used to retrieve the ItemIDs 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 will also 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 is 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 entire 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 VRs 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 Merge DICOM Toolkit each time it is called. Merge DICOM Toolkit 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.
Data in the buffer returned by "CBdataBufferPtr" shall not be reused, because with some configurations Merge DICOM Toolkit may do in-place byte swapping to convert data to a native endian format. If your application needs to keep source data unchanged, you must create a copy of your source data chunk and return its address to the caller.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>FileIDPtr, YourFromMediaFunction, or Filename was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_APPLICATION_ID</td>
<td>The application ID does not identify a valid Merge DICOM Toolkit application.</td>
</tr>
<tr>
<td>MC_INVALID_FILE</td>
<td>An invalid DICOM Prefix was found within the file. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_INVALID_TRANSFER_SYNTAX</td>
<td>An invalid transfer syntax code was found within the file's DICOM File Meta Information.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_CALLBACK_CANNOT_COMPLY</td>
<td>YourFromMediaFunction returned with a status other than MC_NORMAL_COMPLETION.</td>
</tr>
<tr>
<td>MC_CALLBACK_DATA_SIZE_UNEVEN</td>
<td>The CBdataSizePtr parameter returned by YourFromMediaFunction was an uneven number.</td>
</tr>
<tr>
<td>MC_CALLBACK_PARM_ERROR</td>
<td>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.</td>
</tr>
<tr>
<td>MC_OUT_OF_ORDER_TAG</td>
<td>A tag was found in the file that was not in ascending order. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_INVALID_TAG</td>
<td>The file contains an invalid tag.</td>
</tr>
<tr>
<td>MC_VALUE_TOO_LARGE</td>
<td>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.</td>
</tr>
<tr>
<td>MC_UNEXPECTED_EOD</td>
<td>YourFromMediaFunction stopped without passing the entire value for an attribute.</td>
</tr>
<tr>
<td>MC_INVALID_LENGTH_FOR_VR</td>
<td>The value(s) for one of the file attributes was not legal for its value representation.</td>
</tr>
</tbody>
</table>

Any of the status codes returned by **MC_Set_Value** may also be returned.
See Also

MC_Register_Callback_Function  MC_Write_File  MC_Write_File_By_Callback

**MC_Open_Item**

Creates a new Merge DICOM Toolkit item object.

**Synopsis**

```c
#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 defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_MSGFILE_ERROR</td>
<td>An error occurred attempting to access the configuration data file.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

See Also

MC_Open_Empty_Item  MC_Open_Message
MC_Open_Message

Creates a new Merge DICOM Toolkit message object.

Synopsis

```c
#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 Merge DICOM Toolkit 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 Merge DICOM Toolkit:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C_STORE_RQ</td>
<td>DICOM Composite Store Service Request</td>
</tr>
<tr>
<td>C_STORE_RSP</td>
<td>DICOM Composite Store Service Response</td>
</tr>
<tr>
<td>C_ECHO_RQ</td>
<td>DICOM Verification Service Request</td>
</tr>
<tr>
<td>C_ECHO_RSP</td>
<td>DICOM Verification Service Response</td>
</tr>
</tbody>
</table>
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 the enumerated **MC_STATUS** codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_NULL_POINTER_PARM</strong></td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td><strong>MC_INVALID_COMMAND</strong></td>
<td><em>Command</em> is not a supported command.</td>
</tr>
<tr>
<td><strong>MC_MSGFILE_ERROR</strong></td>
<td>An error occurred attempting to access the configuration data file.</td>
</tr>
</tbody>
</table>
MC_SYSTEM_ERROR
An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.

See Also
MC_Open_Empty_Message  MC_Free_Message
MC_Free_Item

MC_Open_Secure_Association MC_Open_Secure_Association_With_Reject_Info
Establishes a connection with a remote DICOM application over a secure socket connection.

Synopsis

```c
#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
)

typedef struct { 
    unsigned short Result;
    unsigned short Source;
    unsigned short Reason;
} AssocRejectInfo;

MC_STATUS MC_Open_Secure_Association_With_Reject_Info ( 
    int ApplicationID,
    int *AssociationID,
    const char *RemoteApplicationTitle,
    int *RemoteHostPortNumber,
    char *RemoteHostTCPIPName,
    char *ServiceList,
    SecureSocketFunctions *SS_functions,
    void *SS_application_context,
    AssocRejectInfo RejectInfo
)
```

**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 Merge DICOM Toolkit while processing network I/O over the secure connection. (see below)

Each of the following is optional. Use NULL if not used

RejectInfo If the association is rejected, this return argument will contain the result/source/reason codes explaining the reason for rejection (See DICOM PS3.8, Section 9.3.4).

RemoteHostPortNumber The TCP/IP port used by the remote application to “listen” for DICOM associations.

RemoteHostTCP/IPName The remote host’s TCP/IP Name. This parameter can be a hostname, an IPv4 address, or an IPv6 address.

ServiceList Name of a service list in the Merge DICOM Toolkit configuration file.

SS_context An optional pointer to application-specific data that Merge DICOM Toolkit 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 RemoteHostTCP/IPName.

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 Merge DICOM Toolkit 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, RemoteHostTCPName and ServiceList* will be defaulted if they are specified as NULL. The default values are obtained from the *RemoteApplicationTitle*’s section in the Merge DICOM Toolkit Application Configuration file.

**SecureSocketFunctions**

When using the **MC_Open_Secure_Association** call, Merge DICOM Toolkit 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). Merge DICOM Toolkit 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:

```c
typedef struct MC_Secure_Socket_Functions_Struct {
    SS_STATUS (NOEXP_FUNC *SS_Session_Start) (MC_SOCKET SocketToUse, CONN_TYPE ConnectionType, void *ApplicationContext, void **SecurityContext);

    SS_STATUS (NOEXP_FUNC *SS_Read) (void *SScontext, void *ApplicationContext, char *Buffer, unsigned int BytesToRead, unsigned int *BytesRead, int Timeout);

    SS_STATUS (NOEXP_FUNC *SS_Write) (void *SScontext, void *ApplicationContext, char *Buffer,
```
unsigned int BytesToWrite,
unsigned 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**

Merge DICOM Toolkit 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. Merge DICOM Toolkit 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**

Merge DICOM Toolkit 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. Merge DICOM Toolkit 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**

Merge DICOM Toolkit 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 Merge DICOM Toolkit 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**

Merge DICOM Toolkit 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 Merge DICOM Toolkit 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, MC_Send_Request, MC_Send_Response_Message or MC_Send_Response call will return MC_SYSTEM_ERROR.)

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally. <em>AssociationID</em> has been set to the association's identification number.</td>
</tr>
<tr>
<td>MC_ASSOCIATION_REJECTED</td>
<td>The remote system rejected the open association request.</td>
</tr>
<tr>
<td>MC_NEGOTIATION_ABORTED</td>
<td>The association was aborted during negotiation</td>
</tr>
<tr>
<td>MC_CONNECTION_FAILED</td>
<td>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.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>*AssociationID, SS_functions or RemoteApplicationTitle was NULL.</td>
</tr>
</tbody>
</table>
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 Merge DICOM
Toolkit 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 Merge
DICOM Toolkit 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_Wait_For_Connection
MC_Wait_For_Connection_On_Port MC_Process_Association_Request
MC_Process_Secure_Association_Request

MC_Process_Association_Request MC_Process_Secure_Association_Request

Process a standard association request or secure association request that was received by the
MC_Wait_For_Connection or MC_Wait_For_Connection_on_Port functions. It also can be used to
process an association received by the internet services daemon (inetd) on UNIX systems.

Synopsis
#include “mergecom.h”
MC_STATUS MC_Process_Association_Request (  
    MC_SOCKET Socket,
    const char *ServiceListName,
    int *ApplicationID,
    int *AssociationID
)
MC_STATUS MC_Process_Secure_Association_Request (  
    MC_SOCKET Socket,  
    const char *ServiceListName,  
    int *ApplicationID,  
    int *AssociationID,  
    SecureSocketFunctions *SS_functions,  
    void *SS_application_context  
  )

Socket  Socket returned by MC_Wait_For_Connection or  
         MC_Wait_For_Connection_On_Port.

ServiceListName  Name of a service list in the Merge DICOM Toolkit  
                 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.

SS_functions  Pointer to a structure containing functions that will be  
               called by Merge DICOM Toolkit while processing  
               network I/O over the secure connection.  
               The SecureSocketFunctions structure is defined in  
               “mergecom.h” (see MC_Open_Secure_Association)  

SS_context  An optional pointer to application-specific data that  
             Merge DICOM Toolkit passes to the functions declared  
             in SS_functions.

Remarks

MC_Process_Association_Request can be used in conjunction with the MC_Wait_For_Connection  
and MC_Wait_For_Connection_On_Port routines to handle incoming connections.

MC_Process_Secure_Association_Request can also be used in conjunction with  
MC_Wait_For_Connection and MC_Wait_For_Connection_On_Port to process secure DICOM  
association requests. See the description to MC_Wait_For_Secure_Association for details on how to  
use secure callback functions.

MC_Process_Association_Request and MC_Process_Secure_Association_Request can also 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. inetd supplies the socket to the process by setting stdin to the socket’s  
value. In this scenario, stdin must be used as the Socket parameter to  
MC_Process_Association_Request.

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
    "/usr/bin/inetd_echo_scp" - argv[0]
    "/usr/bin/merge.ini" - argv[1] - The application's arguments, if any.

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCNORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_TIMEOUT</td>
<td>The Timeout period expired.</td>
</tr>
<tr>
<td>MCNEGOTIATION_ABORTED</td>
<td>An incoming association was aborted during negotiation. Normally this situation is handled by retrying the MC_Process_Association_Request call.</td>
</tr>
<tr>
<td>MCSV_SYSTEM_CALL_INTERRUPTED</td>
<td>The system call used to wait for an association request was interrupted by a signal. Normally this situation is handled by retrying the MC_Process_Association_Request call.</td>
</tr>
<tr>
<td>MCNULL_POINTER_PARM</td>
<td>AssociationID, ApplicationID or ServiceList parameter was NULL.</td>
</tr>
<tr>
<td>MCNO_APPLICATIONS_REGISTERED</td>
<td>No applications have been registered using MC_Register_Application.</td>
</tr>
</tbody>
</table>
MC_INVALID_SERVICE_LIST_NAME

ServiceList points at a null string.

MC_SYSTEM_ERROR

An unexpected, potentially serious problem was detected in the operating environment. A message describing the error had been written to the Merge DICOM Toolkit log file.

MC_ASSOCIATION_REJECTED

The association was rejected as invalid.

MC_ASSOCIATION_CLOSED

The association was closed after handling an echo request.

See Also

MC_Wait_For_Association  MC_Wait_For_Secure_Association
MC_Wait_For_Connection  MC_Wait_For_Connection_On_Port

MC_Read_Message

Reads the next message sent by 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 maximum 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: 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 Merge DICOM Toolkit:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C_STORE_RQ</td>
<td>DICOM Composite Store Service Request</td>
</tr>
<tr>
<td>C_STORE_RSP</td>
<td>DICOM Composite Store Service Response</td>
</tr>
<tr>
<td>C_ECHO_RQ</td>
<td>DICOM Verification Service Request</td>
</tr>
<tr>
<td>C_ECHO_RSP</td>
<td>DICOM Verification Service Response</td>
</tr>
<tr>
<td>C_FIND_RQ</td>
<td>DICOM Composite Find Service Request</td>
</tr>
<tr>
<td>C_FIND_RSP</td>
<td>DICOM Composite Find Service Response</td>
</tr>
<tr>
<td>C_CANCEL_FIND_RQ</td>
<td>Cancel DICOM Composite Find Service Request</td>
</tr>
<tr>
<td>C_GET_RQ</td>
<td>DICOM Composite Get Service Request</td>
</tr>
<tr>
<td>C_GET_RSP</td>
<td>DICOM Composite Get Service Response</td>
</tr>
<tr>
<td>C_CANCEL_GET_RQ</td>
<td>Cancel DICOM Composite Get Service Request</td>
</tr>
<tr>
<td>C_MOVE_RQ</td>
<td>DICOM Composite Move Service Request</td>
</tr>
<tr>
<td>C_MOVE_RSP</td>
<td>DICOM Composite Move Service Response</td>
</tr>
</tbody>
</table>
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 the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_TIMEOUT</td>
<td>The Timeout period expired.</td>
</tr>
<tr>
<td>MC_MUST_CONTINUE_BEFORE_READING</td>
<td>A previous message was not read completely before making this call.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>The MessageID, ServiceName or Command parameter was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_ASSOC_ID</td>
<td>AssociationID is not a valid association object ID.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NETWORK_SHUT_DOWN</td>
<td>The network connect unexpectedly dropped.</td>
</tr>
<tr>
<td>MC_ASSOCIATION_ABORTED</td>
<td>The association has been aborted. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_ASSOCIATION_CLOSED</td>
<td>The association has been closed.</td>
</tr>
<tr>
<td>MC_INACTIVITY_TIMEOUT</td>
<td>A timeout occurred in the middle of receiving a message.</td>
</tr>
<tr>
<td>MC_CONFIG_INFO_ERROR</td>
<td>The message information file describing the message’s service/command pair could not be accessed.</td>
</tr>
</tbody>
</table>
**MC_INVALID_MESSAGE_RECEIVED** An improperly formatted DICOM message was received. A message describing the error has been written to the Merge DICOM Toolkit log file.

**See Also**
- MC_Read_Message
- MC_Free_Message
- MC_Register_Callback_Function
- MC_Continue_Read_Message
- MC_Continue_Read_Message_To_Tag

**MC_Read_Message_To_Tag**

Reads the next message sent by the remote applicate up until a specified tag.

**Synopsis**

```c
#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 maximum 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 Merge DICOM Toolkit:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C_STORE_RQ</td>
<td>DICOM Composite Store Service Request</td>
</tr>
<tr>
<td>C_STORE_RSP</td>
<td>DICOM Composite Store Service Response</td>
</tr>
<tr>
<td>C_ECHO_RQ</td>
<td>DICOM Verification Service Request</td>
</tr>
<tr>
<td>C_ECHO_RSP</td>
<td>DICOM Verification Service Response</td>
</tr>
<tr>
<td>C_FIND_RQ</td>
<td>DICOM Composite Find Service Request</td>
</tr>
<tr>
<td>C_FIND_RSP</td>
<td>DICOM Composite Find Service Response</td>
</tr>
<tr>
<td>C_CANCEL_FIND_RQ</td>
<td>Cancel DICOM Composite Find Service Request</td>
</tr>
<tr>
<td>C_GET_RQ</td>
<td>DICOM Composite Get Service Request</td>
</tr>
<tr>
<td>C_GET_RSP</td>
<td>DICOM Composite Get Service Response</td>
</tr>
<tr>
<td>C_CANCEL_GET_RQ</td>
<td>Cancel DICOM Composite Get Service Request</td>
</tr>
</tbody>
</table>
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 the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_MUST_CONTINUE_BEFORE_READING</td>
<td>A previous message was not read completely before making this call.</td>
</tr>
<tr>
<td>MC_TIMEOUT</td>
<td>The Timeout period expired.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>The MessageID, ServiceName or Command parameter was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_ASSOC_ID</td>
<td>AssociationID is not a valid association object ID.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_INVALID_TAG</td>
<td>StopTag was within the command set (less then (0001,0000)).</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NETWORK_SHUT_DOWN</td>
<td>The network connect unexpectedly dropped.</td>
</tr>
<tr>
<td>MC_ASSOCIATION_ABORTED</td>
<td>The association has been aborted. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>
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 Merge DICOM Toolkit log file.

See Also
MC_Read_Message  MC_Register_Callback_Function
MC_Continue_Read_Message  MC_Free_Message
MC_Continue_Read_Message_To_Tag

MC_Read_To_Stream
Reads the message sent by the remote application directly into a streaming function. Especially useful for the transfer of large data which we don’t want to load into application memory.

Synopsis
#include "mergecom.h"
MC_STATUS MC_Read_To_Stream (  
    int AssociationID,  
    int Timeout,  
    char **ServiceName,  
    MC_COMMAND *Command  
    char *FileName,  
    void *UserInfo,  
    MC_STATUS (*YourReceiveStreamFunction) ()
)

AssociationID  The association object’s identification number.
Timeout  The maximum 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”.
ServiceName  The name of the service is returned here.
Command  The command which is associated with the message is returned here. The MC_COMMAND enumerated values are defined in “mc3msg.h”.
Filename  The name of file associated with the receiving message. This may be NULL.
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 receive blocks of streamed DICOM message data.

The function must be prototyped as follows:

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

- **CBmessageID**: The internal identifier assigned to the receiving message object. It becomes invalid once the message is received.
- **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 Merge DICOM Toolkit is providing the first block of stream data.
- **CBisLast**: Is TRUE (not zero) when Merge DICOM Toolkit is providing the last block of stream data.

### Remarks

**MC_Read_To_Stream** 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:** If `Timeout` is set to less than zero, **MC_Read_To_Stream** 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, `YourReceiveStreamFunction` is called repeatedly providing the blocks of data of a message object. The message's attribute values have been set by the DICOM message stream received from the remote application.

**ServiceName** is set to the name of the service associated with receiving message and **Command** is set to the command associated with `MessageID`.

If **MC_Read_To_Stream** returns either **MC_ASSOCIATION_CLOSED**, **MC_ASSOCIATION_ABORTED**, **MC_NETWORK_SHUT_DOWN**, or **MC_INACTIVITY_TIMEOUT**, **MC_TIMEOUT**, **MC_OPERATION_TIMEOUT**, or **MC_INACTIVE_TIMEOUT**, the calling user can retry the request or stop the application as appropriate.
**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 Merge DICOM Toolkit:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C_STORE_RQ</td>
<td>DICOM Composite Store Service Request</td>
</tr>
<tr>
<td>C_STORE_RSP</td>
<td>DICOM Composite Store Service Response</td>
</tr>
<tr>
<td>C_ECHO_RQ</td>
<td>DICOM Verification Service Request</td>
</tr>
<tr>
<td>C_ECHO_RSP</td>
<td>DICOM Verification Service Response</td>
</tr>
<tr>
<td>C_FIND_RQ</td>
<td>DICOM Composite Find Service Request</td>
</tr>
<tr>
<td>C_FIND_RSP</td>
<td>DICOM Composite Find Service Response</td>
</tr>
<tr>
<td>C_CANCEL_FIND_RQ</td>
<td>Cancel DICOM Composite Find Service Request</td>
</tr>
<tr>
<td>C_GET_RQ</td>
<td>DICOM Composite Get Service Request</td>
</tr>
<tr>
<td>C_GET_RSP</td>
<td>DICOM Composite Get Service Response</td>
</tr>
<tr>
<td>C_CANCEL_GET_RQ</td>
<td>Cancel DICOM Composite Get Service Request</td>
</tr>
<tr>
<td>C_MOVE_RQ</td>
<td>DICOM Composite Move Service Request</td>
</tr>
<tr>
<td>C_MOVE_RSP</td>
<td>DICOM Composite Move Service Response</td>
</tr>
<tr>
<td>C_CANCEL_MOVE_RQ</td>
<td>Cancel DICOM Composite Move Service Request</td>
</tr>
<tr>
<td>N_EVENT_REPORT_RQ</td>
<td>DICOM Normalized Report Service Request</td>
</tr>
<tr>
<td>N_EVENT_REPORT_RSP</td>
<td>DICOM Normalized Report Service Response</td>
</tr>
<tr>
<td>N_GET_RQ</td>
<td>DICOM Normalized Get Service Request</td>
</tr>
<tr>
<td>N_GET_RSP</td>
<td>DICOM Normalized Get Service Response</td>
</tr>
<tr>
<td>N_SET_RQ</td>
<td>DICOM Normalized Set Service Request</td>
</tr>
<tr>
<td>N_SET_RSP</td>
<td>DICOM Normalized Set Service Response</td>
</tr>
<tr>
<td>N_ACTION_RQ</td>
<td>DICOM Normalized Action Service Request</td>
</tr>
<tr>
<td>N_ACTION_RSP</td>
<td>DICOM Normalized Action Service Response</td>
</tr>
<tr>
<td>N_CREATE_RQ</td>
<td>DICOM Normalized Create Service Request</td>
</tr>
<tr>
<td>N_CREATE_RSP</td>
<td>DICOM Normalized Create Service Response</td>
</tr>
<tr>
<td>N_DELETE_RQ</td>
<td>DICOM Normalized Delete Service Request</td>
</tr>
<tr>
<td>N_DELETE_RSP</td>
<td>DICOM Normalized Delete Service Response</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_TIMEOUT</strong></td>
<td>The <em>Timeout</em> period expired.</td>
</tr>
<tr>
<td><strong>MC_MUST_CONTINUE_BEFORE_READING</strong></td>
<td>A previous message was not read completely before making this call.</td>
</tr>
<tr>
<td><strong>MC_NULL_POINTER_PARM</strong></td>
<td>The <em>MessageID</em>, <em>ServiceName</em> or <em>Command</em> parameter was NULL.</td>
</tr>
<tr>
<td><strong>MC_INVALID_ASSOC_ID</strong></td>
<td><em>AssociationID</em> is not a valid association object ID.</td>
</tr>
<tr>
<td><strong>MC_SYSTEM_ERROR</strong></td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

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 Merge DICOM Toolkit 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 Merge DICOM Toolkit log file. |

**See Also**

MC_Read_Message  
MC_Continue_Read_Message  
MC_Continue_Read_Message_To_Tag

**MC_Register_Application**

Registers a Merge DICOM Toolkit application.

**Synopsis**

```c
#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 Merge DICOM Toolkit. This function provides Merge DICOM Toolkit 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 the enumerated **MC_STATUS** codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_NULL_POINTER_PARM</strong></td>
<td><code>ApplicationID</code> or <code>AEtitle</code> was NULL.</td>
</tr>
<tr>
<td><strong>MC_INVALID_LENGTH_FOR_TITLE</strong></td>
<td><code>AEtitle</code> must be 1 to 16 bytes long.</td>
</tr>
<tr>
<td><strong>MC_ALREADY_REGISTERED</strong></td>
<td>An application title with this name has already been registered.</td>
</tr>
<tr>
<td><strong>MC_SYSTEM_ERROR</strong></td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**See Also**

<table>
<thead>
<tr>
<th>MC_Release_Application</th>
<th>MC_Register_Callback_Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_Set_Message_Callbacks</td>
<td>MC_Set_Negotiation_Info</td>
</tr>
</tbody>
</table>

**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**

```c
#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, OD, 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**  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:

```c
MC_STATUS CallbackFunction (  
    int CBmessageID,  
    unsigned long CBtag,  
    void *CBuserInfo,  
    CALLBACK_TYPE CBTtype,  
    unsigned long *CBdataSizePtr,  
    void **CBdataBufferPtr,  
    int CBisFirst,  
    int *CBisLastPtr  
)
```

```c
MC_STATUS pCallbackFunction (  
    int CBmessageID,  
    char *CBprivateCode,  
    unsigned short CBgroup,  
    unsigned char CBelementByte,  
    void *CBuserInfo,  
    CALLBACK_TYPE CBTtype,  
    unsigned long *CBdataSizePtr,  
    void **CBdataBufferPtr,  
    int CBisFirst,  
    int *CBisLastPtr  
)
```

**CBmessageID**  Merge DICOM Toolkit sets this to the identifier assigned to the message, item or file object by the `MC_Open_Message` function.

**CBtag**  Merge DICOM Toolkit 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** | Merge DICOM Toolkit 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 |
| | FREE_DATA |
| | REQUEST_FOR_DATA_WITH_OFFSET |
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 REQUEST_FOR_DATA_WITH_OFFSET, then "CbdataSizePtr" is the pointer to long which contains the offset of required data in the input seekable stream. The stream offset has to be used only when CBisFirst to TRUE (not zero), i.e. when you are to provide the first block of data. After the data reading "CbdataSizePtr" needs to be set to the number of bytes of provided data at "CBdataBufferPtr. This must be an even number and its value must be no larger than INT_MAX.

If CBtype is PROVIDING_DATA, then Merge DICOM Toolkit 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 Merge DICOM Toolkit 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 Merge DICOM Toolkit has set "CBdataSizePtr" to the total size (in bytes) of the attribute’s value.

If CBtype is PROVIDING_OFFSET_TABLE then Merge DICOM Toolkit has set "CBdataSizePtr" to the number of bytes of data it is providing at "CbdataBufferPtr.

If CBtype is FREE_DATA then Merge DICOM Toolkit has set CBdataSizePtr to NULL.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CBdataBufferPtr</strong></td>
<td>If <code>CBtype</code> is <code>REQUEST_FOR_DATA</code>, <code>REQUEST_FOR_DATA_WITH_OFFSET</code> or <code>PROVIDING_DATA</code> 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 <code>CBtype</code> is <code>REQUEST_FOR_DATA_LENGTH</code> or <code>CBtype</code> is <code>PROVIDING_DATA_LENGTH</code> this parameter is not used. If <code>CBtype</code> is <code>PROVIDING_MEDIA_DATA_LENGTH</code>, <code>*CbdataBufferPtr</code> is a pointer to long that contains the offset of the attribute’s value from the beginning of the DICOM file. If <code>CBtype</code> is <code>FREE_DATA</code> then Merge DICOM Toolkit has set <code>CBdataBufferPtr</code> to NULL.</td>
</tr>
<tr>
<td><strong>CBisFirst</strong></td>
<td>If <code>CBtype</code> is <code>REQUEST_FOR_DATA</code>, then Merge DICOM Toolkit will set <code>CBisFirst</code> 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 <code>CBtype</code> is <code>PROVIDING_DATA</code>, then Merge DICOM Toolkit will set <code>CBisFirst</code> 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 <code>CBtype</code> is <code>PROVIDING_OFFSET_TABLE</code>, then Merge DICOM Toolkit will set <code>CBisFirst</code> to TRUE (not zero) the first time it is providing data for the basic offset table. If <code>CBtype</code> is <code>REQUEST_FOR_DATA_LENGTH</code>, <code>CBtype</code> is <code>PROVIDING_DATA_LENGTH</code>, or <code>CBtype</code> is <code>PROVIDING_MEDIA_DATA_LENGTH</code> this parameter is not used.</td>
</tr>
<tr>
<td><strong>CBisLastPtr</strong></td>
<td>If <code>CBtype</code> is <code>REQUEST_FOR_DATA</code> or <code>REQUEST_FOR_DATA_WITH_OFFSET</code>, then you must set <code>*CBisLastPtr</code> 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 <code>CBtype</code> is <code>PROVIDING_DATA</code>, then Merge DICOM Toolkit has set <code>*CBisLastPtr</code> 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).</td>
</tr>
</tbody>
</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.

**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 Merge DICOM Toolkit 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 Merge DICOM Toolkit with an attribute’s value as Merge DICOM Toolkit 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 Merge DICOM Toolkit 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 or MC_Open_File_Upto_Tag_Bypass_Value are 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 Merge DICOM Toolkit 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, Merge DICOM Toolkit 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:

Merge DICOM Toolkit 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 Merge DICOM Toolkit is requesting the size (in bytes) of the attribute’s value. In this case **CallbackFunction** must set `*CBdataSizePtr` to the attribute’s value size. Merge DICOM Toolkit 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 Merge DICOM Toolkit is providing **CallbackFunction** with the total size of the attribute’s value. The value size (in bytes) is placed at `*CBdataSizePtr`. Merge DICOM Toolkit 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 Merge DICOM Toolkit 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 Merge DICOM Toolkit 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). Merge DICOM Toolkit 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 Merge DICOM Toolkit is providing data to the **CallbackFunction**. In this case Merge DICOM Toolkit 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 Merge DICOM Toolkit 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, Merge DICOM Toolkit has set `*CBisLastPtr` to TRUE (non-zero). Merge DICOM Toolkit 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 Merge DICOM Toolkit is providing basic offset table to the **CallbackFunction**. In this case Merge DICOM Toolkit 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 Merge DICOM Toolkit 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. Merge DICOM Toolkit will not call the CallbackFunction with CBtype set to PROVIDING_DATA when it calls CallbackFunction with CBtype set to PROVIDING_MEDIA_DATA_LENGTH.

CBtype is set to FREE_DATA if the memory associated with the enclosing message, item, or file is being freed by Merge DICOM Toolkit, and the USE_FREE_DATA_CALLBACK configuration option has been set to Yes. When this configuration option is set to No, registered callback functions will not be called with the FREE_DATA type.

If CallbackFunction returns with a status other than MC_NORMAL_COMPLETION, Merge DICOM Toolkit will stop calling it.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>CallbackFunction, pCallbackFunction or PrivateCode were NULL.</td>
</tr>
<tr>
<td>MC_INVALID_APPLICATION_ID</td>
<td>ApplicationID does not identify a valid application object.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**See Also**


**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, MC_Get_Frame_To_Function and MC_Duplicate_Message are called.

**Synopsis**

```c
#include “mc3msg.h”

MC_STATUS MC_Register_Compression_Callbacks (int MsgFileItemID,
  *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.

CompressionCallbackFunction Name of a function which will be called repeatedly with
blocks of uncompressed data, and will return
compressed data.

DecompressionCallbackFunction 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(DecompressionCallback)
(  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(DecompressionCallback) is being called.

CBisLast This is TRUE (not zero) the last time
YourDestMsg(DecompressionCallback) 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, represented by functions **MC_Standard_Compressor** and **MC_Standard_Decompressor**, which are capable of handling the following types of pixel data:

Grayscale:
- JPEG_BASELINE (8 bit)
- JPEG_EXTENDED_2_4 (8,10,12 bit)
- JPEG_LOSSLESS_HIER_14 (any from 2 to 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.

**NOTE:** FOR **MC_Standard_(De)compressor**:

JPEG_2000/JPEG_2000_LOSSLESS_ONLY will cause an irreversible, respective 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 the enumerated MC\_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_TRANSFER_SYNTAX</td>
<td>MsgFileItemID’s transfer syntax is not one of the encapsulated transfer syntaxes.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**See Also**

MC\_Set\_Encapsulated\_Value\_From\_Function  
MC\_Set\_Next\_Encapsulated\_Value\_From\_Function  
MC\_Get\_Encapsulated\_Value\_To\_Function  
MC\_Get\_Next\_Encapsulated\_Value\_To\_Function  
MC\_Get\_Frame\_To\_Function  
MC\_Get\_Offset\_Table\_To\_Function  
MC\_Duplicate\_Message  

**MC\_Register\_Enhanced\_MemoryLog\_Function**

Registers a callback function that will be notified as each log message is written.

**Synopsis**

```c
#include "mc3msg.h"

void MC\_Register\_Enhanced\_MemoryLog\_Function (  
    void (*MemLogFunction)()  
)  

MemLogFunction The name of a function that will be called each time  
Merge DICOM Toolkit emits a log message.
```

`MemLogFunction` must be prototyped as follows:

```c
void (*MemLogFunction)(LogInfo *logInfo),
```

* `logInfo` A structure containing information about the message that is being logged.

The `LogInfo` structure is defined as follows:

```c
typedef struct LogInfo_struct  
{  
    MsgLogType typeCode;  
    LogTime timeValues;  
    char *type;  
    char *prefix;  
}  
```
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

A `LogTime` structure containing the exact time the message was being logged. (See structure definition below.)

The string representation of the message type. For example, "E", "W", "I", "T1" etc.

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.

A string that represents the ID of the process that generated the message. In some operating environments this may be an empty string.

A string representation of the time the message was logged. This string may or may not be included in the actual log message.

A string representation of the Merge DICOM Toolkit function that generated the message. This may be an empty string.

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;

**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"`
```c
void MC_Register_MemoryLog_Function(
    void (*MemLogFunction)()
)
```

`MemLogFunction` The name of a function that will be called each time Merge DICOM Toolkit emits a log message.

`MemLogFunction` must be prototyped as follows:

```c
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** *(deprecated)*

Registers a set of callback functions that will be used to capture network data, overriding the default network capture handler.

**NOTE:** Use of this call is deprecated.

**Synopsis**

```c
#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 Merge DICOM Toolkit 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,
        struct sockaddr *SourceIPaddr,
        struct sockaddr *DestIPaddr 
    );

    void (NOEXP_FUNC *CAP_freeConnectionContext) ( 
        void *AppContext,
        void *ConnContext 
    );

    int (NOEXP_FUNC *CAP_conectionOpened) ( 
        void *AppContext,
        void *ConnContext,
        struct sockaddr *SourceIPaddr,
        struct sockaddr *DestIPaddr 
    );

    int (NOEXP_FUNC *CAP_conectionClosed) ( 
```
The functions provided in the MC_NetworkCaptureInfo structure will override the default network capture handler provided by Merge DICOM Toolkit. (Please refer to the Merge DICOM Toolkit Users Manual for more information about the default handler.)

**CAP_applicationContext** - An optional pointer to application context data. Merge DICOM Toolkit passes this pointer when it calls the callback functions. It may be NULL.

**CAP_initialization** - Merge DICOM Toolkit 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 Merge DICOM Toolkit 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 - Merge DICOM Toolkit 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 - Merge DICOM Toolkit 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. Merge DICOM Toolkit will pass this pointer (as the ConnContext parameter) to other callbacks (see below). Merge DICOM Toolkit 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.
SourceIPaddr The first Internet Protocol address used on the connection.
DestIPaddr The second Internet Protocol address used on the connection.

CAP_freeConnectionContext - Merge DICOM Toolkit calls this function when it closes a network connection. Merge DICOM Toolkit 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 - Merge DICOM Toolkit 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.

CAP_conectionClosed - Merge DICOM Toolkit 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.

CAP_dataSent - Merge DICOM Toolkit calls this function
**Remarks**

Merge DICOM Toolkit 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 Merge DICOM Toolkit 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 Merge DICOM Toolkit 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 the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the functions provided in the</td>
</tr>
<tr>
<td></td>
<td>MC_NetworkCaptureInfo structure was NULL.</td>
</tr>
</tbody>
</table>

**See Also**

- `MC_Open_Secure_Association`
- `MC_Wait_For_Secure_Association`
- `MC_Reject_Association_With_Reason_Codes`

**Synopsis**

```c
#include “mergecom.h”

MC_STATUS MC_Reject_Association(
    int AssociationID,
    REJECT_REASON Reason
)
```
MC_STATUS MC_Reject_Association_With_Reason_Codes (  
    int AssociationID,  
    unsigned short ResultCode,  
    unsigned short SourceCode,  
    unsigned short ReasonCode  
)

AssociationID  The association object's identification number
Reason  The reason the DICOM association is being rejected. Use one of the enumerated \texttt{REJECT\_REASON} types defined in “mergecom.h”:

- \texttt{PERMANENT\_NO\_REASON\_GIVEN},
- \texttt{PERMANENT\_NO\_REASON\_GIVEN\_SERV\_USER} (same as \texttt{PERMANENT\_NO\_REASON\_GIVEN}),
- \texttt{PERMANENT\_NO\_REASON\_GIVEN\_SERV\_PROV\_ACSE},
- \texttt{TRANSIENT\_NO\_REASON\_GIVEN},
- \texttt{TRANSIENT\_NO\_REASON\_GIVEN\_SERV\_USER} (same as \texttt{TRANSIENT\_NO\_REASON\_GIVEN}),
- \texttt{TRANSIENT\_NO\_REASON\_GIVEN\_SERV\_PROV\_ACSE},
- \texttt{PERMANENT\_APPLICATION\_CONTEXT\_NAME\_NOT\_SUPPORTED},
- \texttt{PERMANENT\_CALLING\_AE\_TITLE\_NOT\_RECOGNIZED},
- \texttt{PERMANENT\_CALLED\_AE\_TITLE\_NOT\_RECOGNIZED},
- \texttt{PERMANENT\_ACSE\_PROTOCOL\_VERSION\_NOT\_SUPPORTED},
- \texttt{TRANSIENT\_TEMPORARY\_CONGESTION},
- \texttt{TRANSIENT\_LOCAL\_LIMIT\_EXCEEDED}

ReasonCode  The reason field (byte 10 in A-ASSOCIATE-RJ PDU as per DICOM PS3.8, Section 9.3.4)
ResultCode  The result field (byte 8 in A-ASSOCIATE-RJ PDU as per DICOM PS3.8, Section 9.3.4)
SourceCode  The source field (byte 9 in A-ASSOCIATE-RJ PDU as per DICOM PS3.8, Section 9.3.4)

Remarks
If a \texttt{MC\_Wait\_For\_Association} or \texttt{MC\_Wait\_For\_Secure\_Association} function completes normally, one of two functions must be called. \texttt{MC\_Accept\_Association} informs the remote application that the association can proceed. Use \texttt{MC\_Reject\_Association} to reject the association request.

Upon successful return from the \texttt{MC\_Reject\_Association} call, no further calls may be made for the association.

Return Value
One of the enumerated \texttt{MC\_STATUS} codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{MC_NORMAL_COMPLETION}</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>\texttt{MC_NO_REQUEST_PENDING}</td>
<td>There is no pending association request for this \texttt{AssociationID}.</td>
</tr>
</tbody>
</table>
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 the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>The ApplicationID parameter was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_APPLICATION_ID</td>
<td>*ApplicationID is not a valid application ID.</td>
</tr>
</tbody>
</table>

See Also

MC_Register_Application

MC_Release_Callback_Function MC_Release_pCallback_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  
)

MC_STATUS MC_Release_pCallback_Function (  
    int ApplicationID,  
    char *PrivateCode,  
)
unsigned short Group,
unsigned char ElementByte,
)

ApplicationID The application object's identification number
Tag The identifying tag of an attribute which had a callback function registered for it.
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.

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.

MC_pRelease_Callback_Function releases the callback function which was registered for the private attribute. The callback function will no longer be called when the private attribute’s value is being received or when the private 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 the enumerated MC_STATUS codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_APPLICATION_ID</td>
<td>ApplicationID does not identify a valid Merge DICOM Toolkit application.</td>
</tr>
<tr>
<td>MC_INVALID_TAG</td>
<td>There was no callback function registered for Tag.</td>
</tr>
</tbody>
</table>

See Also
MC_Register_Callback_Function MC_Register_pCallback_Function

MC_Release_Library_Exception_Handler
Disables the internal system exception handler and releases the user-defined exception handler.

Synopsis
#include "mc3msg.h"

MC_STATUS MC_Release_Library_Exception_Handler ( 
    void 
)
Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
</tbody>
</table>

See Also

MC_Set_Library_Exception_Handler

MC_Release_Parent_Association (UNIX Only)

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.

Note that this routine is only needed in the parent process when fork() is called to create a child process to handle the association. It is not required when a thread is created to handle the connection.

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_ASSOC_ID</td>
<td>AssociationID does not identify a valid Merge DICOM Toolkit association.</td>
</tr>
</tbody>
</table>
See Also
MC_Wait_For_Association  MC_Wait_For_Secure_Association

MC_Release_Parent_Connection (UNIX Only)

Releases parent's copy of a socket for a connection.

Synopsis
#include "mergecom.h"

void MC_Release_Parent_Connection ( 
    MC_SOCKET Socket
)

Socket A socket returned by a MC_Wait_For_Connection or 
MC_Wait_For_Connection_On_Port function.

Remarks
MC_Release_Parent_Connection is used by a parent process to free its copy of a socket after 
spawning a child process (and passing the socket to the child). This is often done in UNIX 
environments in which a server process loops, calling the MC_Wait_For_Connection or 
MC_Wait_For_Connection_On_Port function. As each connection arrives, the process spawns a child 
process to handle the connection, then uses this function to free its copy of the connection. IT IS 
VERY IMPORTANT that MC_Release_Parent_Connection be called in this situation to avoid memory 
leaks.

Note that this routine is only needed in the parent process when fork() is called to create a child 
process to handle the connection. It is not required when a thread is created to handle the 
connection.

Return Value
There is no return value.

See Also
MC_Wait_For_Connection  MC_Wait_For_Connection_On_Port 
MC_Process_Association_Request 
MC_Process_Secure_Association_Request

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 ( 
    MC_ul_size_t *bytesAllocated, 
    MC_ul_size_t *bytesInUse
)
**bytesAllocated**  The number of bytes that the toolkit has allocated. This parameter is type size_t on 64-bit Windows and unsigned long on other platforms.

**bytesInUse**  The number of bytes of non-freeable memory. This parameter is type size_t on 64-bit Windows and unsigned long on other platforms.

**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 the enumerated **MC_STATUS** codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_LIBRARY_NOT_INITIALIZED</td>
<td>The library has not been properly initialized.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or both of the parameters were non-initialized pointers.</td>
</tr>
</tbody>
</table>

**See Also**
MC_Cleanup_Memory

**MC_Reset_Filename**
Resets the filename associated with a file object

**Synopsis**

```c
#include “mc3media.h”

MC_STATUS MC_Reset_Filename (  
    int FileID,  
    char *NewFilename
)
```

**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 the enumerated `MC_STATUS` codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_FILE_ID</td>
<td>The <code>FileID</code> value is not a valid file object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td><code>NewFilename</code> was NULL.</td>
</tr>
</tbody>
</table>

**See Also**

MC_Write_File    
MC_Empty_File

### MC_Reset_Message_Transfer_Syntax

Resets the transfer syntax over which a message will be sent on the network.

**Synopsis**

```c
#include "mergecom.h"

MC_STATUS MC_Reset_Message_Transfer_Syntax (int MessageID)
```

*MessageID*  
The identifier assigned to this object by the  
MC_Read_Message, MC_Open_Message or  
MC_Open_Empty_Message function.

**Remarks**

`MC_Reset_Message_Transfer_Syntax` resets the DICOM transfer syntax associated with a message. When a message is read off the network, `MC_Read_Message` stores the transfer syntax in which the message was received over the network, similarly to how `MC_Set_Message_Transfer_Syntax` can be used to set the transfer syntax for a message. If the same message is sent again over the network, it must be transferred with the transfer syntax that it was received in.

If the transfer syntax of the message is reset with `MC_Reset_Message_Transfer_Syntax`, Merge DICOM Toolkit will pick any uncompressed transfer syntax to send over the network.

**Return Value**

One of the enumerated `MC_STATUS` codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td><code>MessageID</code> is not a valid message object ID.</td>
</tr>
</tbody>
</table>

**See Also**

MC_Send_Request_Message    
MC_Set_Message_Transfer_Syntax
**MC_Send_Request_Message**

Sends a request message to the remote application.

**Synopsis**

```c
#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. (Merge DICOM Toolkit 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 Merge DICOM Toolkit.
- 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 Merge DICOM Toolkit.
• 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 Merge DICOM Toolkit.

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 the enumerated `MC_STATUS` codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_ASSOC_ID</td>
<td><code>AssociationID</code> is not a valid association object ID.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>The <code>MessageID</code> value is not a valid message object ID.</td>
</tr>
<tr>
<td>MC_MAX_OPERATIONS_EXCEEDED</td>
<td>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.</td>
</tr>
<tr>
<td>MC_REQUIRED_ATTRIBUTE_MISSING</td>
<td>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 <code>MessageID</code> has not been set.</td>
</tr>
<tr>
<td>MC_UNACCEPTABLE_SERVICE</td>
<td>The service related to <code>MessageID</code> has not been successfully negotiated for this association.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_ASSOCIATION_ABORTED</td>
<td>The association has been aborted. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**See Also**

- `MC_Register_Callback_Function`
- `MC_Open_Message`
- `MC_Read_Message`
- `MC_Read_To_Stream`
- `MC_Send_Response_Message`
- `MC_Send_Response`
- `MC_Send_Request_Message_For_Service`
- `MC_Send_Request_For_Service`
MC_Send_Request_Message_For_Service

Sends a request message to the remote application.

Synopsis

```c
#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_For_Service** is called.

If **MC_Register_Callback_Function** was called to register a callback function for any attributes in the message, **MC_Send_Request_Message_For_Service** 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. (Merge DICOM Toolkit 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_For_Service** 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 Merge DICOM Toolkit.

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 Merge DICOM Toolkit.

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 Merge DICOM Toolkit.

If MC_Send_Request_Message_For_Service returns a status code of MC_ASSOCIATION_ABORTED, no further calls may be made for the association.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_ASSOC_ID</td>
<td>AssociationID is not a valid association object ID.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>The MessageID value is not a valid message object ID.</td>
</tr>
<tr>
<td>MC_MAX_OPERATIONS_EXCEEDED</td>
<td>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.</td>
</tr>
<tr>
<td>MC_REQUIRED_ATTRIBUTE_MISSING</td>
<td>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.</td>
</tr>
<tr>
<td>MC_UNACCEPTABLE_SERVICE</td>
<td>The service related to MessageID has not been successfully negotiated for this association.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_ASSOCIATION_ABORTED</td>
<td>The association has been aborted. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>
**See Also**

<table>
<thead>
<tr>
<th>Function</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_Register_Callback Function</td>
<td>MC_Open_Message</td>
</tr>
<tr>
<td>MC_Send_Request_Message</td>
<td>MC_Send_Request</td>
</tr>
<tr>
<td>MC_Send_Response_Message</td>
<td>MC_Send_Response</td>
</tr>
<tr>
<td>MC_Read_Message</td>
<td>MC_Read_To_Stream</td>
</tr>
<tr>
<td>MC_Get_Meta_ServiceName</td>
<td></td>
</tr>
</tbody>
</table>

**MC_Send_Request**

Sends a request message to the remote application directly from a callback function. Especially useful for large data transfers which we don’t want to load into application memory.

**Synopsis**

```c
#include "mergecom.h"

MC_STATUS MC_Send_Request (  
    int AssociationID,  
    char *Filename,  
    void *UserInfo,  
    MC_STATUS (*CallbackFunction) ()  
)
```

- **AssociationID**
  - The association object's identification number

- **Filename**
  - This parameter is unused (legacy). It may be anything, including NULL.

- **UserInfo**
  - Address of user data which will be passed on to `CallbackFunction` each time it is called. This may be NULL.

- **CallbackFunction**
  - Name of a function which will be called repeatedly to provide blocks of request data.

The function must be prototyped as follows:

```c
MC_STATUS CallbackFunction (  
    int CBAssociationID,  
    unsigned long CBtag,  
    void* CBUserInfo,  
    CALLBACK_TYPE CBtype,  
    unsigned long* CBdataSizePtr,  
    void** CBdataBufferPtr,  
    int CBisFirst,  
    int* CBisLastPtr  
)
```

where:

- **CBAssociationID**  
  - The association object's identification number

- **CBtag**  
  - Not used and has undefined value (0xFFFFFFFF)

and all other `CallbackFunction` parameters as described in the `MC_Register_Callback_Function` section.
Remarks

**MC_Send_Request** sends a request to the remote application using user’s *CallbackFunction* as the message data provider. The request message might be associated with DICOM file identified by *Filename* parameter.

The using of *CallbackFunction* mechanism facilitates the sending of large messages, decreases the application memory footprint and in many cases increases the overall performance. The data blocks of the request message will be repeatedly requested through *CallbackFunction* and sent to the remote application across an open DICOM association identified by *AssociationID*.

**NOTE:** Some DICOM services require that values for certain “command level” message attributes (i.e. group 0 attributes) be present. Merge DICOM Toolkit automatically adds command level attributes to the sent message when **MC_Send_Request** is called (see details in **MC_Send_Request_Message** section).

If **MC_Send_Request** returns a status code of **MC_ASSOCIATION_ABORTED**, no further calls may be made for the association.

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_INVALID_ASSOC_ID</strong></td>
<td><em>AssociationID</em> is not a valid association object ID.</td>
</tr>
<tr>
<td><strong>MC_INVALID_MESSAGE_ID</strong></td>
<td>The <em>MessageID</em> value is not a valid message object ID.</td>
</tr>
<tr>
<td><strong>MC_MAX_OPERATIONS_EXCEEDED</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>MC_REQUIRED_ATTRIBUTE_MISSING</strong></td>
<td>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 <em>MessageID</em> has not been set.</td>
</tr>
<tr>
<td><strong>MC_UNACCEPTABLE_SERVICE</strong></td>
<td>The service related to <em>MessageID</em> has not been successfully negotiated for this association.</td>
</tr>
<tr>
<td><strong>MC_SYSTEM_ERROR</strong></td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

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 Merge DICOM Toolkit log file.
MC_Send_Request_For_Service

Sends a request message to the remote application directly from a callback function. Especially useful for large data transfers which we don't want to load into application memory.

Synopsis

```
#include "mergecom.h"

MC_STATUS MC_Send_Request_For_Service (  
    int AssociationID,  
    char *AServiceName  
    char *Filename,  
    void *UserInfo,  
    MC_STATUS (*CallbackFunction) ()  
)  
```

AssociationID
The association object's identification number

AServiceName
Name of the service negotiated to utilize for the presentation context.

Filename
This parameter is unused (legacy). It may be anything, including NULL.

UserInfo
Address of user data which will be passed on to CallbackFunction each time it is called. This may be NULL.

CallbackFunction
Name of a function which will be called repeatedly to provide blocks of request data.

The function must be prototyped as follows:

```
MC_STATUS CallbackFunction (  
    int CBAssociationID,  
    unsigned long CBtag,  
    void* CBuserInfo,  
    CALLBACK_TYPE CBtype,  
    unsigned long* CBdataSizePtr,  
    void** CBdataBufferPtr,  
    int CBisFirst,  
    int* CBisLastPtr  
)  
```

where:

CBAssociationID
The association object's identification number
**CBtag**

Not used and has undefined value (0xFFFFFFFF) and all other *CallbackFunction* parameters as described in *MC_Register_Callback_Function* section.

**Remarks**

*MC_Send_Request_For_Service* sends a request message to the remote application using user’s *CallbackFunction* as the message data provider. The request message might be associated with DICOM file identified by *Filename* parameter.

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 data blocks of the message will be repeatedly requested through *CallbackFunction* and sent to the remote application across an open DICOM association identified by *AssociationID*.

**NOTE:** Some DICOM services require that values for certain “command level” message attributes (i.e. group 0 attributes) be present. Merge DICOM Toolkit automatically adds command level attributes to the sent message when *MC_Send_Request_For_Service* is called (see details in *MC_Send_Request_Message* section).

If *MC_Send_Request_For_Service* returns a status code of *MC_ASSOCIATION_ABORTED*, no further calls may be made for the association.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_INVALID_ASSOC_ID</strong></td>
<td><em>AssociationID</em> is not a valid association object ID.</td>
</tr>
<tr>
<td><strong>MC_INVALID_MESSAGE_ID</strong></td>
<td>The <em>MessageID</em> value is not a valid message object ID.</td>
</tr>
<tr>
<td><strong>MC_MAX_OPERATIONS_EXCEEDED</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>MC_REQUIRED_ATTRIBUTE_MISSING</strong></td>
<td>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 <em>MessageID</em> has not been set.</td>
</tr>
</tbody>
</table>
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 Merge DICOM Toolkit 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 Merge DICOM Toolkit log file.

See Also

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_Register_Callback_Function</td>
<td></td>
</tr>
<tr>
<td>MC_Send_Request_Message</td>
<td></td>
</tr>
<tr>
<td>MC_Send_Response_Message</td>
<td></td>
</tr>
<tr>
<td>MC_Read_Message</td>
<td></td>
</tr>
<tr>
<td>MC_Get_Meta_ServiceName</td>
<td></td>
</tr>
</tbody>
</table>

**MC_Send_Response_Message**

Sends a response to a message received from the remote application.

**Synopsis**

```c
#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** or **MC_Read_To_Stream**.

**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. (Merge DICOM Toolkit 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 Merge DICOM Toolkit.
- 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 Merge DICOM Toolkit.
- 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 Merge DICOM Toolkit.

If MC_Send_Response_Message returns MC_ASSOCIATION_ABORTED, no further calls may be made for that association.

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_ASSOC_ID</td>
<td>AssociationID is not a valid association object ID.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>The MessageID value is not a valid message object ID.</td>
</tr>
<tr>
<td>MC_REQUIRED_ATTRIBUTE_MISSING</td>
<td>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.</td>
</tr>
<tr>
<td>MC_UNACCEPTABLE_SERVICE</td>
<td>The service related to MessageID has not been successfully negotiated for this association.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>
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 Merge DICOM Toolkit log file.

### See Also

- **MC_Send_Request_Message**
- **MC_Read_Message**

## MC_Send_Response

Sends a response to a message received from the remote application. The response is sent directly from a callback function. Especially useful for large data transfers which we don’t want to load into application memory.

### Synopsis

```c
#include "mergecom.h"

MC_STATUS MC_Send_Response(
    int AssociationID,
    RESP_STATUS ResponseStatus,
    char *Filename,
    void *UserInfo,
    MC_STATUS (*CallbackFunction)()
)
```

- **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”.
- **Filename**  
  This parameter is unused (legacy). It may be anything, including NULL.
- **UserInfo**  
  Address of user data which will be passed on to `CallbackFunction` each time it is called. This may be NULL.
- **CallbackFunction**  
  Name of a function which will be called repeatedly to provide blocks of the response message data.

The function must be prototyped as follows:

```c
MC_STATUS CallbackFunction(
    int CBAssociationID,
    unsigned long CBtag,
    void *CBuserInfo,
    CALLBACK_TYPE CBtype,
    unsigned long *CBdataSizePtr,
    void **CBdataBufferPtr,
    int CBisFirst,
    int *CBisLastPtr
)
```

where:
CBAssociationID  The association object’s identification number
CBtag  Not used and has undefined value (0xFFFFFFFF)

and all other CallbackFunction parameters as described in MC_Register_Callback_Function section.

Remarks

MC_Send_Response allows the caller to respond to a message received from the remote application using the callback mechanism. The response data should be provided in the form of a DICOM object. The data blocks of the response message will be repeatedly requested through CallbackFunction and sent to the remote application across an open DICOM association identified by AssociationID.

MC_Send_Response is called after receiving a request message using MC_Read_Message or MC_Read_To_Stream.

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.

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: See MC_Send_Response_Message section for the details regarding the handling of the required message attributes for DICOM services.

If MC_Send_Response returns MC_ASSOCIATION_ABORTED, no further calls may be made for that association.

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_ASSOC_ID</td>
<td>AssociationID is not a valid association object ID.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>The MessageID value is not a valid message object ID.</td>
</tr>
<tr>
<td>MC_REQUIRED_ATTRIBUTE_MISSING</td>
<td>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.</td>
</tr>
<tr>
<td>MC_UNACCEPTABLE_SERVICE</td>
<td>The service related to MessageID has not been successfully negotiated for this association.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>
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 Merge DICOM Toolkit log file.

### See Also

- **MC_Send_Request_Message**
- **MC_Send_Request**
- **MC_Read_Message**
- **MC_Read_To_Stream**

## MC_Set_Bool_Config_Value

Used to set the value of a boolean toolkit configuration parameter at runtime.

### Synopsis

```c
#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:

- ACCEPT_ANY_APPLICATION_TITLE
- ACCEPT_ANY_CONTEXT_NAME
- ACCEPT_ANY_HOSTNAME
- ACCEPT_ANY_PRESENTATION_CONTEXT
- ACCEPT_DIFFERENT_IC_UID
- ACCEPT_DIFFERENT_VERSION
- ACCEPT_MULTIPLE_PRES_CONTEXTS
- ACCEPTRELATEDGENERALSERVICES
- ACCEPT_STORAGE_SERVICE_CONTEXTS
- ALLOW_COMMA_IN_DS_FL_FD_STRINGS
- ALLOW_EMPTY_PDV_LENGTH
- ALLOW_INVALID_PRIVATE_ATTRIBUTES
- ALLOW_INVALID_PRIVATE_CREATOR_CODES
- ALLOW_LIBRARY_EXCEPTION_HANDLER
- ALLOW_OUT_OF_RANGE_BITS_JPEG_LOSSLESS
- ATT_00081190_USE_UT_VR
- ATT_00287FE0_USE_UT_VR
- ATT_0040E010_USE_UT_VR
- ATT_0074100A_USE_ST_VR
- AUTO_ECHO_SUPPORT
- BLANK_FILL_LOG_FILE
- CALCULATE_DEFINED_LENGTH_FOR_CB
- COMPRESSION_ALLOW_FRAGS
- COMPRESSION_J2K_LOSSY_USE_QUALITY
- COMPRESSION_USE_HEADER_QUERY
- COMPRESSION_WHEN_J2K_USE_LOSSY
- CREATE_OFFSET_TABLE
- DEFLATE_ALLOW_FLUSH
- DICOMDIR_STREAM_STORAGE
DUPLICATE_ENCAPSULATED_ICON
ELIMINATE_ITEM_REFERENCES
EMPTY_PRIVATE_CREATOR_CODES
EXPLICIT_VR_TO_UN_FOR_LENGTH_GT_64K
EXPORT_EMPTY_PRIVATE_CREATOR_CODES
EXPORT_GROUP_LENGTHS_TO_MEDIA
EXPORT_GROUP_LENGTHS_TO_NETWORK
EXPORT_PRIVATE_ATTRIBUTES_TO_MEDIA
EXPORT_PRIVATE_ATTRIBUTES_TO_NETWORK
EXPORT_UN_VR_TO_MEDIA
EXPORT_UN_VR_TO_NETWORK
EXPORT_UNDEFAINED_LENGTH_SQ
EXPORT_UNDEFAINED_LENGTH_SQ_IN_DICOMDIR
FORCE_OPEN_EMPTY_ITEM
HARD_CLOSE_TCP_IP_CONNECTION
INSURE_EVEN_UID_LENGTH
LIST_UN_ATTRIBUTES
LOG_FILE_BACKUP
MSG_FILE_ITEM_OBJ_TRACE
NETWORK_CAPTURE
PRIVATE_SYNTAX_1_ENCAPSULATED
PRIVATE_SYNTAX_1_EXPLICIT_VR
PRIVATE_SYNTAX_1_LITTLE_ENDIAN
PRIVATE_SYNTAX_2_ENCAPSULATED
PRIVATE_SYNTAX_2_EXPLICIT_VR
PRIVATE_SYNTAX_2_LITTLE_ENDIAN
REJECT_INVALID_VR
RELEASE_SQ_ITEMS
REMOVE_PADDING_CHARS
REMOVE_SINGLE_TRAILING_SPACE
RETURN_COMMA_IN_DS_FL_FD_STRINGS
REWRITE_CAPTURE_FILES
SEND_ECHO_PRIORITY
SEND_LENGTH_TO_END
SEND_MSG_ID_RESPONSE
SEND_RECOGNITION_CODE
SEND_RESPONSE_PRIORITY
SEND_SOP_CLASS_UID
SEND_SOP_INSTANCE_UID
TCPIP_DISABLE_NAGLE
TOLERATE_INVALID_IN_DEFAULT_CHARSET
UPDATE_GROUP_0028_ON_DUPLICATE
USE_FREE_DATA_CALLBACK

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

\*value\*  The boolean value to which Aparm is to be set.
Remarks

The Merge DICOM 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 toolkit configuration elsewhere in this manual.

Return Value

One of the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MC_NORMAL_COMPLETION</code></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><code>MC_INVALID_PARAMETER_NAME</code></td>
<td>Parameter specified is not one of the above listed parameters.</td>
</tr>
<tr>
<td><code>MC_LIBRARY_NOT_INITIALIZED</code></td>
<td>The library has not been properly initialized.</td>
</tr>
</tbody>
</table>

See Also

- `MC_Get_Int_Config_Value`
- `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_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

```c
#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:

```c
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 Merge DICOM Toolkit 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 Merge DICOM Toolkit, 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.

Merge DICOM Toolkit 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 defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_TRANSFER_SYNTAX</td>
<td>The message’s transfer syntax is non-encapsulated.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
</tbody>
</table>
| 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_IN compatible 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**

```c
#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 the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_FILE_ID</td>
<td>The <code>FileID</code> value is not a valid file object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td><code>Preamble</code> was NULL.</td>
</tr>
</tbody>
</table>

**See Also**

`MC_Get_File_Preamble`

**MC_Set_Int_Config_Value**

Used to set the value of a integer toolkit configuration parameter at runtime.

**Synopsis**

```c
#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:
  - ARTIM_TIMEOUT
  - ASSOC_REPLY_TIMEOUT

© Copyright Merge Healthcare Solutions Inc. 2023
COMPRESSION_CHROM_FACTOR
COMPRESSION_J2K_LOSSY_QUALITY
COMPRESSION_J2K_LOSSY_RATIO
COMPRESSION_LUM_FACTOR
CONNECT_TIMEOUT
DEFLATE_COMPRESSION_LEVEL
DESIRED_LAST_PDU_SIZE
FLATE_GROW_OUTPUT_BUF_SIZE
IGNORE_JPEG_BAD_SUFFIX
INACTIVITY_TIMEOUT
LARGE_DATA_SIZE
LIST_SQ_DEPTH_LIMIT
LIST_VALUE_LIMIT
LOG_FILE_LINE_LENGTH
LOG_FILE_SIZE
LOG_MEMORY_SIZE
MAX_PENDING_CONNECTIONS
NUM_HISTORICAL_LOG_FILES
NUMBER_OF_CAP_FILES
OBOW_BUFFER_SIZE
PEGASUS_NUMBER_OF_THREADS
RELEASE_TIMEOUT
TCPIP_KEEP_ALIVE_INTERVAL
TCPIP_KEEP_ALIVE_TIME
TCPIP_LISTEN_PORT
TCPIP_RECEIVE_BUFFER_SIZE
TCPIP_SEND_BUFFER_SIZE
WORK_BUFFER_SIZE
WRITE_TIMEOUT

These names are the same as those given to the parameters in the toolkit 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 Merge DICOM Toolkit 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 toolkit configuration elsewhere in this manual.

Return Value
One of the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_PARAMETER_NAME</td>
<td>Parameter specified is not one of the above listed parameters.</td>
</tr>
<tr>
<td>MC_MUST_BE_POSITIVE</td>
<td>Parameter value cannot be negative.</td>
</tr>
</tbody>
</table>
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_Library_Exception_Handler

MC_SET.Library_Exception_Handler

Initializes the internal system exception handler and registers the user-defined exception handler which will be called if System exception or System signal are raised.

Synopsis

```c
#include "mc3msg.h"

MC_STATUS MC_Set_Library_Exception_Handler ( 
    MCLibraryExceptionHandler handler 
) 
```

`handler` A user-defined exception handler which is called if System exception or System signal is raised

The handler must be prototyped as follows:

```c
void UserExceptionHandler ( 
    int ExceptionId 
) 
```

Remarks

This call allows you to initialize the internal exception handler and to register a user-defined exception handler callback, which will be called in case System exception or System signal are raised. This allows for the (limited) recovery of the user program by at least giving it a chance to cleanup and exit gracefully and avoid a straight crash.

Note

To enable the system exception handlers, ALLOW_LIBRARY_EXCEPTION_HANDLER configuration parameter should be set to 'Yes'.

Return Value

One of the enumerated `MC_STATUS` codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
</tbody>
</table>

See Also

MC_Release_Library_Exception_Handler
**MC_Set_Log_Destination**

Determines where log messages will be written.

**Synopsis**

```c
#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. `A 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

**Avalue**

A 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 Merge DICOM Toolkit 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 toolkit configuration elsewhere in this manual.

**Return Value**

One of the enumerated `MC_STATUS` codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
</tbody>
</table>
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_Int_Config_Value
MC_Get_Long_Config_Value MC_Set_Long_Config_Value
MC_Get_String_Config_Value MC_Set_String_Config_Value

MC_Set_Log_Prefix

Sets the 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 toolkit.

Return Value
None

See Also
MC_Get_Log_Destination MC_Set_Log_Destination

MC_Set_Long_Config_Value

Sets the value of a long integer toolkit 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:
    CAPTURE_FILE_SIZE,
    PDU_MAXIMUM_LENGTH,
    CALLBACK_MIN_DATA_SIZE,
    PIXEL_BUFFER_SIZE

© Copyright Merge Healthcare Solutions Inc. 2023
This name is the same as those given to the parameters in the toolkit configuration files.

Avalue The long integer value to which Aparm is to be set.

Remarks

The Merge DICOM Toolkit 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 it is set. Please see the detailed description of toolkit configuration elsewhere in this manual.

Return Value

One of the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_PARAMETER_NAME</td>
<td>Parameter specified is not one of the above listed parameters.</td>
</tr>
<tr>
<td>MC_MUST_BE_POSITIVE</td>
<td>Parameter value cannot be negative.</td>
</tr>
<tr>
<td>MC_LIBRARY_NOT_INITIALIZED</td>
<td>The library has not been properly initialized.</td>
</tr>
</tbody>
</table>

See Also

- `MC_Get_Int_Config_Value`
- `MC_Set_Int_Config_Value`
- `MC_Get_Long_Config_Value`
- `MC_Set_Long_Config_Value`
- `MC_Get_String_Config_Value`
- `MC_Set_String_Config_Value`
- `MC_Get_Log_Destination`
- `MC_Set_MergeINI`
- `MC_Set_MergeINI_Unicode`

Set the path of the merge.ini file so that the MERGE_INI environment variable is not used.

Synopsis

```c
#include "mc3msg.h"

MC_STATUS MC_Set_MergeINI ( char *Filename )

MC_STATUS MC_Set_MergeINI_Unicode ( wchar_t *Filename, )
```

Filename An absolute or relative path containing the location of the merge.ini file. For `MC_Set_MergeINI_Unicode`, it must be specified as a null terminated Unicode string: UTF-16 (wchar_t) for Windows platforms and UTF-8 for non Windows.
Remarks

The Merge DICOM Toolkit 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` and `MC_Set_Merge_INIUnicode` 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 the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MC_NORMAL_COMPLETION</code></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><code>MC_SYSTEM_ERROR</code></td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td><code>MC_LIBRARY_ALREADY_INITIALIZED</code></td>
<td>The library has already been initialized.</td>
</tr>
</tbody>
</table>

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

```c
#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 the enumerated **MC_STATUS** codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_APPLICATION_ID</td>
<td>ApplicationID does not identify a valid application object.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>MessageFileID is not a valid message or file object ID.</td>
</tr>
<tr>
<td>MC_LIBRARY_NOT_INITIALIZED</td>
<td>The library has not been properly initialized.</td>
</tr>
</tbody>
</table>

### 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 through the network.

#### Synopsis

```c
#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. Use one of the enumerated TRANSFER_SYNTAX types defined in "mc3msg.h" (see **MC_Get_Transfer_Syntax_From_Enum**).

#### 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 *Merge DICOM Toolkit User's Manual* for a discussion on negotiating multiple transfer syntaxes for a service.

**MC_Set_Message_Transfer_Syntax** can also be used to set the transfer syntax of a DICOM file object. When used with a DICOM file object, the tag (0002,0010) Transfer Syntax UID will be set with the proper UID for the transfer syntax specified.
Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>MessageID is not a valid message object ID.</td>
</tr>
</tbody>
</table>

See Also

MC_Send_Request_Message      MC_Get_Message_Transfer_Syntax
MC_Reset_Message_Transfer_Syntax

MC_Set_Negotiation_Info

Registers extended negotiation information.

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.

NOTE: Use of this call is deprecated.

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 Merge DICOM Toolkit with extended negotiation information which it will use when establishing associations.

When a MC_Open_Association or MC_Open_Secure_Association call is made, Merge DICOM Toolkit 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, Merge DICOM Toolkit 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. Merge DICOM Toolkit 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 the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_APPLICATION_ID</td>
<td><code>ApplicationID</code> is not a valid application identifier.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>Either <code>ServiceName</code> or <code>ExtInfoBuffer</code> was NULL.</td>
</tr>
<tr>
<td>MC_UNKNOWN_SERVICE</td>
<td><code>ServiceName</code> was not registered in the Merge DICOM Toolkit configuration files.</td>
</tr>
<tr>
<td>MC_LIBRARY_NOT_INITIALIZED</td>
<td>The library has not been properly initialized.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**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**

```c
#include "mergecom.h"

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

`AssociationID` An association identification number returned by an `MC_Wait_For_Association` or `MC_Wait_For_Secure_Association` call.
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 Merge DICOM Toolkit 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, Merge DICOM Toolkit 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 the enumerated **MC_STATUS** codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_INVALID_ASSOC_ID</strong></td>
<td>AssociationID is not a valid association identifier.</td>
</tr>
<tr>
<td><strong>MC_NO_REQUEST_PENDING</strong></td>
<td>AssociationID references to an association that has already been accepted.</td>
</tr>
<tr>
<td><strong>MC_NULL_POINTER_PARM</strong></td>
<td>Either <em>ServiceName</em> or <em>ExtInfoBuffer</em> was NULL.</td>
</tr>
<tr>
<td><strong>MC_UNKNOWN_SERVICE</strong></td>
<td><em>ServiceName</em> was not registered in the Merge DICOM Toolkit configuration files or was not negotiated for the association.</td>
</tr>
<tr>
<td><strong>MC_LIBRARY_NOT_INITIALIZED</strong></td>
<td>The library has not been properly initialized.</td>
</tr>
<tr>
<td><strong>MC_SYSTEM_ERROR</strong></td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**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**

```c
#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:

```c
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 Merge DICOM Toolkit 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.
**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 Merge DICOM Toolkit, 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.

Merge DICOM Toolkit 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 defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_INVALID_TRANSFER_SYNTAX</strong></td>
<td>The message's transfer syntax is non-encapsulated.</td>
</tr>
<tr>
<td><strong>MC_NULL_POINTER_PARM</strong></td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td><strong>MC_INVALID_TAG</strong></td>
<td>The message does not contain an attribute with an ID of Tag.</td>
</tr>
<tr>
<td></td>
<td>The attribute can be added with <strong>MC_Add_Standard_Attribute</strong>.</td>
</tr>
<tr>
<td></td>
<td>This status will not be returned if setting the value of a message object which was opened</td>
</tr>
</tbody>
</table>
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_Float (  
    int MsgFileItemID,  
    char *PrivateCode,  
    unsigned short Group,  
    unsigned char ElementByte,  
    float 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_ShortInt (  
    int MsgFileItemID,  
    char *PrivateCode,  
    unsigned short Group,  
    unsigned char ElementByte,  
    short Value
)
```

© Copyright Merge Healthcare Solutions Inc. 2023
unsigned char ElementByte,
short 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_Int ( int MsgFileItemID, char *PrivateCode,
unsigned short Group,
unsigned char ElementByte,
int 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_LongInt ( int MsgFileItemID, char *PrivateCode,
unsigned short Group,
unsigned char ElementByte,
long int 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_LongLong ( int MsgFileItemID, char *PrivateCode,
unsigned short Group,
unsigned char ElementByte,
long long Value
)

MC_STATUS MC_Set_Next_pValue_From_ULongLong ( int MsgFileItemID, char *PrivateCode,
unsigned short Group,
unsigned char ElementByte,
unsigned long long 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_UnicodeString(
  int MsgFileItemID,
  char *PrivateCode,
  unsigned short Group,
  unsigned char ElementByte,
  int ValueLength,
  const MC_Uhar *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
- **LongLong_Type** Binary 64-bit integer number
- **ULongLong_Type** Binary 64-bit unsigned integer number
- **Float_Type** Binary Floating point number
- **Double_Type** Binary Floating point number
- **Buffer_Type** Binary byte value

**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.

For `MC_Set_Next_pValue_From_UnicodeString`, the `ValueLength` parameter is for specifying the number of Unicode characters in the input `Value`. This function requires `MC_Enable_Unicode_Conversion` being called first. See documentation for `MC_Enable_Unicode_Conversion`.

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`.

<table>
<thead>
<tr>
<th>Function</th>
<th>May be used to set values to attributes with these Value Representations</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MC_Set_Next_pValue_From_Float</code></td>
<td>DS, FD, FL, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td><code>MC_Set_Next_pValue_From_Double</code></td>
<td>DS, FD, FL, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td><code>MC_Set_Next_pValue_From_ShortInt</code></td>
<td>DS, FD, FL, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td><code>MC_Set_Next_pValue_From_UShortInt</code></td>
<td>DS, FD, FL, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td><code>MC_Set_Next_pValue_From_Int</code></td>
<td>DS, FD, FL, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td><code>MC_Set_Next_pValue_From_UInt</code></td>
<td>DS, FD, FL, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td><code>MC_Set_Next_pValue_From_LongInt</code></td>
<td>AT, DS, FD, FL, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td><code>MC_Set_Next_pValue_From_ULongInt</code></td>
<td>AT, DS, FD, FL, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td><code>MC_Set_Next_pValue_From_LongLong</code></td>
<td>AT, DS, FD, FL, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td><code>MC_Set_Next_pValue_From_ULongLong</code></td>
<td>AT, DS, FD, FL, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td><code>MC_Set_Next_pValue_From_UnicodeString</code></td>
<td>LO, LT, PN, SH, ST, UC, UT</td>
</tr>
</tbody>
</table>

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 defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
</tbody>
</table>

**INFORMATIONAL MESSAGES:**

**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.

**WARNINGS:**

( **VALUE WAS STILL ENCODED**)

**MC_INCOMPATIBLE_VR** The attribute’s value representation cannot be derived from *Value*. See the table above.

**MC_INVALID_PRIVATE_CODE** The message does not contain any 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_NULL_POINTER_PARM** One or more of the pointer-type parameters was NULL.

**MC_TOOMANY_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 Merge DICOM Toolkit log file.

**ERRORS:**

**MC_INVALID_DATA_TYPE** *DataType* is not valid.

**MC_INVALID_MESSAGE_ID** *MsgFileItemID* is not a valid message object ID, file object ID or item object ID.

**MC_INVALID_VALUE_FOR_VR** *Value* does not conform to the requirements of its value representation.

**MC_INVALID_PRIVATE_CODE** The message does not contain any 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_NULL_POINTER_PARM** One or more of the pointer-type parameters was NULL.

**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 Merge DICOM Toolkit 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

```c
#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 the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INCOMPATIBLE_VR</td>
<td>The attribute identified by <code>Tag</code> can not be set to NULL.</td>
</tr>
<tr>
<td>MC_INVALID_PRIVATE_CODE</td>
<td>The message does not contain any attributes in <code>Group</code> for <code>PrivateCode</code>.</td>
</tr>
</tbody>
</table>
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 Merge DICOM Toolkit 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_Float (  
  int(MsgFileItemID),
  unsigned long(Tag),
  float(Value)
)

MC_STATUS MC_Set_Next_Value_From_Double (  
  int(MsgFileItemID),
  unsigned long(Tag),
  double(Value)
)

MC_STATUS MC_Set_Next_Value_From_ShortInt (  
  int(MsgFileItemID),
  unsigned long(Tag),
  short 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_Int (  
  int(MsgFileItemID),
  unsigned long(Tag),
  int(Value)
)
```

© Copyright Merge Healthcare Solutions Inc. 2023
unsigned long Tag,
int Value
}

MC_STATUS MC_Set_Next_Value_From_UInt (  
    int MsgFileItemID,
    unsigned long Tag,
    unsigned 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_ULongInt (  
    int MsgFileItemID,
    unsigned long Tag,
    unsigned long Value
)

MC_STATUS MC_Set_Next_Value_From_LongLong (  
    int MsgFileItemID,
    unsigned long Tag,
    long long Value
)

MC_STATUS MC_Set_Next_Value_From_ULongLong (  
    int MsgFileItemID,
    unsigned long Tag,
    unsigned long long Value
)

MC_STATUS MC_Set_Next_Value_From_String (  
    int MsgFileItemID,
    unsigned long Tag,
    char *Value
)

MC_STATUS MC_Set_Next_Value_From_UnicodeString (  
    int MsgFileItemID,
    unsigned long Tag,
    int ValueLength,
    const MC_Uhar *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:

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
Int_Type
Ulnt_Type
ShortInt_Type
UShortInt_Type
LongInt_Type
ULongInt_Type
LongLong_Type
ULongLong_Type
Float_Type
Double_Type
Buffer_Type

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.

For MC_Set_Next_Value_From_UnicodeString, the ValueLength parameter is for specifying the number of Unicode characters in the input Value. This function requires MC_Enable_Unicode_Conversion being called first. See documentation for MC_Enable_Unicode_Conversion.

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.

<table>
<thead>
<tr>
<th>Functions</th>
<th>May be used to set values to attributes with these Value Representations</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_Set_Next_Value_From_Float</td>
<td>DS, FD, FL, IS, SL, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Set_Next_Value_From_Double</td>
<td>DS, FD, FL, IS, SL, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Set_Next_Value_From_ShortInt</td>
<td>DS, FD, FL, IS, SL, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Set_Next_Value_From_USHORTInt</td>
<td>DS, FD, FL, IS, SL, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Set_Next_Value_From_Int</td>
<td>DS, FD, FL, IS, SL, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>Functions</td>
<td>May be used to set values to attributes with these Value Representations</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MC_Set_Next_Value_From_UInt</td>
<td>DS, FD, FL, IS, SL, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Set_Next_Value_From_LongInt</td>
<td>AT, DS, FD, FL, IS, SL, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Set_Next_Value_From_ULongInt</td>
<td>AT, DS, FD, FL, IS, SL, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Set_Next_Value_From_LongLong</td>
<td>AT, DS, FD, FL, IS, SL, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Set_Next_Value_From_ULongLong</td>
<td>AT, DS, FD, FL, IS, SL, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Set_Next_Value_From_UnicodeString</td>
<td>LO, LT, PN, SH, ST, UC, UT</td>
</tr>
</tbody>
</table>

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 defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_CHARS_IN_VALUE</td>
<td>A string-type Value had invalid characters for the value representation of Tag.</td>
</tr>
<tr>
<td>MC_INVALID_VALUE_FOR_VR</td>
<td>Value does not conform to the requirements of its value representation.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_TAG</td>
<td>The message does not contain an attribute with an ID of Tag. The attribute can be added with <code>MC_Add_Standard_Attribute</code>. Note: This status will not be returned if setting the value of a message object which was opened using <code>MC_Open_Empty_Message</code> or <code>MC_Create_Empty_File</code>. In that case, the attribute is automatically added to the object before setting the value.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>MsgFileItemID is not a valid message object ID, file object ID or item object ID.</td>
</tr>
<tr>
<td>MC_INCOMPATIBLE_VR</td>
<td>The attribute’s value representation cannot be derived from Value. See the table below.</td>
</tr>
<tr>
<td>MC_VALUE_OUT_OF_RANGE</td>
<td>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).</td>
</tr>
<tr>
<td>MC_TEMP_FILE_ERROR</td>
<td>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.</td>
</tr>
<tr>
<td>MC_INVALID_DATA_TYPE</td>
<td><code>DataType</code> is not valid.</td>
</tr>
<tr>
<td>MC_TOO_MANY_VALUES</td>
<td>This attribute contains the most values that a standard attribute can contain (65535).</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_CANT_COMPLY</td>
<td>The function fails to process the request. Check Merge DICOM Toolkit log for detail.</td>
</tr>
</tbody>
</table>

### INFORMATIONAL MESSAGES:

\[\text{(value was still encoded)}\]

### WARNINGS:

- **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 Merge DICOM Toolkit log file.
- **MC_CANT_COMPLY**: The function fails to process the request. Check Merge DICOM Toolkit log for detail.
Sets the next value of an attribute in a message object to NULL.

**Synopsis**

```c
#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 the enumerated `MC_STATUS` codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INCOMPATIBLE_VR</td>
<td>The attribute identified by <code>Tag</code> can not be set to NULL.</td>
</tr>
<tr>
<td>MC_INVALID_TAG</td>
<td>The message does not contain an attribute identified by <code>Tag</code>. The attribute can be added with <code>MC_Add_Standard_Attribute</code>. Note: This status will not be returned if setting the value of a message object which was opened using <code>MC_Open_Empty_Message</code> or <code>MC_Create_Empty_File</code>. In that case, the attribute is automatically added to the object before setting the value to NULL.</td>
</tr>
</tbody>
</table>
**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 Merge DICOM Toolkit 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**

```c
#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_Float (  
    int MsgFileItemID,  
    char *PrivateCode,  
    unsigned short Group,  
    unsigned char ElementByte,  
    float 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_ShortInt (  
    int MsgFileItemID,  
    char *PrivateCode,  
    unsigned short Group,  
    unsigned char ElementByte,  
    short 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_Int (  
    int MsgFileItemID,
    char *PrivateCode,
    unsigned short Group,
    unsigned char ElementByte,
    int 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_LongInt (  
    int MsgFileItemID,
    char *PrivateCode,
    unsigned short Group,
    unsigned char ElementByte,
    long int Value)

MC_STATUS MC_Set_pValue_From_ULongInt (  
    int MsgFileItemID,
    char *PrivateCode,
    unsigned short Group,
    unsigned char ElementByte,
    unsigned long Value)

MC_STATUS MC_Set_pValue_From_LongLong (  
    int MsgFileItemID,
    char *PrivateCode,
    unsigned short Group,
    unsigned char ElementByte,
    long long Value)

MC_STATUS MC_Set_pValue_From_ULongLong (  
    int MsgFileItemID,
    char *PrivateCode,
    unsigned short Group,
    unsigned char ElementByte,
    unsigned long long 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_UnicodeString (  
    int MsgFileItemID,
    char *PrivateCode,
    unsigned short Group,
    unsigned char ElementByte,
    int ValueLength,
const MC_Uhar* Value
)

MC_STATUS MC_Set_pValue_From_Buffer (  
    int MsgFileItemID,  
    char *PrivateCode,  
    unsigned short Group,  
    unsigned char ElementByte,  
    void *Value  
    unsigned long ValueLength
)

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  
LongLong_Type Binary 64-bit integer number  
ULongLong_Type Binary 64-bit unsigned integer number  
Float_Type Binary Floating point number  
Double_Type Binary Floating point number  
Buffer_Type Binary byte value

Value The attribute should be set to this value.

ValueLength Length of provided 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.

For `MC_Set_pValue_From_UnicodeString`, the `ValueLength` parameter is for specifying the number of Unicode characters in the input `Value`. This function requires `MC_Enable_Unicode_Conversion` being called first. See documentation for `MC_Enable_Unicode_Conversion`.

`MC_Set_pValue_From_Buffer` is intended mainly for setting the value of an attribute whose value representation is unknown. This may occur if `MC_Message_To_Stream` is to be used and one or more of the stream attributes is unknown. The attribute’s value is simply copied (memcpy) from the `Value` buffer according to the length provided in `ValueLength`. The function supports other value representations as well – see table below.

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`.

<table>
<thead>
<tr>
<th>Function</th>
<th>May be used to set values to attributes with these Value Representations</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MC_Set_pValue_From_Float</code></td>
<td>DS, FD, FL, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td><code>MC_Set_pValue_From_Double</code></td>
<td>DS, FD, FL, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td><code>MC_Set_pValue_From_ShortInt</code></td>
<td>DS, FD, FL, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td><code>MC_Set_pValue_From_UShortInt</code></td>
<td>DS, FD, FL, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td><code>MC_Set_pValue_From_Int</code></td>
<td>DS, FD, FL, IS, SS, UL, US, SQ</td>
</tr>
<tr>
<td><code>MC_Set_pValue_From_UInt</code></td>
<td>DS, FD, FL, IS, SS, UL, US, SQ</td>
</tr>
<tr>
<td><code>MC_Set_pValue_From_LongInt</code></td>
<td>AT, DS, FD, FL, IS, SS, UL, SQ</td>
</tr>
<tr>
<td><code>MC_Set_pValue_From_UlongInt</code></td>
<td>AT, DS, FD, FL, IS, SS, UL, SQ</td>
</tr>
<tr>
<td><code>MC_Set_pValue_From_LongLong</code></td>
<td>AT, DS, FD, FL, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td><code>MC_Set_pValue_From_ULongLong</code></td>
<td>AT, DS, FD, FL, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td><code>MC_Set_pValue_From_UnicodeString</code></td>
<td>LO, LT, PN, SH, ST, UC, UT</td>
</tr>
<tr>
<td>Function</td>
<td>May be used to set values to attributes with these Value Representations</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MC_Set_pValue_From_Buffer</td>
<td>UNKNOWN_VR, OB, OW, OV, OL, OD, OF, UR, UT</td>
</tr>
</tbody>
</table>

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 defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
</tbody>
</table>

**INFORMATIONAL MESSAGES:**

**MC_INVALID_TAGS_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 any 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 Merge DICOM Toolkit log file.

See Also
MC_Set_Value... Functions

MC_Set_pValue_From_Function
Sets the value of a message object private attribute which has the binary value representation of OB, OW, OV, OL, OD or OF, or the numeric value representation of AT, SS, US, SL, UL, SV, UV, FL or FD, or the text value representation of UR or UT.

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:

```c
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 Merge DICOM Toolkit 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 UN, OB, OW, OV, OL, OF, AT, SS, US, SL, UL, SV, UV, FL, FD, UR or UT. 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 Merge DICOM Toolkit, 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.
Merge DICOM Toolkit 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 defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MC_NORMAL_COMPLETION</code></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><code>MC_NULL_POINTER_PARM</code></td>
<td><code>PrivateCode</code> or <code>YourSetFunction</code> was NULL.</td>
</tr>
<tr>
<td><code>MC_INVALID_MESSAGE_ID</code></td>
<td><code>MsgFileItemID</code> is not a valid message object ID, file object ID or item object ID.</td>
</tr>
<tr>
<td><code>MC_INVALID_TAG</code></td>
<td>The message does not contain an attribute identified by <code>ElementByte</code> in <code>Group</code> for <code>PrivateCode</code>.</td>
</tr>
<tr>
<td><code>MC_INVALID_GROUP</code></td>
<td><code>Group</code> was not an odd number.</td>
</tr>
<tr>
<td><code>MC_INVALID_PRIVATE_CODE</code></td>
<td>The message does not contain any attributes in <code>Group</code> for <code>PrivateCode</code>.</td>
</tr>
<tr>
<td><code>MC_INCOMPATIBLE_VR</code></td>
<td>The attribute’s value representation was not one of UN, OB, OW, OV, OL, OD, OF, AT, SS, US, SL, UL, SV, UV, FL, FD, UR or UT.</td>
</tr>
<tr>
<td><code>MC_CALLBACK_PARM_ERROR</code></td>
<td>The value set by <code>YourSetFunction</code> in its <code>*CBdataBufferPtr</code> parameter was NULL and <code>*CBisLastPtr</code> was not set to true (non-zero).</td>
</tr>
<tr>
<td><code>MC_CALLBACK_DATA_SIZE_NEGATIVE</code></td>
<td>The value set by <code>YourSetFunction</code> in its <code>*CBdataSizePtr</code> parameter was a negative number</td>
</tr>
<tr>
<td><code>MC_CALLBACK_DATA_SIZE_UNEVEN</code></td>
<td>The value set by <code>YourSetFunction</code> in its <code>*CBdataSizePtr</code> parameter was an odd number</td>
</tr>
<tr>
<td><code>MC_CALLBACK_CANNOT_COMPLY</code></td>
<td><code>YourSetFunction</code> returned a status other than <code>MC_NORMAL_COMPLETION</code>.</td>
</tr>
<tr>
<td><code>MC_VALUE_TOO_LARGE</code></td>
<td>The value set by <code>YourSetFunction</code> int <code>*CBdataSizePtr</code> was too large. It may not be larger than the largest unsigned int on your machine.</td>
</tr>
<tr>
<td><code>MC_SYSTEM_ERROR</code></td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>
MC_Set_pValue_Representation

Sets a private attribute's value representation code.

Synopsis

```c
#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":

Remarks

`MC_Set_pValue_Representation` is used to change a private attribute’s value representation. 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. This method can be used to facilitate reading of the values for such attributes. This function can also be used to manage the attributes that have more than one possible VR.

The valid VR conversions are:
<table>
<thead>
<tr>
<th>Original VR</th>
<th>New VR</th>
</tr>
</thead>
<tbody>
<tr>
<td>US</td>
<td>US</td>
</tr>
<tr>
<td></td>
<td>SS</td>
</tr>
<tr>
<td></td>
<td>OW</td>
</tr>
<tr>
<td>SS</td>
<td>US</td>
</tr>
<tr>
<td></td>
<td>OW</td>
</tr>
<tr>
<td>OW</td>
<td>US</td>
</tr>
<tr>
<td></td>
<td>SS</td>
</tr>
<tr>
<td>UV</td>
<td>SV</td>
</tr>
<tr>
<td></td>
<td>OV</td>
</tr>
<tr>
<td></td>
<td>OV</td>
</tr>
<tr>
<td>OV</td>
<td>UV</td>
</tr>
<tr>
<td>OV</td>
<td>SV</td>
</tr>
<tr>
<td>OW</td>
<td>OB</td>
</tr>
<tr>
<td>OB</td>
<td>OW</td>
</tr>
<tr>
<td>OV</td>
<td>OB</td>
</tr>
<tr>
<td>OB</td>
<td>OV</td>
</tr>
<tr>
<td>UN</td>
<td>Any VR</td>
</tr>
</tbody>
</table>

Any other conversion will result in an error return status.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td><em>MsgFileItemID</em> is not a valid message object ID, file object ID or item object ID.</td>
</tr>
<tr>
<td>MC_INVALID_PRIVATE_CODE</td>
<td>The message does not contain any attributes in <em>Group</em> for <em>PrivateCode</em>.</td>
</tr>
<tr>
<td>MC_INVALID_GROUP</td>
<td><em>Group</em> was not an odd number.</td>
</tr>
<tr>
<td>MC_INVALID_TAG</td>
<td>The message does not contain an attribute identified by <em>ElementByte</em> in <em>Group</em> for <em>PrivateCode</em>.</td>
</tr>
<tr>
<td>MC_INVALID_VR_CODE</td>
<td><em>ValueRep</em> is invalid.</td>
</tr>
<tr>
<td>MC_ATTRIBUTE_HAS_VALUES</td>
<td>When <em>ValueRep</em> is SQ, the attribute may not already have a value.</td>
</tr>
<tr>
<td>MC_VR_ALREADY_VALID</td>
<td>The attribute already has a valid Value Representation.</td>
</tr>
</tbody>
</table>

**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.Emily 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 defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_PRIVATE_CODE</td>
<td>The message does not contain any attributes in Group for PrivateCode.</td>
</tr>
<tr>
<td>MC_INVALID_GROUP</td>
<td>Group was not an odd number.</td>
</tr>
<tr>
<td>MC_INVALID_TAG</td>
<td>The message does not contain an attribute identified by ElementByte in Group for PrivateCode.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>MsgFileItemID is not a valid message object ID, file object ID or item object ID.</td>
</tr>
</tbody>
</table>

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**

```c
#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 the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_PRIVATE_CODE</td>
<td>The message does not contain any attributes in Group for PrivateCode.</td>
</tr>
<tr>
<td>MC_INVALID_GROUP</td>
<td>Group was not an odd number.</td>
</tr>
<tr>
<td>MC_INVALID_TAG</td>
<td>The message does not contain an attribute identified by ElementByte in Group for PrivateCode.</td>
</tr>
</tbody>
</table>
MC_INVALID_MESSAGE_ID

`MsgFileItemID` is not a valid message object ID, file object ID or item object ID.

### See Also

<table>
<thead>
<tr>
<th><code>MC_Set_pValue_To_Empty</code></th>
<th><code>MC_Set_Value_To_Empty</code></th>
</tr>
</thead>
</table>

### MC_Set_Service_Command

Associates a Merge DICOM Toolkit message or file object with a given DICOM service/command pair.

#### Synopsis

```c
#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 Merge DICOM Toolkit 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 Merge DICOM Toolkit:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C_STORE_RQ</td>
<td>DICOM Composite Store Service Request</td>
</tr>
</tbody>
</table>
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 Response
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 the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>ServiceName was NULL.</td>
</tr>
</tbody>
</table>

© Copyright Merge Healthcare Solutions Inc. 2023
### 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 Merge DICOM Toolkit 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 toolkit configuration parameter at runtime.

#### Synopsis

```c
#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:

- MERGECOM_3_PROFILE
- MERGECOM_3_SERVICES
- MERGECOM_3_APPLICATIONS
- CAPTURE_FILE
- COMPRESSION_RGB_TRANSFORM_FORMAT
- DECODER_TAG_FILTER
- DEFLATED_EXPLICIT_LITTLE_ENDIAN_SYNTAX
- DICTIONARY_ACCESS
- DICTIONARY_FILE
- ENCAPSULATED_UNCOMPRESSED_ELE_SYNTAX
- EXPLICIT_BIG_ENDIAN_SYNTAX
- EXPLICIT_LITTLE_ENDIAN_SYNTAX
- HEVC_H265_M10P_LEVEL_5_1_SYNTAX
- HEVC_H265_MP_LEVEL_5_1_SYNTAX
- IMPLEMENTATION_CLASS_UID
- IMPLEMENTATION_VERSION
- IMPLICIT_BIG_ENDIAN_SYNTAX
- IMPLICIT_LITTLE_ENDIAN_SYNTAX
- INITIATOR_NAME
- IP_TYPE
- JPEG_2000LOSSLESSONLY_SYNTAX
- JPEG_2000_MC_LOSSLESSONLY_SYNTAX
- JPEG_2000_MC_SYNTAX
- JPEG_2000_SYNTAX
- JPEG_BASELINE_SYNTAX
- JPEG_EXTENDED_2_4_SYNTAX
- JPEG_EXTENDED_3_5_SYNTAX
- JPEG_EXTENDED_HIER_16_18_SYNTAX
- JPEG_EXTENDED_HIER_17_19_SYNTAX
- JPEG_FULL_PROG_HIER_24_26_SYNTAX
- JPEG_FULL_PROG_HIER_25_27_SYNTAX
- JPEG_FULL_PROG_NON_HIER_10_12_SYNTAX
- JPEG_FULL_PROG_NON_HIER_11_13_SYNTAX
- JPEG_LOSSLESS_HIER_14_SYNTAX
- JPEG_LOSSLESS_HIER_28_SYNTAX
- JPEG_LOSSLESS_HIER_29_SYNTAX
- JPEG_LOSSLESS_NON_HIER_14_SYNTAX
- JPEG_LOSSLESS_NON_HIER_15_SYNTAX
- JPEG_LS_LOSSLESS_SYNTAX
- JPEG_LS_LOSSY_SYNTAX
- JPEG_SPEC_HIER_20_22_SYNTAX
- JPEG_SPEC_HIER_21_23_SYNTAX
- JPEG_SPEC_NON_HIER_6_8_SYNTAX
- JPEG_SPEC_NON_HIER_7_9_SYNTAX
- JPIP_REFERENCED_DEFLATE_SYNTAX
- JPIP_REFERENCED_SYNTAX
- LARGE_DATA_STORE
- LICENSE
- LOCAL_APPL_CONTEXT_NAME
- LOG_FILE
- MPEG2_MPHEL_SYNTAX
MPEG2_MPML_SYNTAX
MPEG4_AVC_H264_BDC_HP_LEVEL_4_1_SYNTAX
MPEG4_AVC_H264_HP_LEVEL_4_1_SYNTAX
MPEG4_AVC_H264_HP_LEVEL_4_2_2D_SYNTAX
MPEG4_AVC_H264_HP_LEVEL_4_2_3D_SYNTAX
MPEG4_AVC_H264_STEREO_HP_LEVEL_4_2_SYNTAX
MSG_INFO_FILE
NULL_TYPE3_VALIDATION
PEGASUS_DISP_REG_NAME
PEGASUS_DISP_REGISTRATION
PEGASUS_OP_D2SEPLUS_NAME
PEGASUS_OP_D2SEPLUS_REGISTRATION
PEGASUS_OP_J2KE_NAME
PEGASUS_OP_J2KE_REGISTRATION
PEGASUS_OP_J2KP_NAME
PEGASUS_OP_J2KP_REGISTRATION
PEGASUS_OP_JLSE_NAME
PEGASUS_OP_JLSE_REGISTRATION
PEGASUS_OP_JLSP_NAME
PEGASUS_OP_JLSP_REGISTRATION
PEGASUS_OP_LIE3PLUS_NAME
PEGASUS_OP_LIE3PLUS_REGISTRATION
PEGASUS_OP_LIP3PLUS_NAME
PEGASUS_OP_LIP3PLUS_REGISTRATION
PEGASUS_OP_SE2DPLUS_NAME
PEGASUS_OP_SE2DPLUS_REGISTRATION
PEGASUS_OPCODE_PATH
PRIVATE_SYNTAX_1_SYNTAX
PRIVATE_SYNTAX_2_SYNTAX
RECEIVER_NAME
RLE_SYNTAX
SMPTE_ST_2110_20_UNCOMPRESSED_PROGRESSIVE_ACTIVE_VIDEO_SYNTAX
SMPTE_ST_2110_20_UNCOMPRESSED_INTERLACED_ACTIVE_VIDEO_SYNTAX
SMPTE_ST_2110_30_PCM_DIGITAL_AUDIO_SYNTAX
TEMP_FILE_DIRECTORY
UNKNOWN_VR_CODE

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

Avalue The character string value to which Aparm is to be set.

Remarks
The Merge DICOM Toolkit 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 toolkit configuration elsewhere in this manual.

Return Value
One of the 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 | A value 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_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_Set_Value... Functions
Sets the value of an attribute in a message object.

#### Synopsis
```c
#include "mc3msg.h"

MC_STATUS MC_Set_Value ( 
    int MsgFileItemID,
    unsigned long Tag,
    MC_DT DataType,
    void *Value
)

MC_STATUS MC_Set_Value_From_Float ( 
    int MsgFileItemID,
    unsigned long Tag,
    float Value
)

MC_STATUS MC_Set_Value_From_Double ( 
    int MsgFileItemID,
    unsigned long Tag,
    double Value
)

MC_STATUS MC_Set_Value_From_ShortInt ( 
    int MsgFileItemID,
    unsigned long Tag,
    short int Value
)

MC_STATUS MC_Set_Value_From_UShortInt ( 
    int MsgFileItemID,
    unsigned long Tag,
    unsigned short Value
)

MC_STATUS MC_Set_Value_From_Int ( 
    int MsgFileItemID,
    unsigned long Tag,
    int Value
)```
MC_STATUS MC_Set_Value_From_UInt (int MsgFileItemID, unsigned long Tag, unsigned int Value)

MC_STATUS MC_Set_Value_From_LongInt (int MsgFileItemID, unsigned long Tag, long int Value)

MC_STATUS MC_Set_Value_From_ULongInt (int MsgFileItemID, unsigned long Tag, unsigned long Value)

MC_STATUS MC_Set_Value_From_LongLong (int MsgFileItemID, unsigned long Tag, long long Value)

MC_STATUS MC_Set_Value_From_ULongLong (int MsgFileItemID, unsigned long Tag, unsigned long long Value)

MC_STATUS MC_Set_Value_From_String (int MsgFileItemID, unsigned long Tag, char* Value)

MC_STATUS MC_Set_Value_From_UnicodeString (int MsgFileItemID, unsigned long Tag, int ValueLength, const MC_Uhar* Value)

MC_STATUS MC_Set_Value_From_Buffer (int MsgFileItemID, unsigned long Tag, void *Value, unsigned long ValueLength)

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
LongLong_Type  Binary 64-bit integer number
UlongLong_Type Binary 64-bit unsigned integer number
Float_Type     Binary Floating point number
Double_Type    Binary Floating point number
Buffer_Type    Binary byte value

Value    The value to be used as the attribute’s value.
ValueLength    Length of provided 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.

For MC_Set_Value_From_Unicodestring, the ValueLength parameter is for specifying the number of Unicode characters in the input Value. This function requires MC_Enable_Unicode_Conversion being called first. See documentation for MC_Enable_Unicode_Conversion.

MC_Set_Value_From_Buffer is intended mainly for setting the value of an attribute whose value representation is unknown. This may occur if MC_Message_To_Stream is to be used and one or more of the stream attributes is unknown. The attribute’s value is simply copied (memcpy) from the Value buffer according to the length provided in ValueLength. The function supports other value representations as well – see table below.

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.
<table>
<thead>
<tr>
<th>Function</th>
<th>May be used to set values to attributes with these Value Representations</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_Set_Value_From_Float</td>
<td>DS, FD, FL, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Set_Value_From_Double</td>
<td>DS, FD, FL, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Set_Value_From_ShortInt</td>
<td>DS, FD, FL, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Set_Value_From_UShortInt</td>
<td>DS, FD, FL, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Set_Value_From_Int</td>
<td>DS, FD, FL, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Set_Value_From_UINT</td>
<td>DS, FD, FL, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Set_Value_From_LongInt</td>
<td>AT, DS, FD, FL, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Set_Value_From_ULongInt</td>
<td>AT, DS, FD, FL, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Set_Value_From_LongLong</td>
<td>AT, DS, FD, FL, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Set_Value_From_ULongLong</td>
<td>AT, DS, FD, FL, IS, SS, SV, UL, US, UV, SQ</td>
</tr>
<tr>
<td>MC_Set_Value_From_UnicodeString</td>
<td>LO, LT, PN, SH, ST, UC, UT</td>
</tr>
<tr>
<td>MC_Set_Value_From_Buffer</td>
<td>UNKNOWN_VR, OB, OW, OV, OL, OD, OF, UR, UT</td>
</tr>
<tr>
<td>MC_Set_Value_From_Function</td>
<td>UNKNOWN_VR, OB, OW, OV, OL, OD, OF, AT, SS, US, SL, UL, SV, UV, FL, FD, UR, UT</td>
</tr>
</tbody>
</table>

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 defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>INFORMATIONAL MESSAGES:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>WARNINGS:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>MC_INVALID_CHARS_IN_VALUE</strong></td>
<td>A string-type <code>Value</code> had invalid characters for the value representation of <code>Tag</code>.</td>
</tr>
</tbody>
</table>
**MC_INVALID_VALUE_FOR_VR**  
*Value* does not conform to the requirements of its value representation.

**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 Merge DICOM Toolkit log file.

**MC_CANNOT_COMPLY**  
The function fails to process the request. Check Merge DICOM Toolkit log for detail.

**See Also**

- **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_Enable_Unicode_Conversion**
MC_Set_Value_From_Function

Sets the value of a message object attribute which has the binary value representation of OB, OW, OV, OL, OD or OF, or the numeric value representation of AT, SS, US, SL, UL, SV, UV, FL or FD, or the text value representation of UR or UT.

**Synopsis**

```c
#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:

```c
MC_STATUS YourSetFunction (  
    int(CBMsgFileItemID),  
    unsigned long(Cbtag),  
    int(CbrisFirst),  
    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.

- **CbrisFirst**
  This is TRUE (non-zero) the first time Merge DICOM Toolkit 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, OV, OD 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 UN, OB, OW, OV, OL, OD, OF, AT, SS, US, SL, UL, SV, UV, FL, FD, UR or UT. 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 Merge DICOM Toolkit, 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.

Merge DICOM Toolkit 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 defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_TAG</td>
<td>The message does not contain an attribute with an ID of <strong>Tag</strong>. The attribute can be added with <strong>MC_Add_Standard_Attribute</strong>.</td>
</tr>
<tr>
<td>Note: This status will not be returned if setting the value of a message object which was opened using <strong>MC_Open_Empty_Message</strong> or <strong>MC_Create_Empty_File</strong>. In that case, the attribute is automatically added to the object before setting the value.</td>
<td></td>
</tr>
</tbody>
</table>

© Copyright Merge Healthcare Solutions Inc. 2023
### 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 one of UN, OB, OW, OV, OL, OD, OF, AT, SS, US, SL, UL, SV, UV, FL, FD, UR or UT.

### 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_CANT_COMPLY`.

### See Also

- MC_Set_Value... Functions
- MC_Set_Next_Value... Functions

### MC_Set_Value_Representation

Sets the value representation code of an attribute.

#### Synopsis

```c
#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":  

#### Remarks

`MC_Set_Value_Representation` is used to change an attribute’s value representation.

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. This method can be used to facilitate reading of the values for such attributes.

This function can also be used to manage the attributes that have more than one possible VR.
The valid VR conversions are:

<table>
<thead>
<tr>
<th>Original VR</th>
<th>New VR</th>
</tr>
</thead>
<tbody>
<tr>
<td>US</td>
<td>US</td>
</tr>
<tr>
<td></td>
<td>SS</td>
</tr>
<tr>
<td></td>
<td>OW</td>
</tr>
<tr>
<td>SS</td>
<td>US</td>
</tr>
<tr>
<td>SS</td>
<td>OW</td>
</tr>
<tr>
<td>OW</td>
<td>US</td>
</tr>
<tr>
<td>OW</td>
<td>SS</td>
</tr>
<tr>
<td>UV</td>
<td>SV</td>
</tr>
<tr>
<td></td>
<td>OV</td>
</tr>
<tr>
<td></td>
<td>OV</td>
</tr>
<tr>
<td>OV</td>
<td>UV</td>
</tr>
<tr>
<td>OV</td>
<td>OV</td>
</tr>
<tr>
<td>OW</td>
<td>OB</td>
</tr>
<tr>
<td>OB</td>
<td>OW</td>
</tr>
<tr>
<td>OB</td>
<td>OB</td>
</tr>
<tr>
<td>OB</td>
<td>OV</td>
</tr>
<tr>
<td>UN</td>
<td>Any VR</td>
</tr>
</tbody>
</table>

Any other conversion will result in an error return status.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_INVALID_MESSAGE_ID</strong></td>
<td><em>MsgFileItemID</em> is not a valid message object ID, file object ID or item object ID.</td>
</tr>
<tr>
<td><strong>MC_INVALID_TAG</strong></td>
<td>The message does not contain an attribute with an ID of <em>Tag</em>.</td>
</tr>
<tr>
<td><strong>MC_INVALID_VR_CODE</strong></td>
<td><em>ValueRep</em> is invalid.</td>
</tr>
<tr>
<td><strong>MC_ATTRIBUTE_HAS_VALUES</strong></td>
<td>When <em>ValueRep</em> is SQ, the attribute may not already have a value.</td>
</tr>
<tr>
<td><strong>MC_VR_ALREADY_VALID</strong></td>
<td>The attribute already has a valid Value Representation.</td>
</tr>
</tbody>
</table>

**See Also**

MC_Set_pValue_Representation

**MC_Set_Value_To_Empty**

Removes all values from an attribute in a message object.

**Synopsis**

```cpp
#include "mc3msg.h"
```
MC_STATUS MC_Set_Value_To_Empty (  
   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_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 defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_TAG</td>
<td>The message does not contain an attribute with an ID of Tag.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>MsgFileItemID is not a valid message object ID, file object ID or item object ID.</td>
</tr>
</tbody>
</table>

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

```c
#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 defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_TAG</td>
<td>The message does not contain an attribute with an ID of Tag. The attribute can be added with <strong>MC_Add_Standard_Attribute</strong>. Note: This status will not be returned if setting the value of a message object which was opened using <strong>MC_Open_Empty_Message</strong> or <strong>MC_Create_Empty_File</strong>. In that case, the attribute is automatically added to the object before setting the value to NULL.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>MsgFileItemID is not a valid message object ID, file object ID or item object ID.</td>
</tr>
</tbody>
</table>

See Also

- MC_Set_Value_To_Empty
- MC_Set_pValue_To_Null
- MC_Set_pValue_To_Empty
**MC_SR_Add_Child**

Adds a lower level SR node to a specified SR tree management.

**Synopsis**

```c
#include "mc3media.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 Merge DICOM Toolkit.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The <code>SRID</code> value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_INVALID_ITEM_ID</td>
<td>The <code>ItemID</code> value is not a valid SR tree management record object ID.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**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**

```c
#include "mc3media.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 a 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 the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The SRID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>The SRID parameter was NULL.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

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 “mc3media.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 removes the specified child SR object and all of its associated child objects from the SR tree management structures. It also frees the system resources used by the specified child object and all of its child SR object(s).
**MC_SR_Delete_Child** updates all of the location values for affected SR objects within the SR tree.

**Return Value**
One of the enumerated **MC_STATUS** codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The <code>SRID</code> value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**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_Child_Count

**MC_SR_Get_Child_Count**

Returns the number of children of an SR tree management node.

**Synopsis**

```c
#include “mc3media.h”

MC_STATUS MC_SR_First_Child (  
    int SRID,  
    int *Count  
)
```

- **SRID** The identifier assigned to this SR tree management node.
- **Count** Upon successful completion, this parameter returns the number of children for `SRID`

**Remarks**

**MC_SR_Get_Child_Count** returns the number of children of the SR tree management item `ItemID`.

**Return Value**
One of the enumerated **MC_STATUS** codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The <code>SRID</code> value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>The <code>Count</code> parameter was NULL.</td>
</tr>
<tr>
<td>MC_UNABLE_TO_GET_SR_ID</td>
<td>An internal error was encountered, Merge DICOM Toolkit was unable to retrieve the ID of an SR object.</td>
</tr>
</tbody>
</table>
## 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

```c
#include "mc3media.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 the enumerated `MC_STATUS` codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The <code>SRID</code> value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ITEM_ID</td>
<td>The <code>ItemID</code> value is not a valid SR tree management record object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>The <code>ItemID</code>, or <code>IsLast</code> parameter was NULL.</td>
</tr>
<tr>
<td>MC_MSGFILE_ERROR</td>
<td>An error occurred attempting to access the configuration data file for the directory record.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>
## MC_SR_Get_Location

**Synopsis**

```c
#include "mc3media.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 the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The <code>SRID</code> value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>The <code>Location</code> parameter was NULL.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

### See Also

- MC_SR_Add_Child
- MC_SR_Get_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 "mc3media.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.

**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 the enumerated MC_STATUS codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The SRID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>The NextChildID or IsLast parameter was NULL.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the</td>
</tr>
<tr>
<td></td>
<td>operating environment. A message describing the error</td>
</tr>
<tr>
<td></td>
<td>has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**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**

```c
#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 the enumerated `MC_STATUS` codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The <code>MsgID</code> value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>The <code>SRID</code> parameter was NULL.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**See Also**

- `MC_SR_Add_Child`
- `MC_SR_Get_First_Child`
- `MC_SR_Get_Next_Child`
- `MC_SR_Delete_Child`
- `MC_SR_To_Message`  
  Converts a toolkit-managed SR tree object into a toolkit-managed message object.

**Synopsis**

```c
#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 the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The SRID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

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

```c
#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
- ENCAPSULATED_UNCOMPRESSED_ENDIAN
- 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
- HEVC_H265_M10P_LEVEL_5_1
- HEVC_H265_MP_LEVEL_5_1
- JPIP_REFERENCED
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:

```c
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 Merge DICOM Toolkit 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 Merge DICOM Toolkit by YourStreamToFunction. Merge DICOM Toolkit 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. Merge DICOM Toolkit passes the address to YourStreamToFunction in CbuserInfo each time it is called. UserInfo may be NULL.

If an error occurs while processing the stream, Merge DICOM Toolkit 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 ItemIDs 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 VRs 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 Merge DICOM Toolkit each time it is called. As a convenience, Merge DICOM Toolkit 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 the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td><code>YourStreamToFunction</code> or <code>ErrorTagPtr</code> was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>The <code>MessageID</code> value is not a valid message object ID.</td>
</tr>
<tr>
<td>MC_INVALID_TRANSFER_SYNTAX</td>
<td>An invalid code was used for the <code>SyntaxType</code> parameter.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_CALLBACK_CANNOT_COMPLY</td>
<td><em>YourStreamToFunction</em> returned with a status other than <code>MC_NORMAL_COMPLETION</code>.</td>
</tr>
<tr>
<td>MC_CALLBACK_DATA_SIZE_UNEVEN</td>
<td>The <code>CbdataSizePtr</code> parameter returned by the <code>YourStreamToFunction</code> was an uneven number.</td>
</tr>
<tr>
<td>MC_CALLBACK_PARM_ERROR</td>
<td>A callback function registered by your application returned an empty (NULL) data buffer when the buffers length was specified as being non-zero. See <code>MC_Register_Callback_Function</code>.</td>
</tr>
<tr>
<td>Error Code</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MC_OUT_OF_ORDER_TAG</td>
<td>A tag was found in the file that was not in ascending order. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_INVALID_TAG</td>
<td>The message contains an invalid tag. The tag is placed at &quot;ErrorTagPtr.&quot;</td>
</tr>
<tr>
<td>MC_VALUE_TOO_LARGE</td>
<td>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.</td>
</tr>
<tr>
<td>MC_UNEXPECTED_EOD</td>
<td>YourStreamToFunction stopped without passing the entire value for an attribute.</td>
</tr>
<tr>
<td>MC_INVALID_LENGTH_FOR_VR</td>
<td>The value(s) for one of the stream attributes was not legal for its value representation.</td>
</tr>
</tbody>
</table>

Any of the status codes returned by MC_Set_Value may also be returned.

See Also
MC_Register_Callback_Function  MC_Message_To_Stream

**MC_Thread_Release**

Releases thread specific resources used by the Merge DICOM Toolkit library.

**Synopsis**

```c
#include "mc3msg.h"

MC_STATUS MC_Thread_Release ( )
```

**Remarks**

MC_Thread_Release releases resources allocated for the calling thread.

MC_Thread_Release should be called before the end of each thread that uses either MC_Standard_Compressor or MC_Standard_Decompressor to avoid memory leaks in Pegasus libraries from Accusoft.

This call has no effect on platforms that do not support threads.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_LIBRARY_NOT_INITIALIZED</td>
<td>The library has not been initialized by a call to MC_Library_INITIALIZATION.</td>
</tr>
</tbody>
</table>

See Also
MC_Library_Initialization
**MC_Unic peace_Get_Substitution_Characters**

A utility function to retrieve the replacement characters for a DICOM character set encoder.

**Synopsis**

```c
#include "mc3msg.h"

MC_STATUS MC_Unic peace_Get_Substitution_Characters ( 
    char *Charset, 
    char *SubChars, 
    int *Length 
)
```

- **Charset** The DICOM specific character set name
- **SubChars** The substitution characters
- **Length** On input the capacity of SubChars, on output the number of bytes copied to it

**Remarks**

**MC_Unic peace_Get_Substitution_Characters** requires **MC_Enable_Unicode_Conversion** being called first.

**Charset** should be set to a value corresponding to one of the DICOM defined terms for the Specific Character Set (0008,0005).

When returns status is MC_NORMAL_COMPLETION, **SubChars** contains the substitution characters as a byte array. **Length** contains how many bytes are in the **SubChars** buffer.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_CANNOT_COMPLY</td>
<td>Fail to process the input. Check the Merge DICOM Toolkit log file for detail of failure.</td>
</tr>
<tr>
<td>MC_BUFFER_TOO_SMALL</td>
<td>The output buffer doesn’t have enough space to hold the output.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>Any of the arguments is null. An unexpected, potentially serious problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**See Also**

- **MC_B yte_To_Unicode**
- **MC_Enable_Unicode_Conversion**
- **MC_Get_Value_To_UnicodeString**
- **MC_G et_Next_Value_To_UnicodeString**
- **MC_Set_Value_From_UnicodeString**
- **MC_S et_Next_Value_From_UnicodeString**
- **MC_Unic peace_To_Byte**
## MC_Unicode_To_Byte

A utility function to convert Unicode to DICOM character set.

### Synopsis

```c
#include "mc3msg.h"

MC_STATUS MC_Unicode_To_Byte (char **Charsets, int NumCharsets, const MC_UChar *Val, int InLen, int BufferSize, int *OutLen, char *OutVal)
```

- **Charsets**: An array of character strings from the DICOM specific character set attribute.
- **NumCharsets**: Number of character set strings in the **Charsets** array.
- **Val**: Input Unicode array.
- **InLen**: Input Unicode character count.
- **BufferSize**: Output buffer size.
- **OutLen**: Output length (returned by this call).
- **OutVal**: Output buffer to hold the byte.

### Remarks

**MC_Unicode_To_Byte** requires **MC_Enable_Unicode_Conversion** being called first.

**Charsets** should be set to the value of (0008,0005) as an array of character strings. If **NumCharsets** is 0, **Charsets** is NULL or an empty string, the default ISO_IR 6 (ASCII) character set will be used.

**InLen** can be set to -1 and the toolkit will calculate length if the input Unicode buffer is U+0000 terminated.

When the return status is **MC_NORMAL_COMPLETION**, **OutVal** contains the output multi-byte characters with a NULL terminator. **OutLen** contains the number of bytes present in the **OutVal** buffer (excluding the terminator).

### Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_CANNOT_COMPLY</strong></td>
<td>Fail to process the input. Check the Merge DICOM Toolkit log file for details of failure.</td>
</tr>
<tr>
<td><strong>MC_BUFFER_TOO_SMALL</strong></td>
<td>The output buffer doesn’t have enough space to hold the output.</td>
</tr>
</tbody>
</table>
MC_SYSTEM_ERROR

An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.

See Also

MC_Byte_To_Unicode
MC_Enable_Unicode_Conversion
MC_Get_Value_To_UnicodeString
MC_Get_Next_Value_To_UnicodeString
MC_Set_Value_From_UnicodeString
MC_Set_Next_Value_From_UnicodeString

MC_Validate_Attribute

Ensures that an attribute meets DICOM validation criteria.

Synopsis

```cpp
#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”:

```cpp
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 */
    int ParentMsgID; /* ID of parent of message or item object containing the attribute */
    int MsgLevel; /* Nesting level in data set hierarchy of the message or item object containing the attribute */
} 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 ensure 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.

**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 Merge DICOM Toolkit 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 the enumerated **MC_STATUS** codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_MESSAGE_VALIDATES</td>
<td>The attribute contained no validation violations of the type specified in ErrorLevel.</td>
</tr>
<tr>
<td>MC_DOES_NOT_VALIDATE</td>
<td>The attribute contained at least one validation violation of the type specified in ErrorLevel.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>MessageFileID does not identify a valid message or file object.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_TAG</td>
<td>Tag was not in the message.</td>
</tr>
<tr>
<td>MC_MSGFILE_ERROR</td>
<td>An error occurred involving the message info file for the file object. The exact error is logged in the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_INVALID_ITEM_ID</td>
<td>The message or file object contains a sequence of items attribute with an invalid item ID value.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**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:**

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NO_CONDITION_FUNCTION</td>
<td>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.</td>
</tr>
<tr>
<td>MC_UNABLE_TO_CHECK_CONDITION</td>
<td>Not enough information is available to check whether or not a DICOM type “1C” or “2C” attribute is required.</td>
</tr>
<tr>
<td>MC_NOT_ONE_OF_DEFINED_TERMS</td>
<td>A value set for this attribute is not one of the valid defined terms assigned to this attribute.</td>
</tr>
<tr>
<td>MC_NON_SERVICE_ATTRIBUTE</td>
<td>The attribute is not one of those defined for the file’s service. Note that private attributes will not cause this violation</td>
</tr>
</tbody>
</table>

**ERRORS:**

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_INVALID_VR_CODE</td>
<td>The attribute’s value representation is unknown.</td>
</tr>
<tr>
<td>MC_REQUIRED_ATTRIBUTE_MISSING</td>
<td>An attribute which is required for the message service has been deleted from the file object.</td>
</tr>
<tr>
<td>MC_REQUIRED_VALUE_MISSING</td>
<td>The attribute is required to have a value and does not.</td>
</tr>
<tr>
<td>MC_VALUE_MAY_NOT_BE_NULL</td>
<td>The attribute is DICOM type “1” or type “1C” and it has been encoded with a NULL value.</td>
</tr>
</tbody>
</table>
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
Ensures 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 */
  int ParentMsgID; /* ID of parent of message or item object containing the attribute */
  int MsgLevel; /* Nesting level in data set hierarchy of the message or item object containing the attribute */
}
Validating a file

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 ensure 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 Merge DICOM Toolkit 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 the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
</table>
| VAL_ERR              | The level of validation checking is specified here. Use one of the enumerated VAL_LEVEL types defined in “mc3msg.h”:
<p>| Validation_Level1:   | Report only Errors.          |
| Validation_Level2:   | Report Errors and Warnings.  |
| Validation_Level3:   | Report Errors, Warnings and Info messages. |
| 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. |</p>
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_MESSAGE_VALIDATES</td>
<td>The file object contained no validation violations of the type specified in <strong>ErrorLevel</strong>.</td>
</tr>
<tr>
<td>MC_DOES_NOT_VALIDATE</td>
<td>The file object contained at least one validation violation of the type specified in <strong>ErrorLevel</strong>.</td>
</tr>
<tr>
<td>MC_INVALID_FILE_ID</td>
<td><strong>FileID</strong> does not identify a valid file object.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_MSGFILE_ERROR</td>
<td>An error occurred involving the message info file for the file object. The exact error is logged in the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_INVALID_ITEM_ID</td>
<td>The file object contains a sequence of items attribute with an invalid item ID value.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**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:**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NO_CONDITION_FUNCTION</td>
<td>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.</td>
</tr>
<tr>
<td>MC_UNABLE_TO_CHECK_CONDITION</td>
<td>Not enough information is available to check whether or not a DICOM type “1C” or “2C” attribute is required.</td>
</tr>
</tbody>
</table>

**WARNINGS:**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NOT_ONE_OF_DEFINED_TERMS</td>
<td>A value set for this attribute is not one of the valid defined terms assigned to this attribute.</td>
</tr>
<tr>
<td>MC_NON_SERVICE_ATTRIBUTE</td>
<td>The attribute is not one of those defined for the file’s service. Note that private attributes will not cause this violation</td>
</tr>
</tbody>
</table>

**ERRORS:**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_INVALID_VR_CODE</td>
<td>The attribute’s value representation is unknown.</td>
</tr>
<tr>
<td>MC_REQUIRED_ATTRIBUTEMISSING</td>
<td>An attribute which is required for the message service has been deleted from the file object.</td>
</tr>
<tr>
<td>MC_REQUIRED_VALUE_MISSING</td>
<td>The attribute is required to have a value and does not.</td>
</tr>
<tr>
<td>MC_VALUE_MAY_NOT_BE_NULL</td>
<td>The attribute is DICOM type “1” or type “1C” and it has been encoded with a NULL value.</td>
</tr>
</tbody>
</table>
**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
Ensures that a message meets DICOM validation criteria.

#### Synopsis

```c
#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”:

```c
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 */
    int ParentMsgID;  /* ID of parent of message or item */
} ValErr;
```
int MsgLevel; /* Nesting level in data set hierarchy of the message or item object containing the attribute */
}

### 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 ensure 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 Merge DICOM Toolkit 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_VALIDATE` 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 the enumerated `MC_STATUS` codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_MESSAGE_VALIDATE</td>
<td>The message contained no validation violations of the type specified in <code>ErrorLevel</code>.</td>
</tr>
<tr>
<td>MC_DOES_NOT_VALIDATE</td>
<td>The message contained at least one validation violation of the type specified in <code>ErrorLevel</code>.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td><code>MessageID</code> does not identify a valid message object.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_MSGFILE_ERROR</td>
<td>An error occurred involving the message info file for the message. The exact error is logged in the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_INVALID_ITEM_ID</td>
<td>The message contains a sequence of items attribute with an invalid item ID value.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**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:**

<table>
<thead>
<tr>
<th>Message</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NO_CONDITION_FUNCTION</td>
<td>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.</td>
</tr>
<tr>
<td>MC_UNABLE_TO_CHECK_CONDITION</td>
<td>Not enough information is available to check whether or not a DICOM type “1C” or “2C” attribute is required.</td>
</tr>
</tbody>
</table>

**WARNINGS:**

<table>
<thead>
<tr>
<th>Message</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NOT_ONE_OF_DEFINED_TERMS</td>
<td>A value set for this attribute is not one of the valid defined terms assigned to this attribute.</td>
</tr>
<tr>
<td>MC_NON_SERVICE_ATTRIBUTE</td>
<td>The attribute is not one of those defined for the message’s service. Note that private attributes will not cause this violation</td>
</tr>
</tbody>
</table>

**ERRORS:**

<table>
<thead>
<tr>
<th>Message</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_INVALID_VR_CODE</td>
<td>The attribute’s value representation is unknown.</td>
</tr>
</tbody>
</table>
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
MC_Wait_For_Association_On_Address

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  
)
MC_STATUS MC_Wait_For_Association_On_Address (  
    char *ServiceList,  
    int Timeout,  
    int ApplicationID,  
    int Port,  
    const char *Address,  
    int *AssociationID  
)

_ServiceList_ Name of a service list in the Merge DICOM Toolkit configuration file.

_Timeout_ The maximum 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.

_Address_ The network name or IP address of the network interface to listen on. If this argument is NULL the toolkit will accept connections on all available network interfaces.

**Remarks**

_MC_Wait_For_Association, MC_Wait_For_Association_On_Port_ and _MC_Wait_For_Association_On_Address_ 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_, while _MC_Wait_For_Association_On_Port_ and _MC_Wait_For_Association_On_Address_ will only negotiate associations for the specific Application passed to the call.

If _Timeout_ is a positive number Merge DICOM Toolkit will wait no longer than _Timeout_ seconds for an association request. If _Timeout_ is negative, Merge DICOM Toolkit will not return until an association request has arrived or an error occurs. If _Timeout_ is zero, Merge DICOM Toolkit 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 Merge DICOM Toolkit 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_ or _MC_Wait_For_Association_On_Address_, Merge DICOM Toolkit rejects the association request and
continues to wait for a valid association request. Optionally, "ACCEPT_ANY_APPLICATION_TITLE = YES" may be specified in the Merge DICOM Toolkit 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.

Merge DICOM Toolkit validates each received association request based on the protocol rules of DICOM and on configurable specifications contained in the Merge DICOM Toolkit 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 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 the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_TIMEOUT</td>
<td>The Timeout period expired.</td>
</tr>
<tr>
<td>MC_NEGOTIATION_ABORTED</td>
<td>An incoming association was aborted during negotiation. Normally this situation is handled by retrying the MC_Wait_For_Association call.</td>
</tr>
<tr>
<td>MC_ASSOCIATION_REJECTED</td>
<td>An incoming association was aborted because no services were acceptable.</td>
</tr>
<tr>
<td>MC_SYSTEM_CALL_INTERRUPTED</td>
<td>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.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>AssociationID, ApplicationID or ServiceList parameter was NULL.</td>
</tr>
<tr>
<td>MC_NO_APPLICATIONS_REGISTERED</td>
<td>No applications have been registered using MC_Register_Application.</td>
</tr>
<tr>
<td>MC_INVALID_SERVICE_LIST_NAME</td>
<td>ServiceList points at a null string.</td>
</tr>
<tr>
<td>MC_CONFIG_INFO_MISSING</td>
<td>Could not access ServiceList configuration parameters.</td>
</tr>
<tr>
<td>MC_CONFIG_INFO_ERROR</td>
<td>The ServiceList contained too many services.</td>
</tr>
</tbody>
</table>
**MC_SYSTEM_ERROR**  An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit 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_Connection**  
- **MC_Close_Listen_Port_On_Address**  
- **MC_Wait_For_Connection_On_Port**  
- **MC_Process_Association_Request**  
- **MC_Wait_For_Connection_On_Address**  
- **MC_Process_Secure_Association_Request**  
- **MC_Wait_For_Secure_Association_On_Address**

### MC_Wait_For_Connection

- **MC_Wait_For_Connection_On_Port**  
- **MC_Wait_For_Connection_On_Address**

Waits for an incoming network connection from a remote DICOM application

### Synopsis

```c
#include "mergecom.h"

MC_STATUS MC_Wait_For_Connection ( 
    int Timeout, 
    MC_SOCKET *Socket 
) 

MC_STATUS MC_Wait_For_Connection_On_Port ( 
    int Timeout, 
    int Port, 
    MC_SOCKET *Socket 
) 

MC_STATUS MC_Wait_For_Connection_On_Address ( 
    int Timeout, 
    int Port, 
    const char *Address, 
    MC_SOCKET *Socket 
) 
```

**Timeout**  The maximum 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”.

**Socket**  The socket for which the incoming connection has been accepted is returned here.

**Port**  Port number to listen on.
Address of the network interface to listen on. This can be the IP address or the network name of the interface. If this argument is NULL, the listener will be set up on all available network interfaces.

Remarks

MC_Wait_For_Connection, MC_Wait_For_Connection_On_Port and MC_Wait_For_Connection_On_Address are used to wait for a remote DICOM application to make a request for an association to this application. These routines work in conjunction with the MC_Process_Association_Request and MC_Process_Secure_Association_Request functions which actually process the incoming association request.

If Timeout is a positive number Merge DICOM Toolkit will wait no longer than Timeout seconds for an association request. If Timeout is negative, Merge DICOM Toolkit will not return until an association request has arrived or an error occurs. If Timeout is zero, Merge DICOM Toolkit 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.

These functions were introduced to address a problem with the MC_Wait_For_Association function where a poorly performing DICOM application connecting to Merge DICOM Toolkit could cause delays in processing DICOM associations. With MC_Wait_For_Association, when Merge DICOM Toolkit receives an association, it will not process other new incoming connections until it has read the initial association request from the network. The MC_Wait_For_Connection, MC_Wait_For_Connection_On_Port and MC_Wait_For_Connection_On_Address routines allow an application to create a thread or process to handle a new network connection as soon as it is received to eliminate this problem. The routines do not read any data from the network connection after it has been accepted.

After a MC_Wait_For_Connection returns, the socket returned must be passed to MC_Process_Association_Request or MC_Process_Secure_Association_Request for the association to be negotiated.

The port that MC_Wait_For_Connection 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_Connection 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 the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_TIMEOUT</td>
<td>The Timeout period expired.</td>
</tr>
<tr>
<td>MC_SYSTEM_CALL_INTERRUPTED</td>
<td>The system call used to wait for an association request was interrupted by a signal. Normally this situation is handled by retrying the MC_XXX call.</td>
</tr>
</tbody>
</table>
MC_UNKNOWN_HOST_CONNECTED A host whose remote hostname could be be resolved has attempted to connect and the connection has been dropped.

MC_NEGOTIATION_ABORTED An unexpected network error occurred when attempting to accept the connection.

MC_NULL_POINTER_PARM The Socket parameter 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 Merge DICOM Toolkit log file.

See Also

MC_Open_Association    MC_Open_Secure_Association
MC_Wait_For_Secure_Association    MC_Close_Listen_Port
MC_Close_Listen_Port_On_Address    MC_Wait_For_Association
MC_Wait_For_Association_On_Port    MC_Process_Association_Request
MC_Wait_For_Association_On_Address    MC_Wait_For_Secure_Association
MC_Process_Secure_Association_Request
MC_Release_Parent_Connection

MC_Wait_For_Secure_Association MC_Wait_For_Secure_Association_On_Port
MC_Wait_For_Secure_Association_On_Address

Waits for an association request from a remote DICOM application using a secure socket connection.

Synopsis

```c
#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  
)

MC_STATUS MC_Wait_For_Secure_Association_On_Address (  
    char *ServiceList,  
    int Timeout,  
    int ApplicationID,  
    int AddressId,  
    SecureSocketFunctions *SS_functions,  
    void *SS_context  
)
```

© Copyright Merge Healthcare Solutions Inc. 2023
```c
int Port,
const char *Address,
int *AssociationID,
SecureSocketFunctions *SS_functions,
void *SS_context
)

ServiceList  Name of a service list in the Merge DICOM Toolkit configuration file.

Timeout     The maximum 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.

Address      The network name or IP address of the network interface to listen on. If this argument is NULL the toolkit will accept connections on all available network interfaces.

SS_functions Pointer to a structure containing functions that will be called by Merge DICOM Toolkit while processing network I/O over the secure connection. (see below)

SS_context   An optional pointer to application-specific data that Merge DICOM Toolkit passes to the functions declared in SS_functions.

Remarks

Use MC_Wait_For_Secure_Association, MC_Wait_For_Secure_Association_On_Port and MC_Wait_For_Secure_Association_On_Address 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, while MC_Wait_For_Secure_Association_On_Port and MC_Wait_For_Secure_Association_On_Address will only negotiate associations for the specific Application passed to the call.

If Timeout is a positive number Merge DICOM Toolkit will wait no longer than Timeout seconds for an association request to arrive. If Timeout is negative, Merge DICOM Toolkit will not return until an association request has arrived or an error occurs. If Timeout is zero, Merge DICOM Toolkit 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 Merge DICOM Toolkit 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 or MC_Wait_For_Secure_Association_On_Address, Merge DICOM Toolkit rejects the association request and continues to wait for a valid association request. Optionally, “ACCEPT_ANY_APPLICATION_TITLE = YES” may be specified in the Merge DICOM Toolkit 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.

Merge DICOM Toolkit validates each received association request based on the protocol rules of DICOM and on configurable specifications contained in the Merge DICOM Toolkit 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 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, MC_Wait_For_Secure_Association_On_Port or MC_Wait_For_Secure_Association_On_Address calls, Merge DICOM Toolkit 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). Merge DICOM Toolkit 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:

```c
typedef struct MC_Secure_Socket_Functions_Struct
{
    SS_STATUS (NOEXP_FUNC *SS_Session_Start) (MC_SOCKET SocketToUse,
        CONN_TYPE ConnectionType,
        void *ApplicationContext,
```
void (**SecurityContext
);

SS_STATUS (NOEXP_FUNC *SS_Read) (  
  void *SScontext,
  void *ApplicationContext,
  char *Buffer,
  unsigned int BytesToRead,
  unsigned int *BytesRead,
  int Timeout
);

SS_STATUS (NOEXP_FUNC *SS_Write) (  
  void *SScontext,
  void *ApplicationContext,
  char *Buffer,
  unsigned int BytesToWrite,
  unsigned 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

Merge DICOM Toolkit 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. Merge DICOM Toolkit 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

Merge DICOM Toolkit 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. Merge DICOM Toolkit 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**

Merge DICOM Toolkit 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 Merge DICOM Toolkit 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**

Merge DICOM Toolkit 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 Merge DICOM Toolkit 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:** If *SS_ERROR* is returned, an outstanding MC_Send_Request_Message, MC_Send_Response_Message or MC_Send_Response call will return *MC_SYSTEM_ERROR*.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
</table>

© Copyright Merge Healthcare Solutions Inc. 2023
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_TIMEOUT</td>
<td>The Timeout period expired.</td>
</tr>
<tr>
<td>MC_NEGOTIATION_ABORTED</td>
<td>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.</td>
</tr>
<tr>
<td>MC_ASSOCIATION_REJECTED</td>
<td>An incoming association was aborted because no services were acceptable.</td>
</tr>
<tr>
<td>MC_SYSTEM_CALL_INTERRUPTED</td>
<td>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.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>AssociationID, ApplicationID, SS_functions or ServiceList parameter was NULL.</td>
</tr>
<tr>
<td>MC_NULL_VALUE</td>
<td>One or more of the function parameters within SS_functions was NULL.</td>
</tr>
<tr>
<td>MC_NO_APPLICATIONS_REGISTERED</td>
<td>No applications have been registered using MC_Register_Application.</td>
</tr>
<tr>
<td>MC_INVALID_SERVICE_LIST_NAME</td>
<td>ServiceList points at a null string.</td>
</tr>
<tr>
<td>MC_CONFIG_INFO_MISSING</td>
<td>Could not access ServiceList configuration parameters.</td>
</tr>
<tr>
<td>MC_CONFIG_INFO_ERROR</td>
<td>The ServiceList contained too many services.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**See Also**

- MC_Open_Association
- MC_Open_Secure_Association
- MC_Accept_Association
- MC_Reject_Association
- MC_Wait_For_Association
- MC_Close_Listen_Port
- MC_Close_Listen_Port_On_Address

**MC_Write_File**

**MC_Write_File_By_Callback**

Writes a file object to media.

**Synopsis**

```c
#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:

```c
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 Merge DICOM Toolkit is providing the first block of file data.

*CbisLast* Is TRUE (not zero) when Merge DICOM Toolkit 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. Merge DICOM Toolkit 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 configuration value in the mergecom.pro file.

YourToMediaFunction

YourToMediaFunction will be called repeatedly to pass blocks of data to it. Merge DICOM Toolkit 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 the enumerated MC_STATUS codes defined in “mcstatus.h”:
<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_APPLICATION_ID</td>
<td>The ApplicationID value is not a valid application object ID.</td>
</tr>
<tr>
<td>MC_INVALID_FILE_ID</td>
<td>The FileID value is not a valid file object ID.</td>
</tr>
<tr>
<td>MC_INVALID_PAD</td>
<td>The NumBytes parameter contained too large of a number or an odd number.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the</td>
</tr>
<tr>
<td></td>
<td>operating environment. A message describing the error has been written</td>
</tr>
<tr>
<td></td>
<td>to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_MESSAGE_EMPTY</td>
<td>The file has no attributes in it.</td>
</tr>
<tr>
<td>MC_CALLBACK_CANNOT_COMPLY</td>
<td>YourToMediaFunction returned a value other than MC_NORMAL_COMPLETION.</td>
</tr>
<tr>
<td>MC_INVALID_TRANSFER_SYNTAX</td>
<td>The transfer syntax is improperly specified or is not specified in the</td>
</tr>
<tr>
<td></td>
<td>DICOM File Meta Information attributes.</td>
</tr>
<tr>
<td>MC_TEMPFILE_ERROR</td>
<td>A value for an attribute is stored temporarily on file and an I/O</td>
</tr>
<tr>
<td></td>
<td>error occurred.</td>
</tr>
</tbody>
</table>

See Also

- MC_Open_File
- MC_Create_File
- MC_Create_Empty_File

**MC_XML_To_Message**

Reads attribute values from a Merge DICOM Model XML string into a message, file or item object.

**Synopsis**

```c
#include "mc3msg.h"

MC_STATUS MC_XML_To_Message (  
    int MessageID,  
    void *UserInfo,  
    MC_STATUS (*YourGetXMLFunction)()  
)
```

- **MessageID**
  - The identifier of a message, file or item object.
- **UserInfo**
  - Address of data which will be passed on to YourGetXMLFunction each time it is called. This may be NULL.
- **YourGetXMLFunction**
  - Name of a function which will be called repeatedly to get the XML string.

The function must be prototyped as follows:

```c
MC_STATUS YourGetXMLFunction (  
    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_XML_To_Message function. This may be NULL.

CBdataSize  Set *CBdataSize to the number of bytes you are providing.

CBdataBuffer  Set *CBdataBuffer to the address of the data you are
providing.

CBisFirst  Set to TRUE (not zero) by the toolkit on the first call.

CBisLast  Set *CBisLast to TRUE (not zero) when you are returning
with the last block of XML data.

Remarks

MC_XML_To_Message requests that the XML buffer be converted into a DICOM message. The XML
data is requested from the user in blocks by calling YourGetXMLFunction repeatedly until the entire
XML content has been received by the Merge DICOM Toolkit.

YourGetXMLFunction

YourGetXMLFunction will be called repeatedly to get blocks of data from it. Merge DICOM Toolkit
sets CBisFirst to TRUE(non-zero) the first time it calls YourGetXMLFunction for this message and
YourGetXMLFunction should set *CBisLast to TRUE(non-zero) when it gives Merge DICOM Toolkit
the final block of data to be converted.

**CBdataBuffer is set to the address of a buffer containing the data block to be converted, and
CBdataSize is set to the number of bytes to be placed at **CBdataBuffer.

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>YourGetXMLFunction parameter was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>The MessageID value is not a valid message, file or item object ID.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the</td>
</tr>
<tr>
<td></td>
<td>operating environment. A message describing the error has been written</td>
</tr>
<tr>
<td></td>
<td>to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_CALLBACK_CANNOT_COMPLY</td>
<td>YourGetXMLFunction returned a value other than MC_NORMAL_COMPLETION.</td>
</tr>
<tr>
<td>MC_LIBRARY_NOT_INITIALIZED</td>
<td>This call was made prior to the initialization of the library.</td>
</tr>
</tbody>
</table>
See Also
MC_Message_To_XML

MC_XML_Native_To_Message

Reads attribute values from a Native DICOM Model XML string into a message, file or item object.

Synopsis
#include "mc3msg.h"

MC_STATUS MC_XML_Native_To_Message
(int MessageID,
void *UserInfo,
MC_STATUS (*YourGetXMLFunction)())

MessageID The identifier of a message, file or item object.
UserInfo Address of data which will be passed on to
YourGetXMLFunction each time it is called.
This may be NULL.
YourGetXMLFunction Name of a function which will be called
repeatedly to get the XML string.

The function must be prototyped as follows:
MC_STATUS YourGetXMLFunction
(int CBmessageID,
void *CBuserInfo,
int *CBdataSize,
void **CBdataBuffer,
in CBisFirst,
in *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_XML_Native_To_Message function. This may be NULL.
CBdataSize Set *CBdataSize to the number of bytes you are providing.
CBdataBuffer Set *CBdataBuffer to the address of the data you are
providing.
CBisFirst Set to TRUE (not zero) by the toolkit on the first call.
CBisLast Set *CBisLast to TRUE (not zero) when you are returning
with the last block of XML data.

Remarks
MC_XML_Native_To_Message requests that the Native DICOM Model XML buffer be converted into
a DICOM message. The XML data is requested from the user in blocks by calling
YourGetXMLFunction repeatedly until the entire XML content has been received by the Merge DICOM Toolkit.

**YourGetXMLFunction**

YourGetXMLFunction will be called repeatedly to get blocks of data from it. Merge DICOM Toolkit sets CBisFirst to TRUE(non-zero) the first time it calls YourGetXMLFunction for this message and YourGetXMLFunction should set *CBisLast to TRUE(non-zero) when it gives Merge DICOM Toolkit the final block of data to be converted.

**CBdataBuffer** is set to the address of a buffer containing the data block to be converted, and CBdataSize is set to the number of bytes to be placed at **CBdataBuffer.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>YourGetXMLFunction parameter was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>The MessageID value is not a valid message, file or item object ID.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_CALLBACK_CANNOT_COMPLY</td>
<td>YourGetXMLFunction returned a value other than MC_NORMAL_COMPLETION.</td>
</tr>
<tr>
<td>MC_LIBRARY_NOT_INITIALIZED</td>
<td>This call was made prior to the initialization of the library.</td>
</tr>
</tbody>
</table>

**See Also**

MC_Message_To_XML_Native

**High Level API Reference**

This section of the Reference Manual includes the function reference for the domain-specific functionality of the Toolkit. Each reference page provides:

- 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
- a "See also" cross reference to other functions

**MC_DDH_Create**

Creates a new DICOMDIR file and its associated toolkit object.
# Synopsis

```c
#include "mc3media.h"

MC_STATUS MC_DDH_Create (  
    const char *FilePath,  
    const char *FileSetID,  
    int TemplateFileID,  
    int *DirMsgIDPtr  
)
```

- **FilePath**: Path to the DICOMDIR file (including the file name).
- **FileSetID**: The value to be assigned to the File Set ID attribute (0004,1130). If this argument is NULL the toolkit will copy the value of the File Set ID attribute from the template file if specified.
- **TemplateFileID**: The ID of a file object containing attribute values for the new DICOMDIR. If this argument is 0 the toolkit creates an empty DICOMDIR with a minimal number of attributes.
- **DirMsgIDPtr**: Pointer to a variable receiving the toolkit identifier of the new directory object.

# Remarks

This function writes a new **DICOMDIR** file with no records and returns the directory object identifier representing the new directory file. If a file with the same name exists, its content is overwritten.

The directory object must be freed via **MC_Free_File** when it is no longer needed.

The toolkit copies group 2 and 4 attribute values from the specified template file. If the Media Storage SOP Instance UID attribute is not present in the template file or the template file ID is 0 the toolkit will generate a UID based on the **IMPLEMENTATION_CLASS_UID** configuration item.

See **MC_DDH_Open** function for details on managing the directory object.

# Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td><strong>FilePath</strong> or <strong>DirMsgIDPtr</strong> was NULL.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>The <strong>TemplateFileID</strong> is not a valid toolkit file object identifier.</td>
</tr>
<tr>
<td>MC_INVALID_CHARS_IN_VALUE</td>
<td>The <strong>FileSetID</strong> argument contains invalid characters.</td>
</tr>
<tr>
<td>MC_INVALID_VALUE_FOR_VR</td>
<td>The <strong>FileSetID</strong> argument does not conform to the requirements of the CS value representation.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>
**MC_CALLBACK_CANNOT_COMPLY**  
A file read or write operation failed.

**MC_LIBRARY_NOT_INITIALIZED**  
The library has not been properly initialized.

**MC_TEMPFILE_ERROR**  
A value for an attribute is stored temporarily on file and an I/O error occurred.

### See Also

- **MC_DDH_Open**
- **MC_Free_File**

---

### MC_DDH_Open

**Synopsis**

```c
#include "mc3media.h"
MC_STATUS MC_DDH_Create  
    (  
        const char  *FilePath,  
        int  *DirMsgIDPtr  
    )
```

- **FilePath**  
  Path to the DICOMDIR file (including the file name).

- **DirMsgIDPtr**  
  Pointer to a variable receiving the toolkit identifier of the new directory object.

### Remarks

This function opens an existing DICOMDIR file for reading and/or updating and returns the directory object identifier representing its content. If the specified file does not exist or is not accessible the toolkit returns an error.

As opposed to opening DICOMDIR files using **MC_Open_File**, this function does not read the whole content of the DICOMDIR in the memory but instead reads only the first directory record if it exists. Subsequent directory records are read only on demand, when the application requires access to them, providing fast and efficient access to large DICOMDIR's.

The directory object created by **MC_DDH_Open** allows the application to incrementally add new records or delete existing records using **MC_DDH_Add_Record**, **MC_DDH_Delete_Record** and **MC_DDH_Update** function, without rewriting the whole file.

The returned directory object identifier can be used with all **MC_DDH...** functions that require a directory ID or with **MC_Get_Value...** functions to obtain group 2 and 4 attribute values, but it can not be used with any **MC_Dir...** function.

The Directory Record Sequence (0004,1220) attribute and its content is not accessible to the application as it is managed internally by the toolkit.

Although the toolkit allows the application to modify the content of existing directory records or the attributes in the directory object itself, the changes will not be reflected in the DICOMDIR file.

The directory file object must be freed via **MC_Free_File** when it is no longer needed.
**Return Value**

One of the enumerated `MC_STATUS` codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MC_NORMAL_COMPLETION</code></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><code>MC_UNEXPECTED_EOD</code></td>
<td>The specified file does not contain all data necessary to decode an attribute's value.</td>
</tr>
<tr>
<td><code>MC_INVALID_FILE</code></td>
<td>The specified file does not conform to the requirements of a DICOMDIR file. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td><code>MC_INVALID_TRANSFER_SYNTAX</code></td>
<td>An invalid transfer syntax code was found within the file's DICOM File Meta Information.</td>
</tr>
<tr>
<td><code>MC_OUT_OF_ORDER_TAG</code></td>
<td>A tag was found in the file that was not in ascending order. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td><code>MC_INVALID_TAG</code></td>
<td>The file contains an invalid tag.</td>
</tr>
<tr>
<td><code>MC_VALUE_TOO_LARGE</code></td>
<td>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.</td>
</tr>
<tr>
<td><code>MC_CALLBACK_CANNOT_COMPLY</code></td>
<td>A file read operation failed.</td>
</tr>
<tr>
<td><code>MC_INVALID_LENGTH_FOR_VR</code></td>
<td>The value length for one of the file attributes was not legal for its value representation.</td>
</tr>
<tr>
<td><code>MC_INVALID_CHARS_IN_VALUE</code></td>
<td>A value for an attribute does not contain valid characters for its value representation.</td>
</tr>
<tr>
<td><code>MC_INVALID_VALUE_FOR_VR</code></td>
<td>The <code>FileSetID</code> argument does not conform to the requirements of the CS value representation.</td>
</tr>
<tr>
<td><code>MC_NULL_POINTER_PARM</code></td>
<td><code>FilePath</code> or <code>DirMsgIDPtr</code> was NULL.</td>
</tr>
<tr>
<td><code>MC_TEMPFILE_ERROR</code></td>
<td>A value for an attribute is stored temporarily on file and an I/O error occurred.</td>
</tr>
<tr>
<td><code>MC_UNABLE_TO_GET_ITEM_ID</code></td>
<td>The toolkit failed to read the first directory record.</td>
</tr>
<tr>
<td><code>MC_SYSTEM_ERROR</code></td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td><code>MC_LIBRARY_NOT_INITIALIZED</code></td>
<td>The library has not been properly initialized.</td>
</tr>
</tbody>
</table>

Any of the status codes returned by `MC_Set_Value` may also be returned.

**See Also**

`MC_Free_File`

**MC_DDH_Update**

Commits all pending changes to the DICOMDIR file.
Synopsis

#include "mc3media.h"

MC_STATUS MC_DDH_Update ( 
    int DirMsgID
)

DirMsgID The identifier of a directory object obtained using MC_DDH_Create or MC_DDH_Open functions.

Remarks

This function updates the DICOMDIR file to reflect all changes to the dicomdir object requested through MC_DDH_Add_Record and MC_DDH_Delete_Record functions. The toolkit writes new records to the end of the directory record sequence and bridges over deleted records so that they are no longer accessible.

MC_DDH_Update does not make changes to existing records or the attributes of the directory object, except for the values of the offset attributes required to represent the modified record hierarchy.

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_DICOMDIR_ID</td>
<td>DirMsgID is not a valid toolkit directory object identifier.</td>
</tr>
<tr>
<td>MC_INVALID_FILE</td>
<td>A failure occurred while writing the file. The DICOMDIR file may be</td>
</tr>
<tr>
<td></td>
<td>corrupted.</td>
</tr>
<tr>
<td>MC_CALLBACK_CANNOT_COMPLY</td>
<td>The DICOMDIR file could not be opened for writing or a file read or</td>
</tr>
<tr>
<td></td>
<td>write operation failed. The DICOMDIR file is in a consistent state</td>
</tr>
<tr>
<td></td>
<td>but the directory object can not be reliably used and it should be</td>
</tr>
<tr>
<td></td>
<td>freed using MC_Free_File.</td>
</tr>
<tr>
<td>MC_INVALID_DIRECTORY_RECORD_OFFSET</td>
<td>A directory record offset value is not correct and new records cannot</td>
</tr>
<tr>
<td></td>
<td>be written, the directory object can still be used in read-only mode.</td>
</tr>
<tr>
<td>MC_TEMPFILE_ERROR</td>
<td>A value for an attribute is stored temporarily on file and an I/O</td>
</tr>
<tr>
<td></td>
<td>error occurred. The DICOMDIR file may be in an inconsistent state.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the</td>
</tr>
<tr>
<td></td>
<td>operating environment. A message describing the error has been written</td>
</tr>
<tr>
<td></td>
<td>to the Merge DICOM Toolkit log file. The DICOMDIR file is in a</td>
</tr>
<tr>
<td></td>
<td>consistent state but the directory object can not be reliably used</td>
</tr>
<tr>
<td></td>
<td>and it should be freed using MC_Free_File.</td>
</tr>
<tr>
<td>MC_LIBRARY_NOT_INITIALIZED</td>
<td>The library has not been properly initialized.</td>
</tr>
</tbody>
</table>
See Also

MC_DDH_Create  MC_DDH_Open
MC_DDH_Add_Record  MC_DDH_Delete_Record

MC_DDH_Traverse_Records
Provides easy record traversal functionality.

Synopsis

```c
#include "mc3media.h"

MC_STATUS MC_DDH_Traverse_Records ( 
    int RootID, 
    void *UserData, 
    MC_TRAVERSAL_STATUS (*YourTraverseCallback) () 
)
```

- **RootID**: The object identifier of the record or directory representing the root for the traversal.
- **UserData**: The pointer to be passed to the user's callback. The value of this argument is not interpreted by the toolkit and it can be NULL.
- **YourTraverseCallback**: The function to be called for each record being visited.

The function must be prototyped as follows:

```c
MC_TRAVERSAL_STATUS YourTraverseCallback ( 
    int CurrentRecID, 
    void *UserData 
)
```

- **CurrentRecID**: The identifier of the directory record being traversed.
- **UserData**: The value passed in the `UserData` argument of the `MC_DDH_Traverse_Records` function.

Remarks

This function traverses all directory records that are below the specified directory record, or all records if the RootID is the identifier of a directory object. The application can control the traversal through the various status codes returned by the callback for each call:

- **MC_TS_CONTINUE**: The traversal continues normally with the lower level records of the current record or with the next record if the current record does not have lower level records.
- **MC_TS_STOP_LEVEL**: The traversal continues with the next record of the current record's parent, skipping all lower level records and next records of the current record.
- **MC_TS_STOP_LOWER**: The traversal continues with the next record of the current record, skipping the current record's lower level records.
- MC_TS_STOP - Stop the traversal. **MC_DDH_Traverse_Records** will return MC_NORMAL_COMPLETION.
- MC_TS_ERROR - Stop the traversal because an error occurred in the callback. **MC_DDH_Traverse_Records** will return MC_CALLBACK_CANNOT_COMPLY.

**MC_DDH_Traverse_Records** provides an easy and fast way of searching and counting records.

### Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_DICOMDIR_ID</td>
<td>The RootID argument does not represent a valid record or directory object identifier</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>The YourTraverseCallback argument was NULL.</td>
</tr>
<tr>
<td>MC_CALLBACK_CANNOT_COMPLY</td>
<td>The callback function returned MC_TS_ERROR or an invalid status or a file read operation failed.</td>
</tr>
<tr>
<td>MC_INVALID_DIRECTORY_RECORD_OFFSET</td>
<td>A directory record offset value is not correct.</td>
</tr>
<tr>
<td>MC_UNEXPECTED_EOD</td>
<td>Not enough data to decode an attribute's value.</td>
</tr>
<tr>
<td>MC_OUT_OF_ORDER_TAG</td>
<td>A tag was found in the file that was not in ascending order. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_INVALID_TAG</td>
<td>A record item contains an invalid tag.</td>
</tr>
<tr>
<td>MC_VALUE_TOO_LARGE</td>
<td>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.</td>
</tr>
<tr>
<td>MC_INVALID_LENGTH_FOR_VR</td>
<td>The value length for one of the file attributes was not legal for its value representation.</td>
</tr>
<tr>
<td>MC_INVALID_VALUE_FOR_VR</td>
<td>The value for one of the file attributes was not legal for its value representation.</td>
</tr>
<tr>
<td>MC_INVALID_CHARS_IN_VALUE</td>
<td>A value for an attribute does not contain valid characters for its value representation.</td>
</tr>
<tr>
<td>MC_TEMPFILE_ERROR</td>
<td>A value for an attribute is stored temporarily on file and an I/O error occurred.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_LIBRARY_NOT_INITIALIZED</td>
<td>The library has not been properly initialized.</td>
</tr>
</tbody>
</table>

**MC_DDH_Get_Record_Type**

Gets a directory record's type.
Synopsis
#include "mc3media.h"

MC_STATUS MC_DDH_Get_Record_Type (  
    int RecordID,  
    MC_DIR_RECORD_TYPE *RecType  
)

  **RecordID**
  The toolkit identifier of the record.

  **RecType**
  Pointer to the variable receiving the record type. The 
  MC_DIR_RECORD_TYPE enumeration is defined in mc3media.h.

Remarks
This function can be used to obtain an integer representing the type of the specified directory record 
as specified by the value of the Directory Record Type attribute (0004,1430).

If the value of the Directory Record Type attribute does not correspond to any of the values in the 
MC_DIR_RECORD_TYPE enum the toolkit returns MC_REC_TYPE_UNKNOWN as the record type.

Return Value
One of the enumerated MC_STATUS codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_RECORD_ID</td>
<td>The RecordID argument does not represent a valid record</td>
</tr>
<tr>
<td></td>
<td>object identifier.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>The RecType argument was NULL.</td>
</tr>
<tr>
<td>MC_LIBRARY_NOT_INITIALIZED</td>
<td>The library has not been properly initialized.</td>
</tr>
</tbody>
</table>

MC_DDH_Get_Parent_Record
Gets the parent directory record of the specified record.

Synopsis
#include "mc3media.h"

MC_STATUS MC_DDH_Get_Parent_Record (  
    int RecordID,  
    int *ParentID  
)

  **RecordID**
  The identifier of the record object for which the parent 
  record is requested.

  **ParentID**
  Pointer to the variable receiving the parent record's 
  identifier.
Remarks

MC_DDH_Get_Parent_Record provides the parent directory record of a record object. If the specified record is a top level record this function sets the value pointed by the ParentID argument to 0 and returns MC_NORMAL_COMPLETION.

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_RECORD_ID</td>
<td>The RecordID argument does not represent a valid record object identifier.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>The ParentID argument was NULL.</td>
</tr>
<tr>
<td>MC_CALLBACK_CANNOT_COMPLY</td>
<td>A file read operation failed.</td>
</tr>
<tr>
<td>MC_INVALID_DIRECTORY_RECORD_OFFSET</td>
<td>A directory record offset value is not correct.</td>
</tr>
<tr>
<td>MC_UNEXPECTED_EOD</td>
<td>Not enough data to decode an attribute's value.</td>
</tr>
<tr>
<td>MC_OUT_OF_ORDER_TAG</td>
<td>A tag was found in the file that was not in ascending order. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_INVALID_TAG</td>
<td>A record item contains an invalid tag.</td>
</tr>
<tr>
<td>MC_VALUE_TOO_LARGE</td>
<td>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.</td>
</tr>
<tr>
<td>MC_INVALID_LENGTH_FOR_VR</td>
<td>The value length for one of the file attributes was not legal for its value representation.</td>
</tr>
<tr>
<td>MC_INVALID_VALUE_FOR_VR</td>
<td>The value for one of the file attributes was not legal for its value representation.</td>
</tr>
<tr>
<td>MC_INVALID_CHARS_IN_VALUE</td>
<td>A value for an attribute does not contain valid characters for its value representation.</td>
</tr>
<tr>
<td>MC_TEMPFILE_ERROR</td>
<td>A value for an attribute is stored temporarily on file and an I/O error occurred.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_LIBRARY_NOT_INITIALIZED</td>
<td>The library has not been properly initialized.</td>
</tr>
</tbody>
</table>

MC_DDH_Get_Next_Record

Gets the next directory record in a directory entity.
Synopsis

```c
#include "mc3media.h"

MC_STATUS MC_DDH_Get_Next_Record (  
    int RecordID,  
    int *NextID  
)
```

*RecordID* The identifier of the record object of which the next record is requested.

*ParentID* Pointer to the variable receiving the next record’s identifier.

Remarks

`MC_DDH_Get_Next_Record` retrieves the directory record following the specified record in the directory record hierarchy. If the specified record is the last record in the list of lower level record of its parent record this function sets the value of the *NextID* to 0 and returns `MC_NORMAL_COMPLETION`.

The returned record identifier can be used with the `MC_Get_Value...` functions to access the record’s attribute values.

Return Value

One of the enumerated `MC_STATUS` codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MC_NORMAL_COMPLETION</code></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><code>MC_INVALID_RECORD_ID</code></td>
<td>The <em>RecordID</em> argument does not represent a valid record object identifier.</td>
</tr>
<tr>
<td><code>MC_NULL_POINTER_PARM</code></td>
<td>The <em>NextID</em> argument was NULL.</td>
</tr>
<tr>
<td><code>MC_CALLBACK_CANNOT_COMPLY</code></td>
<td>A file read operation failed.</td>
</tr>
<tr>
<td><code>MC_INVALID_DIRECTORY_RECORD_OFFSET</code></td>
<td>A directory record offset value is not correct.</td>
</tr>
<tr>
<td><code>MC_UNEXPECTED_EOD</code></td>
<td>Not enough data to decode an attribute's value.</td>
</tr>
<tr>
<td><code>MC_OUT_OF_ORDER_TAG</code></td>
<td>A tag was found in the file that was not in ascending order. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td><code>MC_INVALID_TAG</code></td>
<td>A record item contains an invalid tag.</td>
</tr>
<tr>
<td><code>MC_VALUE_TOO_LARGE</code></td>
<td>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.</td>
</tr>
<tr>
<td><code>MC_INVALID_LENGTH_FOR_VR</code></td>
<td>The value length for one of the file attributes was not legal for its value representation.</td>
</tr>
<tr>
<td><code>MC_INVALID_VALUE_FOR_VR</code></td>
<td>The value for one of the file attributes was not legal for its value representation.</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>MC_INVALID_CHARS_IN_VALUE</td>
<td>A value for an attribute does not contain valid characters for its value representation.</td>
</tr>
<tr>
<td>MC_TEMPFILE_ERROR</td>
<td>A value for an attribute is stored temporarily on file and an I/O error occurred.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_LIBRARY_NOT_INITIALIZED</td>
<td>The library has not been properly initialized.</td>
</tr>
</tbody>
</table>

### MC_DDH_Get_First_Lower_Record

Gets the first lower level directory record of the specified record.

**Synopsis**

```c
#include "mc3media.h"

MC_STATUS MC_DDH_Get_First_Lower_Record (  
    int ParentID,  
    int *LowerID  
)
```

- **ParentID**
  - The identifier of the record object for which the lower level record is requested or the identifier of the directory object if the first record in the root entity is requested.
- **LowerID**
  - Pointer to the variable receiving the first lower level record's identifier.

**Remarks**

`MC_DDH_Get_First_Lower_Record` retrieves the first lower level directory record of the specified record. If the specified record does not have lower level records this function sets the value of the `LowerID` to 0 and returns `MC_NORMAL_COMPLETION`.

The returned record identifier can be used with the `MC_Get_Value...` functions to access the record's attribute values.

**Return Value**

One of the enumerated `MC_STATUS` codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_RECORD_ID</td>
<td>The <code>ParentID</code> argument does not represent a valid record or DICOMDIR object identifier.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>The <code>LowerID</code> argument was NULL.</td>
</tr>
<tr>
<td>MC_CALLBACK_CANNOT_COMPLY</td>
<td>A file read operation failed.</td>
</tr>
</tbody>
</table>
**MC_INVALID_DIRECTORY_RECORD_OFFSET**
A directory record offset value is not correct.

**MC_UNEXPECTED_EOD**
Not enough data to decode an attribute's value.

**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 Merge DICOM Toolkit log file.

**MC_INVALID_TAG**
A record item 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_INVALID_LENGTH_FOR_VR**
The value length for one of the file attributes was not legal for its value representation.

**MC_INVALID_VALUE_FOR_VR**
The value for one of the file attributes was not legal for its value representation.

**MC_INVALID_CHARS_IN_VALUE**
A value for an attribute does not contain valid characters for its value representation.

**MC_TEMPFILE_ERROR**
A value for an attribute is stored temporarily on file and an I/O error occurred.

**MC_SYSTEM_ERROR**
An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.

**MC_LIBRARY_NOT_INITIALIZED**
The library has not been properly initialized.

---

**MC_DDH_Copy_Values**
Copies attribute values from one message or item to another message or item.

**Synopsis**

```c
#include "mc3media.h"

MC_STATUS MC_DDH_Copy_Values (  
    int SourceID,  
    int DestID,  
    unsigned long *TagList  
)
```

**SourceID**
The ID of the source message, file, item or record object containing the values to be copied.

**DestID**
The ID of the destination message, file, item or record object which will receive the values.
TagList A zero ended array of tag numbers of the attributes to be copied. If a tag is a group length tag, the attributes in the whole group are copied. If this parameter is NULL, all attributes are copied.

Remarks
This function can be used to fill in directory record values from messages when adding instances to a file set.

Return Value
One of the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_MESSAGE_ID</td>
<td>The source or destination ID value is not a valid toolkit object identifier.</td>
</tr>
<tr>
<td>MC_TEMPFILE_ERROR</td>
<td>A value for an attribute is stored temporarily on file and an I/O error occurred.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_LIBRARY_NOT_INITIALIZED</td>
<td>The library has not been properly initialized.</td>
</tr>
</tbody>
</table>

MC_DDH_Add_Record

Adds a new directory record to a directory object.

Synopsis

```c
#include "mc3media.h"

MC_STATUS MC/DDH_Add_Record (  
  int ParentID,  
  const char *RecordType  
  int *RecordID
)
```

**ParentID** The identifier of a directory object obtained using MC/DDH_Create or MC/DDH_Open functions or the identifier of a record object obtained using one of the MC/DDH_... record navigation functions.
**RecordType**
The directory record type name for the new record or the SOP Class UID of the referenced instance. If this argument is NULL the toolkit will set the record type based on the type of the parent record: "PATIENT" for root level records, "STUDY" if the parent is "PATIENT", "SERIES" if the parent is "STUDY" and "IMAGE" if the parent is "SERIES". The application can change the value of the Directory Record Type attribute of the new record item after this function returns, before calling `MC_DDH_Update`.

**RecordID**
Pointer to the variable receiving the identifier of the new record.

**Remarks**

`MC_DDH_Add_Record` creates a new directory record object and appends it to the end of the child record list of the specified parent record or to the end of the root entity if the specified parent is the directory object itself.

If the type of the record is not specified the toolkit will try to determine it based on the type of the parent record. For instance level records, the application can specify the SOP Class UID of the referenced object in the `RecordType` argument and the toolkit will create the corresponding record type.

The returned record identifier can be used with the `MC_Set_Value` or `MC_DDH_Copy_Values` functions to set the attribute values in the new record.

The new record and any values set for its attributes are written to the DICOMDIR file when the application calls `MC_DDH_Update`. If the directory object is freed without calling `MC_DDH_Update`, all changes to the record hierarchy will be lost.

**Return Value**

One of the enumerated `MC_STATUS` codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MC_NORMAL_COMPLETION</code></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><code>MC_INVALID_RECORD_ID</code></td>
<td>The <code>ParentID</code> argument does not represent a valid record or DICOMDIR object identifier.</td>
</tr>
<tr>
<td><code>MC_NULL_POINTER_PARM</code></td>
<td>The <code>RecordID</code> argument was NULL.</td>
</tr>
<tr>
<td><code>MC_INVALID_LOWER_DIR_RECORD</code></td>
<td>The record type specified in the <code>RecordType</code> argument is not a valid lower level record of the parent record.</td>
</tr>
<tr>
<td><code>MC_BAD_DIR_RECORD_TYPE</code></td>
<td>The type of the parent record is unknown.</td>
</tr>
<tr>
<td><code>MC_CALLBACK_CANNOT_COMPLY</code></td>
<td>A file read operation failed.</td>
</tr>
<tr>
<td><code>MC_INVALID_DIRECTORY_RECORD_OFFSET</code></td>
<td>A directory record offset value is not correct.</td>
</tr>
<tr>
<td><code>MC_UNEXPECTED_EOD</code></td>
<td>Not enough data to decode an attribute's value.</td>
</tr>
</tbody>
</table>
**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 Merge DICOM Toolkit log file.

**MC_INVALID_TAG**
A record item 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_INVALID_LENGTH_FOR_VR**
The value length for one of the file attributes was not legal for its value representation.

**MC_INVALID_VALUE_FOR_VR**
The value for one of the file attributes was not legal for its value representation.

**MC_INVALID_CHARS_IN_VALUE**
A value for an attribute does not contain valid characters for its value representation.

**MC_TEMPFILE_ERROR**
A value for an attribute is stored temporarily on file and an I/O error occurred.

**MC_SYSTEM_ERROR**
An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.

**MC_LIBRARY_NOT_INITIALIZED**
The library has not been properly initialized.

**See Also**
MC_DDH_Create
MC_DDH_Open
MC_DDH_Update

---

**MC_DDH_Delete_Record**
Deletes a directory record.

**Synopsis**
```
#include "mc3media.h"

MC_STATUS MC_DDH_Delete_Record (  
    int RecordID  
)
```

*RecordID* The toolkit identifier of the directory record to delete.

**Remarks**
**MC_DDH_Delete_Record** removes the specified directory record and any lower level record from the record hierarchy.

The resources allocated for the directory record objects are automatically freed.

The changes made by this function are not reflected in the DICOMDIR file until the application calls **MC_DDH_Update**.
**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_RECORD_ID</td>
<td>The RecordID argument does not represent a valid record object identifier.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
<tr>
<td>MC_LIBRARY_NOT_INITIALIZED</td>
<td>The library has not been properly initialized.</td>
</tr>
</tbody>
</table>

**See Also**

MC_DDH_Update

---

**MC_DDH_Release_Record**

Releases the content of the record.

**Synopsis**

```c
#include “mc3media.h”

MC_STATUS MC_DDH_Release_Record (  
    int RecordID
  )
```

- **RecordID**  
  The toolkit identifier of the directory record to release.

**Remarks**

This function releases the specified record and its lower level records to reduce the toolkit’s memory usage. Although the record identifier becomes invalid after this call, the record itself is not deleted and can be accessed again using one of the record navigation functions, **MC_DDH_Get_Next_Record** or **MC_DDH_Get_First_Lower_Level_Record**, in which case the toolkit will read the content of the record from the DICOMDIR file again and assign a new record identifier.

This function does not modify the directory record hierarchy.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_RECORD_ID</td>
<td>The RecordID argument does not represent a valid record object identifier.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>
**MC_SRH_Create_SR**

Creates a new SR root node.

**Synopsis**

```c
MC_STATUS MC_SRH_Create_SR (
    const char *AserviceName,
    const char *AtemplateId,
    SR_CONT_CONTINUITY Acontinuity,
    const char *AconceptNameValue,
    const char *AconceptNameScheme,
    const char *AconceptNameMeaning,
    int *AsrRootID
)
```

- `AserviceName`: The service (Sop Class) the SR uses.
- `AtemplateId`: DICOM Template Identifier that describes the content of this Content Item and its subsidiary Content Items. Can be NULL.
- `Acontinuity`: Enumerated value specifies whether or not its contained Content Items are logically linked in a continuous textual flow, or are separate items.
- `AconceptNameValue`: Code Value describing the concept represented by the root Content Item.
- `AconceptNameScheme`: Coding Scheme Designator
- `AconceptNameMeaning`: Code Meaning
- `AsrRootID`: Pointer that receives a numeric identifier for an SR node.

**Remarks**

`MC_SRH_Create_SR` creates SR tree management object. This returned item identifier can then be used to set and retrieve SR values.

**Return Value**

One of the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
</tbody>
</table>
### MC_NULL_POINTER_PARM
One of the parameters 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 Merge DICOM Toolkit log file.

### MC_SRH_Free_SR
Releases all memory allocated by SR.

#### Synopsis

```
MC_STATUS MC_SRH_Free_SR (int AsrRootID)
```

- **AsrRootID**: The identifier of the SR root node to be released.

#### Remarks
Underlying message object will be released as well.

#### Return Value
One of the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrRootID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

### MC_SRH_Add_Child
Adds a child node to the provided parent SR node.

#### Synopsis

```
MC_STATUS MC_SRH_Add_Child (int AsrNodeID,
                           SR_RELATIONSHIP Arelationship,
                           int *AsrChildNodeID)
```

- **AsrNodeID**: The identifier assigned to the given SR object to which the child is to be added.
- **Arelationship**: Enumerated value representing a relationship.
AsrChildNodeID

Pointer to the identifier assigned to the child item object which is to be added to the parent. On successful completion returns a new child SR node identifier.

Remarks

This function shall be used for adding existing items to the SR tree. It is recommended to use specific functions like MC_SRH_Add_XXX_Child to create child nodes from scratch.

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_VALUE_NOT_ALLOWED</td>
<td>The Arelationship value is not valid.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

MC_SRH_Add_TEXT_Child

Creates a new child TEXT node under the provided parent.

Synopsis

MC_STATUS MC_SRH_Add_TEXT_Child (  
    int AsrNodeID,  
    SR_RELATIONSHIP Arelationship,  
    const char *AconceptNameValue,  
    const char *AconceptNameScheme,  
    const char *AconceptNameMeaning,  
    const char *Atext,  
    int *AsrChildNodeID  
)  

AsrNodeID

An Identifier of the parent SR item to which the child is to be added.

Arelationship

Enumerated value representing a relationship.

AconceptNameValue

Code Value describing the concept represented by this Content Item.

AconceptNameScheme

Coding Scheme Designator

AconceptNameMeaning

Code Meaning

Atext

A text value to be stored in this node.
AsrChildNodeID  Pointer to the variable receiving an identifier of the new SR node.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_VALUE_NOT_ALLOWED</td>
<td>A relationship value is not valid.</td>
</tr>
<tr>
<td>MC_VALUE_OUT_OF_RANGE</td>
<td>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).</td>
</tr>
<tr>
<td>MC_TEMP_FILE_ERROR</td>
<td>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.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**MC_SRH_Create_TEXT_Node**

Creates a new TEXT node.

**Synopsis**

```c
MC_STATUS MC_SRH_Create_TEXT_Node (  
    const char *AconceptNameValue,  
    const char *AconceptNameScheme,  
    const char *AconceptNameMeaning,  
    const char *Atext,  
    int *AsrNodeID  
)
```

* AconceptNameValue Code Value describing the concept represented by this Content Item.
* AconceptNameScheme Coding Scheme Designator
* AconceptNameMeaning Code Meaning
* Atext A text value to be stored in this node.
* AsrNodeID Pointer to the variable receiving an identifier of the new SR node.

**Return Value**

One of the enumerated **MC_STATUS** codes defined in “mcstatus.h”: 
<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_TEMP_FILE_ERROR</td>
<td>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.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**MC_SRH_Set_Concept_Name**

Sets the Concept Name values for the node.

**Synopsis**

```cpp
MC_STATUS MC_SRH_Set_Concept_Name (  
    int AsrNodeID,  
    const char *AconceptNameValue,  
    const char *AconceptNameScheme,  
    const char *AconceptNameMeaning
)
```

- `AsrNodeID` Numeric handle pointing to the SR node.
- `AconceptNameValue` Code Value describing the concept represented by this Content Item.
- `AconceptNameScheme` Coding Scheme Designator
- `AconceptNameMeaning` Code Meaning
- `AsrNodeID` Pointer to the variable receiving an identifier of the new SR node.

**Return Value**

One of the enumerated `MC_STATUS` codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_TEMP_FILE_ERROR</td>
<td>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.</td>
</tr>
</tbody>
</table>
MC_SYSTEM_ERROR

An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.

MC_SRH_Add_CODE_Child

Creates a new child CODE node under the provided parent.

Synopsis

```c
MC_STATUS MC_SRH_Add_CODE_Child (
    int AsrNodeID,
    SR_RELATIONSHIP Arelationship,
    const char *AconceptNameValue,
    const char *AconceptNameScheme,
    const char *AconceptNameMeaning,
    const char *AconceptCodeValue,
    const char *AconceptCodingScheme,
    const char *AconceptCodeMeaning,
    int *AsrChildNodeID
)
```

- **AsrNodeID**: An Identifier of the parent SR item to which the child is to be added.
- **Arelationship**: Enumerated value representing a relationship.
- **AconceptNameValue**: Code Value describing the concept represented by this Content Item.
- **AconceptNameScheme**: Coding Scheme Designator
- **AconceptNameMeaning**: Code Meaning
- **AconceptCodeValue**: Coded expression of the concept.
- **AconceptCodingScheme**: Coding Scheme Designator of the concept.
- **AconceptCodeMeaning**: Code Meaning of the concept.
- **AsrChildNodeID**: Pointer to the variable receiving an identifier of the new SR node.

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_VALUE_NOT_ALLOWED</td>
<td>The Arelationship value is not valid.</td>
</tr>
<tr>
<td>MC_TEMP_FILE_ERROR</td>
<td>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.</td>
</tr>
</tbody>
</table>
MC_SYSTEM_ERROR

An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.

MC_SRH_Create_CODE_Node

Creates a new CODE node.

Synopsis

MC_STATUS MC_SRH_Create_CODE_Node (  
    const char *AconceptNameValue,  
    const char *AconceptNameScheme,  
    const char *AconceptNameMeaning,  
    const char *AconceptCodeValue,  
    const char *AconceptCodingScheme,  
    const char *AconceptCodeMeaning,  
    int *AsrNodeID)

AconceptNameValue  

Code Value describing the concept represented by this Content Item.

AconceptNameScheme  

Coding Scheme Designator

AconceptNameMeaning  

Code Meaning

AconceptCodeValue  

Coded expression of the concept.

AconceptCodingScheme  

Coding Scheme Designator of the concept.

AconceptCodeMeaning  

Code Meaning of the concept.

AsrNodeID  

Pointer to the variable receiving an identifier of the new SR node.

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_TEMP_FILE_ERROR</td>
<td>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.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>
MC_SRH_Add_NUM_Child

Creates a new child NUM node under the provided parent.

Synopsis

MC_STATUS MC_SRH_Add_NUM_Child (  
    int AsrNodeID,  
    SR_RELATIONSHIP Arelationship,  
    const char *AconceptNameValue,  
    const char *AconceptNameScheme,  
    const char *AconceptNameMeaning,  
    const char *AnumericValue,  
    const char *AunitsCodeValue,  
    const char *AunitsCodingScheme,  
    const char *AunitsCodeMeaning,  
    int *AsrChildNodeID  
)

AsrNodeID          An Identifier of the parent SR item to which the child is to be added.
Arelationship      Enumerated value representing a relationship.
AconceptNameValue  Code Value describing the concept represented by this Content Item.
AconceptNameScheme Coding Scheme Designator
AconceptNameMeaning Code Meaning
AnumericValue       Numeric Value
AunitsCodeValue     Coded expression of measurement units.
AunitsCodingScheme Coding Scheme Designator of the measurement units.
AunitsCodeMeaning   Code Meaning of the measurement units.
AsrChildNodeID      Pointer to the variable receiving an identifier of the new SR node.

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_VALUE_OUT_OF_RANGE</td>
<td>A numeric Value was larger than could be accommodated by the attribute.</td>
</tr>
<tr>
<td></td>
<td>(E.g. setting an attribute with an unsigned int VR to the value 123.45,</td>
</tr>
<tr>
<td></td>
<td>or setting it to 128000, or setting it to -1).</td>
</tr>
<tr>
<td>MC_VALUE_NOT_ALLOWED</td>
<td>The Arelationship value is not valid.</td>
</tr>
</tbody>
</table>
**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_SYSTEM_ERROR**

An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.

---

**MC_SRH_Create_NUM_Node**

Creates a new NUM node. Note that the Measured Value Sequence (0040, A300) may be empty to convey the concept of a measurement whose value is unknown or missing, or a measurement or calculation failure.

**Synopsis**

```c
MC_STATUS MC_SRH_Create_NUM_Node(
    const char *AconceptNameValue,
    const char *AconceptNameScheme,
    const char *AconceptNameMeaning,
    const char *AnumericValue,
    const char *AunitsCodeValue,
    const char *AunitsCodingScheme,
    const char *AunitsCodeMeaning,
    int *AsrNodeID
)
```

- **AconceptNameValue**  
  Code Value describing the concept represented by this Content Item.

- **AconceptNameScheme**  
  Coding Scheme Designator

- **AconceptNameMeaning**  
  Code Meaning

- **AnumericValue**  
  Numeric Value. Can be NULL.

- **AunitsCodeValue**  
  Coded expression of measurement units.

- **AunitsCodingScheme**  
  Coding Scheme Designator of the measurement units.

- **AunitsCodeMeaning**  
  Code Meaning of the measurement units.

- **AsrNodeID**  
  Pointer to the variable receiving an identifier of the new SR node.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_NULL_POINTER_PARM</strong></td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
</tbody>
</table>
**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_SYSTEM_ERROR**

An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.

---

**MC_SRH_Set_NUM_Qualifier**

Sets the Qualification of Numeric Value in the Measured Value Sequence, or as a reason for the absence of Measured Value Sequence Item.

**Synopsis**

```c
MC_STATUS MC_SRH_Set_NUM_Qualifier (  
    int AsrNodeID,  
    const char *AqualifierCodeValue,  
    const char *AqualifierCodingScheme,  
    const char *AqualifierCodeMeaning  
)
```

- **AsrNodeID**
  An Identifier of the NUM SR item.

- **AqualifierCodeValue**
  Sets Qualification of Numeric Value.

- **AqualifierCodingScheme**
  Coding Scheme Designator

- **AqualifierCodeMeaning**
  Code Meaning

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_TEMP_FILE_ERROR</td>
<td>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.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>
**MC_SRH_Set_NUM_Next_Data**

Sets the next Numeric Value.

**Synopsis**

```c
MC_STATUS MC_SRH_Set_NUM_Next_Data (  
    int AsrNodeID,    
    char *AnumericValue
)
```

**AsrNodeID** an identifier of the NUM SR item.  
**AnumericValue** numeric value

**Return Value**

One of the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_VALUE_OUT_OF_RANGE</td>
<td>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).</td>
</tr>
<tr>
<td>MC_TEMP_FILE_ERROR</td>
<td>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.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**MC_SRH_Add_DATETIME_Child**

Creates a new child DATETIME node under the provided parent.

**Synopsis**

```c
MC_STATUS MC_SRH_Add_DATETIME_Child (  
    int AsrNodeID,  
    SR_RELATIONSHIP Arelationship,  
    const char *AconceptNameValue,  
    const char *AconceptNameScheme,  
    const char *AconceptNameMeaning,  
    const char *Adatetime,  
    int *AsrChildNodeID
)
```
AsrNodeID
An Identifier of the parent SR item to which the child is to be added.

Arelationship
Enumerated value representing a relationship.

AconceptNameValue
Code Value describing the concept represented by this Content Item.

AconceptNameScheme
Coding Scheme Designator

AconceptNameMeaning
Code Meaning

Adatetime
A date time string in the DICOM format.

AsrChildNodeID
Pointer to the variable receiving an identifier of the new SR node.

Return Value
One of the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_VALUE_NOT_ALLOWED</td>
<td>The Arelationship value is not valid.</td>
</tr>
<tr>
<td>MC_TEMP_FILE_ERROR</td>
<td>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.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

MC_SRH_Create_DATETIME_Node
Creates a new DATETIME node.

Synopsis

```c
MC_STATUS MC_SRH_Create_DATETIME_Node (  
    const char *AconceptNameValue,
    const char *AconceptNameScheme,
    const char *AconceptNameMeaning,
    const char *Adatetime,
    int *AsrNodeID  
)
```
AsrNodeID
An Identifier of the parent SR item to which the child is to be added.

Arelationship
Enumerated value representing a relationship.

AconceptNameValue
Code Value describing the concept represented by this Content Item.

AconceptNameScheme
Coding Scheme Designator

AconceptNameMeaning
Code Meaning

Adatetime
A date time string in the DICOM format.

AsrNodeID
Pointer to the variable receiving an identifier of the new SR node.

Return Value
One of the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_TEMP_FILE_ERROR</td>
<td>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.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

MC_SRH_Add_DATE_Child
Creates a new child DATE node under the provided parent.

Synopsis
MC_STATUS MC_SRH_Add_DATE_Child (  
    int AsrNodeID,
    SR_RELATIONSHIP Arelationship,
    const char *AconceptNameValue,
    const char *AconceptNameScheme,
    const char *AconceptNameMeaning,
    const char *Adate,
    int *AsrChildNodeID
)

AsrNodeID
An Identifier of the parent SR item to which the child is to be added.

Arelationship
Enumerated value representing a relationship.
**Synopsis**

```c
MC_STATUS MC_SRH_Create_DATE_Node (  
    const char *AconceptNameValue,  
    const char *AconceptNameScheme,  
    const char *AconceptNameMeaning,  
    const char *Adate,  
    int *AsrChildNodeID  
)
```

**Return Value**

One of the enumerated `MC_STATUS` codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_VALUE_NOT_ALLOWED</td>
<td>The Arelationship value is not valid.</td>
</tr>
<tr>
<td>MC_TEMP_FILE_ERROR</td>
<td>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.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**MC_SRH_Create_DATE_Node**

Creates a new DATE node.

**AconceptNameValue**

Code Value describing the concept represented by this Content Item.

**AconceptNameScheme**

Coding Scheme Designator.

**AconceptNameMeaning**

Code Meaning.

**Adate**

A date string in the DICOM format.

**AsrChildNodeID**

Pointer to the variable receiving an identifier of the new SR node.
**AconceptNameScheme**  Coding Scheme Designator

**AconceptNameMeaning**  Code Meaning

**Adate**  A date string in the DICOM format.

**AsrNodeID**  Pointer to the variable receiving an identifier of the new SR node.

---

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_NULL_POINTER_PARM</strong></td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td><strong>MC_TEMP_FILE_ERROR</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>MC_SYSTEM_ERROR</strong></td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

---

**MC_SRH_Add_TIME_Child**

Creates a new child DATETIME node under the provided parent.

**Synopsis**

```c
MC_STATUS MC_SRH_Add_TIME_Child (  
    int AsrNodeID,  
    SR_RELATIONSHIP Arelationship,  
    const char *AconceptNameValue,  
    const char *AconceptNameScheme,  
    const char *AconceptNameMeaning,  
    const char *Atime,  
    int *AsrChildNodeID  
)
```

**AsrNodeID**  An Identifier of the parent SR item to which the child is to be added.

**Arelationship**  Enumerated value representing a relationship.

**AconceptNameValue**  Code Value describing the concept represented by this Content Item.

**AconceptNameScheme**  Coding Scheme Designator

**AconceptNameMeaning**  Code Meaning
**Atime**
A time string in the DICOM format.

**AsrChildNodeID**
Pointer to the variable receiving an identifier of the new SR node.

### Return Value
One of the enumerated **MC_STATUS** codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_VALUE_NOT_ALLOWED</td>
<td>The Arelationship value is not valid.</td>
</tr>
<tr>
<td>MC_TEMP_FILE_ERROR</td>
<td>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.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

### MC_SRH_Create_TIME_Node
Creates a new TIME node.

#### Synopsis
```
MC_STATUS MC_SRH_Create_TIME_Node (  
    const char *AconceptNameValue,  
    const char *AconceptNameScheme,  
    const char *AconceptNameMeaning,  
    const char *Atime,  
    int *AsrNodeID  
)
```

**AsrNodeID**
An Identifier of the parent SR item to which the child is to be added.

**Arelationship**
Enumerated value representing a relationship.

**AconceptNameValue**
Code Value describing the concept represented by this Content Item.

**AconceptNameScheme**
Coding Scheme Designator.

**AconceptNameMeaning**
Code Meaning.

**Atime**
A time string in the DICOM format.
AsrNodeID

Pointer to the variable receiving an identifier of the new SR node.

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_TEMP_FILE_ERROR</td>
<td>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.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

MC_SRH_Add_UIDREF_Child

Creates a new child UIDREF node under the provided parent.

Synopsis

MC_STATUS MC_SRH_Add_UIDREF_Child (  
    int AsrNodeID,  
    SR_RELATIONSHIP Arelationship,  
    const char *AconceptNameValue,  
    const char *AconceptNameScheme,  
    const char *AconceptNameMeaning,  
    const char *AuidRef,  
    int *AsrChildNodeID  
)

AsrNodeID

An Identifier of the parent SR item to which the child is to be added.

Areleationship

Enumerated value representing a relationship.

AconceptNameValue

Code Value describing the concept represented by this Content Item.

AconceptNameScheme

Coding Scheme Designator

AconceptNameMeaning

Code Meaning

AuidRef

Unique Identifier (UID) of the entity identified by the Concept Name.

AsrChildNodeID

Pointer to the variable receiving an identifier of the new SR node.
**Return Value**

One of the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_VALUE_NOT_ALLOWED</td>
<td>The Arelationship value is not valid.</td>
</tr>
<tr>
<td>MC_TEMP_FILE_ERROR</td>
<td>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.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**MC_SRH_Create_UIDREF_Node**

Creates a new UIDREF node.

**Synopsis**

```c
MC_STATUS MC_SRH_Create_UIDREF_Node (  
    const char  *AconceptNameValue,  
    const char  *AconceptNameScheme,  
    const char  *AconceptNameMeaning,  
    const char  *AuidRef,  
    int         *AsrNodeID  
)
```

- **AconceptNameValue**  
  Code Value describing the concept represented by this Content Item.

- **AconceptNameScheme**  
  Coding Scheme Designator

- **AconceptNameMeaning**  
  Code Meaning

- **AuidRef**  
  Unique Identifier (UID) of the entity identified by the Concept Name.

- **AsrNodeID**  
  Pointer to the variable receiving an identifier of the new SR node.
Return Value

One of the enumerated `MC_STATUS` codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_TEMP_FILE_ERROR</td>
<td>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.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**MC_SRH_Add_PNAME_Child**

Creates a new child PNAME node under the provided parent.

**Synopsis**

```c
MC_STATUS MC_SRH_Add_PNAME_Child (  
    int AsrNodeID,  
    SR_RELATIONSHIP Arelationship,  
    const char *AconceptNameValue,  
    const char *AconceptNameScheme,  
    const char *AconceptNameMeaning,  
    const char *ApersonName,  
    int *AsrChildNodeID
)
```

- **AsrNodeID**: An Identifier of the parent SR item to which the child is to be added.
- **Arelationship**: Enumerated value representing a relationship.
- **AconceptNameValue**: Code Value describing the concept represented by this Content Item.
- **AconceptNameScheme**: Coding Scheme Designator
- **AconceptNameMeaning**: Code Meaning
- **ApersonName**: Person name of the person whose role is described by the Concept Name.
- **AsrChildNodeID**: Pointer to the variable receiving an identifier of the new SR node.

**Return Value**

One of the enumerated `MC_STATUS` codes defined in "mcstatus.h":

© Copyright Merge Healthcare Solutions Inc. 2023
### Value Meaning

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_VALUE_NOT_ALLOWED</td>
<td>The Arelationship value is not valid.</td>
</tr>
<tr>
<td>MC_TEMP_FILE_ERROR</td>
<td>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.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

### MC_SRH_Create_PNAME_Node

Creates a new PNAME node.

#### Synopsis

```c
MC_STATUS MC_SRH_Create_PNAME_Node ( 
    const char *AconceptNameValue,
    const char *AconceptNameScheme,
    const char *AconceptNameMeaning,
    const char *ApersonName,
    int *AsrNodeID
 )
```

- **AconceptNameValue**: Code Value describing the concept represented by this Content Item.
- **AconceptNameScheme**: Coding Scheme Designator
- **AconceptNameMeaning**: Code Meaning
- **ApersonName**: Person name of the person whose role is described by the Concept Name.
- **AsrNodeID**: Pointer to the variable receiving an identifier of the new SR node.

#### Return Value

One of the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
</tbody>
</table>
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_SYSTEM_ERROR  An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.

**MC_SRH_Add_SCOORD_Child**

Creates a new child SCOORD node under the provided parent.

**Synopsis**

```c
MC_STATUS MC_SRH_Add_SCOORD_Child (  
    int AsrNodeID,    
    SR_RELATIONSHIP Arelationship,  
    float Acolumn,  
    float ArOW,  
    SR_GRAPHIC_TYPE AgraphicType,  
    int *AsrChildNodeID
)
```

**AsrNodeID**  An Identifier of the parent SR item to which the child is to be added.

**Arelationship**  Enumerated value representing a relationship.

**Acolumn, Arow**  Spatial coordinates of a geometric region of interest in the DICOM image coordinate system. The IMAGE Content Item from which spatial coordinates are selected is denoted by a SELECTED FROM relationship. Data shall be provided as an ordered set of (column, row) pairs.

This attribute defines the type of geometry of the annotated region of interest. The following Enumerated Values are specified for image spatial coordinate geometries: SR_GRT_POINT = a single pixel denoted by a single (column, row) pair SR_GRT_MULTIPOINT = multiple pixels each denoted by an (column, row) pair SR_GRT_POLYLINE = a series of connected line segments with ordered vertices denoted by (column, row) pairs; if the first and last vertices are the same it is a closed polygon SR_GRT_CIRCLE = a circle defined by two (column, row) pairs. The first point is the central pixel. The second point is a pixel on the perimeter of the circle. SR_GRT_ELLIPSE = an ellipse defined by four pixel (column, row) pairs, the first two points specifying the endpoints of the major axis and the second two points specifying the endpoints of the minor axis of an ellipse.
AsrChildNodeID  Pointer to the variable receiving an identifier of the new SR node.

Remarks
The data can be set by using MC_SRH_Set_SCOORD_First_Data and MC_SRH_Set_SCOORD_Next_Data.

Return Value
One of the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_INCOMPATIBLE_VR</td>
<td>The attribute’s value representation cannot be derived from Value. See</td>
</tr>
<tr>
<td></td>
<td>the table below.</td>
</tr>
<tr>
<td>MC_VALUE_OUT_OF_RANGE</td>
<td>A numeric Value was larger than could be accommodated by the attribute.</td>
</tr>
<tr>
<td></td>
<td>(E.g. setting an attribute with an unsigned int VR to the value 123.45,</td>
</tr>
<tr>
<td></td>
<td>or setting it to 128000, or setting it to -1).</td>
</tr>
<tr>
<td>MC_VALUE_NOT_ALLOWED</td>
<td>The Arelationship value is not valid.</td>
</tr>
<tr>
<td>MC_TEMP_FILE_ERROR</td>
<td>If the attribute’s value is large, it may be stored in a temporary file</td>
</tr>
<tr>
<td></td>
<td>(if so configured). If a file I/O error occurs, this status is returned.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the</td>
</tr>
<tr>
<td></td>
<td>operating environment. A message describing the error has been written</td>
</tr>
<tr>
<td></td>
<td>to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

MC_SRH_Set_SCOORD_Next_Data

Appends the next coordinated pair to the SCOORD node.

Synopsis

```
MC_STATUS MC_SRH_Set_SCOORD_Next_Data (  
    int AsrNodeID,  
    float Acolumn,  
    float Arow  
)
```

AsrNodeID  An Identifier of the SCOORD SR item.

Acolumn, Arow  Spatial coordinates of a geometric region of interest in the DICOM image coordinate system. The IMAGE Content Item from which spatial coordinates are selected is denoted by a SELECTED FROM relationship. Data shall be provided as an ordered set of (column, row) pairs.
**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_Pointer_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_INCOMPATIBLE_VR</td>
<td>The attribute’s value representation cannot be derived from Value. See the table below.</td>
</tr>
<tr>
<td>MC_VALUE_OUT_OF_RANGE</td>
<td>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).</td>
</tr>
<tr>
<td>MC_TEMP_FILE_ERROR</td>
<td>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.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**MC_SRH_Create_SCOORD_Node**

Creates a new SCOORD node.

**Synopsis**

```c
MC_STATUS MC_SRH_Create_SCOORD_Node (  
    float Acolumn, 
    float Arow, 
    SR_GRAPHIC_TYPE AgraphicType, 
    int *AsrNodeID
)
```

Acolumn, Arow  

Spatial coordinates of a geometric region of interest in the DICOM image coordinate system. The IMAGE Content Item from which spatial coordinates are selected is denoted by a SELECTED FROM relationship. Data shall be provided as an ordered set of (column, row) pairs.
**AgraphicType**

This attribute defines the type of geometry of the annotated region of interest. The following Enumerated Values are specified for image spatial coordinate geometries:

- **SR_GRT_POINT** = a single pixel denoted by a single (column, row) pair
- **SR_GRT_MULTIPOINT** = multiple pixels each denoted by an (column, row) pair
- **SR_GRT_POLYLINE** = a series of connected line segments with ordered vertices denoted by (column, row) pairs; if the first and last vertices are the same it is a closed polygon
- **SR_GRT_CIRCLE** = a circle defined by two (column, row) pairs. The first point is the central pixel. The second point is a pixel on the perimeter of the circle.
- **SR_GRT_ELLIPSE** = an ellipse defined by four pixel (column, row) pairs, the first two points specifying the endpoints of the major axis and the second two points specifying the endpoints of the minor axis of an ellipse.

**AsrNodeID**

Pointer to the variable receiving an identifier of the new SR node.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_NULL_POINTER_PARM</strong></td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td><strong>MC_VALUE_OUT_OF_RANGE</strong></td>
<td>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).</td>
</tr>
<tr>
<td><strong>MC_TEMP_FILE_ERROR</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>MC_SYSTEM_ERROR</strong></td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>
MC_SRH_Add_TCOORD_R_Child

Creates a new child TCOORD node under the provided parent where data are provided as Referenced Sample Positions.

Synopsis

MC_STATUS MC_SRH_Add_TCOORD_R_Child (  
  int AsrNodeID,  
  SR_RELATIONSHIP Arelationship,  
  unsigned long ArefSamplePosition,  
  SR_TRANGE_TYPE AtempRangeType,  
  int *AsrChildNodeID  
)

AsrNodeID

An Identifier of the parent SR item to which the child is to be added.

Arelationship

Enumerated value representing a relationship.

ArefSamplePosition

The first value from the list of samples within a multiplex group specifying temporal points of the referenced data. Position of first sample is 1.

Defines the type of temporal extent of the region of interest. A temporal point (or instant of time) may be defined by a waveform sample offset (for a single waveform multiplex group only), time offset, or absolute time. Following enumerated values shall be used:

- SR_TRT_POINT a single temporal point
- SR_TRT_MULTIPOINT multiple temporal points
- SR_TRT_SEGMENT a range between two temporal points
- SR_TRT_MULTISEGMENT multiple segments, each denoted by two temporal points
- SR_TRT_BEGIN a range beginning at one temporal point, and extending beyond the end of the acquired data
- SR_TRT_END a range beginning before the start of the acquired data, and extending to (and including) the identified temporal point

AsrChildNodeID

Pointer to the variable receiving an identifier of the new SR node.

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
</table>

© Copyright Merge Healthcare Solutions Inc. 2023
MC_NORMAL_COMPLETION The function completed normally.
MC_INVALID_SR_ID The AsrNodeId value is not a valid SR tree management object ID.
MC_NULL_POINTER_PARM One or more of the pointer-type parameters was NULL.
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_VALUE_NOT_ALLOWED The Arelationship value is not valid.
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_SYSTEM_ERROR An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.

MC_SRH_Create_TCOORD_R_Node
Creates a new child TCOORD node under the provided parent where data are provided as Referenced Sample Positions.

Synopsis

MC_STATUS MC_SRH_Create_TCOORD_R_Node (unsigned long ArefSamplePosition, SR_TRANGE_TYPE AtempRangeType, int *AsrNodeID)

ArefSamplePosition The first value from the list of samples within a multiplex group specifying temporal points of the referenced data. Position of first sample is 1.

AtempRangeType Defines the type of temporal extent of the region of interest. A temporal point (or instant of time) may be defined by a waveform sample offset (for a single waveform multiplex group only), time offset, or absolute time. Following enumerated values shall be used:

SR_TRT_POINT a single temporal point
SR_TRT_MULTIPOINT multiple temporal points
SR_TRT_SEGMENT a range between two temporal points
SR_TRT_MULTISEGMENT multiple segments, each denoted by two temporal points
SR_TRT_BEGIN a range beginning at one temporal point, and extending beyond the end of the acquired data
SR_TRT_END a range beginning before the start of the acquired data, and extending to (and including) the identified temporal point

AsrNodeID Pointer to the variable receiving an identifier of the new SR node.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_NULL_POINTER_PARM</strong></td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td><strong>MC_VALUE_OUT_OF_RANGE</strong></td>
<td>A numeric <em>Value</em> 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).</td>
</tr>
<tr>
<td><strong>MC_TEMP_FILE_ERROR</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>MC_SYSTEM_ERROR</strong></td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**MC_SRH_Set_TCOORD_R_Next_Data**

Adds a value to the existing TCOORD node where data are provided as Referenced Sample Positions.

**Synopsis**

```c
MC_STATUS MC_SRH_Set_TCOORD_R_Next_Data(
    int AsrNodeID,
    unsigned long ArefSamplePosition
)
```

**AsrNodeID** An Identifier of the TCOORD SR item.

**ArefSamplePosition** Sequential value from the list of samples within a multiplex group specifying temporal points of the referenced data. Position of first sample is 1.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
</table>
MC_NORMAL_COMPLETION  The function completed normally.
MC_INVALID_SR_ID   The SRID value is not a valid SR tree management 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 Merge DICOM Toolkit log file.

MC_SRH_Add_TCOORD_O_Child

Creates a new child TCOORD node under the provided parent where data are provided as Referenced Time Offsets.

Synopsis

```c
MC_STATUS MC_SRH_Add_TCOORD_O_Child (  
    int AsrNodeID,  
    SR_RELATIONSHIP Arelationship,  
    const char *AtimeOffset,  
    SR_TRANGE_TYPE AtempRangeType,  
    int *AsrChildNodeID
)
```

- **AsrNodeID**: An Identifier of the parent SR item to which the child is to be added.
- **Arelationship**: Enumerated value representing a relationship.
- **AtimeOffset**: The first point from temporal points for reference by number of seconds after start of data Data shall be provided as a DICOM DS string format. Defines the type of temporal extent of the region of interest. A temporal point (or instant of time) may be defined by a waveform sample offset (for a single waveform multiplex group only), time offset, or absolute time. Following enumerated values shall be used:
  - SR_TRT_POINT a single temporal point
  - SR_TRT_MULTIPOINT multiple temporal points
  - SR_TRT_SEGMENT a range between two temporal points
  - SR_TRT_MULTISEGMENT multiple segments, each denoted by two temporal points
  - SR_TRT_BEGIN a range beginning at one temporal point, and extending beyond the end of the acquired data
  - SR_TRT_END a range beginning before the start of the acquired data, and extending to (and including) the identified temporal point

- **AsrChildNodeID**: Pointer to the variable receiving an identifier of the new SR node.
Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_VALUE_OUT_OF_RANGE</td>
<td>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).</td>
</tr>
<tr>
<td>MC_VALUE_NOT_ALLOWED</td>
<td>The Arelationship value is not valid.</td>
</tr>
<tr>
<td>MC_TEMP_FILE_ERROR</td>
<td>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.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**MC_SRH_Create_TCOORD_O_Node**

Creates a new TCOORD node where data are provided as Referenced Time Offsets.

**Synopsis**

```cpp
MC_STATUS MC_SRH_Create_TCOORD_O_Node (  
    const char *AtimeOffset,  
    SR_TRANGE_TYPE AtempRangeType,  
    int *AsrNodeID
)
```

*AtimeOffset*  
The first point from temporal points for reference by number of seconds after start of data Data shall be provided as a DICOM DS string format.
AtempRangeType

Defines the type of temporal extent of the region of interest. A temporal point (or instant of time) may be defined by a waveform sample offset (for a single waveform multiplex group only), time offset, or absolute time. Following enumerated values shall be used:

- SR_TRT_POINT a single temporal point
- SR_TRT_MULTIPOINT multiple temporal points
- SR_TRT_SEGMENT a range between two temporal points
- SR_TRT_MULTISEGMENT multiple segments, each denoted by two temporal points
- SR_TRT_BEGIN a range beginning at one temporal point, and extending beyond the end of the acquired data
- SR_TRT_END a range beginning before the start of the acquired data, and extending to (and including) the identified temporal point

AsrNodeID

Pointer to the variable receiving an identifier of the new SR node.

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_VALUE_OUT_OF_RANGE</td>
<td>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).</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

MC_SRH_Set_TCOORD_O_Next_Data

Adds a value to the existing TCOORD node where data are provided as Referenced Time Offsets.

Synopsis

```c
MC_STATUS MC_SRH_Set_TCOORD_O_Next_Data (  
    int AsrNodeID,  
    const char *AtimeOffset  
)
```

AsrNodeID An Identifier of the TCOORD SR item.
**AtimeOffset**  Sequential temporal point for reference by number of seconds after start of data. Data shall be provided as a DICOM DS string format.

**Return Value**
One of the enumerated **MC_STATUS** codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**MC_SRH_Add_TCOORD_D_Child**

Creates a new child TCOORD node under the provided parent where data are provided as Referenced DateTime.

**Synopsis**

```c
MC_STATUS MC_SRH_Add_TCOORD_D_Child (  
    int AsrNodeID,  
    SR_RELATIONSHIP Arelationship,  
    const char *AdateTimeOffset,  
    SR_TRANGE_TYPE AtempRangeType,  
    int *AsrChildNodeID  
)
```

**AsrNodeID**  An Identifier of the parent SR item to which the child is to be added.

**Arelationship**  Enumerated value representing a relationship.

**AdateTimeOffset**  The first point from temporal points list for reference by absolute time. Data shall be provided as DICOM DT string.

**AtempRangeType**  Defines the type of temporal extent of the region of interest. A temporal point (or instant of time) may be defined by a waveform sample offset (for a single waveform multiplex group only), time offset, or absolute time. Following enumerated values shall be used:

- SR_TRT_POINT a single temporal point
- SR_TRT_MULTIPOINT multiple temporal points
SR_TRT_SEGMENT a range between two temporal points
SR_TRT_MULTISEGMENT multiple segments, each denoted by two temporal points
SR_TRT_BEGIN a range beginning at one temporal point, and extending beyond the end of the acquired data
SR_TRT_END a range beginning before the start of the acquired data, and extending to (and including) the identified temporal point

AsrChildNodeID  Pointer to the variable receiving an identifier of the new SR node.

Return Value
One of the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_VALUE_NOT_ALLOWED</td>
<td>The Arelationship value is not valid.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

MC_SRH_Create_TCOORD_D_Node

Creates a new TCOORD node where data are provided as Referenced DateTime.

Synopsis
MC_STATUS MC_SRH_Create_TCOORD_D_Node (  
    const char *AdateTimeOffset,  
    SR_TRANGE_TYPE AtempRangeType,  
    int *AsrNodeID  
)

AsrNodeID  An Identifier of the parent SR item to which the child is to be added.
Arelationship  Enumerated value representing a relationship.
AdateTimeOffset  The first point from temporal points list for reference by absolute time. Data shall be provided as DICOM DT string.
**AtempRangeType**

Defines the type of temporal extent of the region of interest. A temporal point (or instant of time) may be defined by a waveform sample offset (for a single waveform multiplex group only), time offset, or absolute time. Following enumerated values shall be used:

- **SR_TRT_POINT** a single temporal point
- **SR_TRT_MULTIPOINT** multiple temporal points
- **SR_TRT_SEGMENT** a range between two temporal points
- **SR_TRT_MULTISEGMENT** multiple segments, each denoted by two temporal points
- **SR_TRT_BEGIN** a range beginning at one temporal point, and extending beyond the end of the acquired data
- **SR_TRT_END** a range beginning before the start of the acquired data, and extending to (and including) the identified temporal point

**AsrChildNodeID**

Pointer to the variable receiving an identifier of the new SR node.

### Remarks

None.

### Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_NULL_POINTER_PARM</strong></td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td><strong>MC_SYSTEM_ERROR</strong></td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

### MC_SRH_Set_TCOORD_D_Next_Data

Adds a value to the existing TCOORD node where data are provided as Referenced DateTime.

**Synopsis**

```c
MC_STATUS MC_SRH_Set_TCOORD_D_Next_Data (int AsrNodeID,
                                       const char *AdatetimeOffset)
```

- **AsrNodeID** An Identifier of the TCOORD SR item.
A dateTimeOffset
Sequential point from temporal points list for reference
by absolute time. Data shall be provided as DICOM DT
string.

Return Value
One of the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

MC_SRH_Add_COMPOSITE_Child
Creates a new child COMPOSITE node under the provided parent.

Synopsis

```c
MC_STATUS MC_SRH_Add_COMPOSITE_Child (  
    int AsrNodeID,  
    SR_RELATIONSHIP Arelationship,  
    const char *AsopClassUid,  
    const char *AsopInstanceUid,  
    int *AsrChildNodeID  
)  
```

AsrNodeID
An Identifier of the parent SR item to which the child is to be added.

Arelationship
Enumerated value representing a relationship.

AsopClassUid
A SOP Class UID of a Composite object which is not an Image or Waveform.

AsopInstanceUid
A SOP Instance UID of a Composite object which is not an Image or Waveform.

AsrChildNodeID
Pointer to the variable receiving an identifier of the new SR node.

Return Value
One of the enumerated MC_STATUS codes defined in “mcstatus.h”: 

**MC_SRH_Create_COMPOSITE_Node**

Creates a new COMPOSITE node.

**Synopsis**

```c
MC_STATUS MC_SRH_Create_COMPOSITE_Node (  
    const char *AsopClassUid,  
    const char *AsopInstanceUid,  
    int *AsrNodeID  
)
```

- **AsopClassUid**: A SOP Class UID of a Composite object which is not an Image or Waveform.
- **AsopInstanceUid**: A SOP Instance UID of a Composite object which is not an Image or Waveform.
- **AsrNodeID**: Pointer to the variable receiving an identifier of the new SR node.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_VALUE_NOT_ALLOWED</td>
<td>The Arelationship value is not valid.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>
**MC_SRH_Add_IMAGE_Child**

Creates a new child IMAGE node under the provided parent.

**Synopsis**

```c
MC_STATUS MC_SRH_Add_IMAGE_Child(
    int AsrNodeID,
    SR_RELATIONSHIP Arelationship,
    const char *AsopClassUid,
    const char *AsopInstanceUid,
    int *AsrChildNodeID
)
```

- **AsrNodeID** - An Identifier of the parent SR item to which the child is to be added.
- **Arelationship** - Enumerated value representing a relationship.
- **AsopClassUid** - A SOP Class UID of an Image object.
- **AsopInstanceUid** - A SOP Instance UID of an Image object.
- **AsrChildNodeID** - Pointer to the variable receiving an identifier of the new SR node.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_VALUE_NOT_ALLOWED</td>
<td>The Arelationship value is not valid.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**MC_SRH_Create_IMAGE_Node**

Creates a new IMAGE node.

**Synopsis**

```c
MC_STATUS MC_SRH_Create_IMAGE_Node(
    const char *AsopClassUid,
    const char *AsopInstanceUid,
    int *AsrNodeID
)
```
AsopClassUid  An SOP Class UID of an Image object.
AsopInstanceUid  An SOP Instance UID of an Image object.
AsrChildNodeID  Pointer to the variable receiving an identifier of the new SR node.

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**MC_SRH_Set_IMAGE_Frames**

Sets frame numbers within Referenced SOP Instance to which the reference applies.

**Synopsis**

```c
MC_STATUS MC_SRH_Set_IMAGE_Frames (  
    int AsrNodeID,  
    long AreferencedFrames[],  
    long AframesArraySize  
)
```

AsrNodeID  An Identifier of the IMAGE SR item.
AreferencedFrames  Identifies the frame numbers within the Referenced SOP Instance to which the reference applies. The first frame shall be denoted as frame number 1.
AframesArraySize  Size of the AreferencedFrames array.

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
</tbody>
</table>
### 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_SYSTEM_ERROR
An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.

---

**MC_SRH_Add_WAVEFORM_Child**

Creates a new child WAVEFORM node under the provided parent.

#### Synopsis

```c
MC_STATUS MC_SRH_Add_WAVEFORM_Child (
    int AsrNodeID,
    SR_RELATIONSHIP Arelationship,
    const char *AsopClassUid,
    const char *AsopInstanceUid,
    int *AsrChildNodeID
)
```

- **AsrNodeID**: An Identifier of the parent SR item to which the child is to be added.
- **Arelationship**: Enumerated value representing a relationship.
- **AsopClassUid**: An SOP Class UID of a Waveform object.
- **AsopInstanceUid**: An SOP Instance UID of a Waveform object.
- **AsrChildNodeID**: Pointer to the variable receiving an identifier of the new SR node.

#### Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_INVALID_SR_ID</strong></td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td><strong>MC_NULL_POINTER_PARM</strong></td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td><strong>MC_VALUE_NOT_ALLOWED</strong></td>
<td>The Arelationship value is not valid.</td>
</tr>
</tbody>
</table>
**MC_SYSTEM_ERROR**

An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.

**MC_SRH_Create_WAVEFORM_Node**

Creates a new WAVEFORM node.

**Synopsis**

```c
MC_STATUS MC_SRH_Create_WAVEFORM_Node (  
    const char *AsopClassUid,  
    const char *AsopInstanceUid,  
    int *AsrNodeID  
)
```

- **AsopClassUid**: An SOP Class UID of a Waveform object.
- **AsopInstanceUid**: An SOP Instance UID of a Waveform object.
- **AsrNodeID**: Pointer to the variable receiving an identifier of the new SR node.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_VALUE_NOT_ALLOWED</td>
<td>The Arelationship value is not valid.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**MC_SRH_Set_WAVEFORM_Channels**

Sets channels referenced within the Referenced SOP Instance to which the reference applies.

**Synopsis**

```c
MC_STATUS MC_SRH_Set_WAVEFORM_Channels (  
    int AsrNodeID,  
    unsigned short ArefChannels[],  
    long ArefChannelsSize  
)
```

- **AsrNodeID**: An Identifier of the SR WAVEFORM node.
**ArefChannels**

A multi-value attribute which lists the channels referenced. Each channel is specified as a pair of values (M,C), where the first value is the sequence item number of the Waveform Sequence (5400,0100) attribute in the referenced object (i.e. the Multiplex Group Number), and the second value is the sequence item number of the Channel Definition Sequence (003A,0200) attribute (i.e., the Channel Number) within the multiplex group.

**ArefChannelsSize**  
Size of the ArefChannels array.

### Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_VALUE_NOT_ALLOWED</td>
<td>The ArefChannelsSize value is &lt; 1.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**MC_SRH_Add_CONTAINER_Child**

Creates a new child CONTAINER node under the provided parent.

### Synopsis

```c
MC_STATUS MC_SRH_Add_CONTAINER_Child (  
    int AsrNodeID,  
    SR_RELATIONSHIP Arelationship,  
    SR_CONT_CONTINUITY Acontinuity,  
    const char *AtemplateId,  
    int *AsrChildNodeID
)
```

- **AsrNodeID**  
  An Identifier of the parent SR item to which the child is to be added.

- **Arelationship**  
  Enumerated value representing a relationship.

- **Acontinuity**  
  Enumerated value specifies whether or not its contained Content Items are logically linked in a continuous textual flow, or are separate items.
A*templateId* Template Identifier that describes the content of this Content Item and its subsidiary Content Items.

A*srChildNodeId* Pointer to the variable receiving an identifier of the new SR node.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_VALUE_NOT_ALLOWED</td>
<td>The Arelationship value is not valid.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**MC_SRH_Create_CONTAINER_Node**

Creates a new CONTAINER node.

**Synopsis**

```c
MC_STATUS MC_SRH_Create_CONTAINER_Node (  
    SR_CONT_CONTINUITY Acontinuity,  
    const char *AtemplateId,  
    int *AsrNodeID  
)
```

*Acontinuity* Enumerated value specifies whether or not its contained Content Items are logically linked in a continuous textual flow, or are separate items.

*AtemplateId* DICOM Template Identifier that describes the content of this Content Item and its subsidiary Content Items. Can be NULL.

*AsrNodeID* Pointer to the variable receiving an identifier of the new SR node.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_VALUE_NOT_ALLOWED</td>
<td>The Arelationship value is not valid.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>
**MC_NORMAL_COMPLETION**
The function completed normally.

**MC_NULL_POINTER_PARM**
One or more of the pointer-type parameters 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 Merge DICOM Toolkit log file.

---

**MC_SRH_Add_TABLE_Child**
Create a new child TABLE node under the provided parent.

**Synopsis**

```c
MC_STATUS MC_SRH_Add_TABLE_Child (  
    SR_RELATIONSHIP Arelationship,  
    const char *AconceptNameValue,  
    const char *AconceptNameScheme,  
    const char *AconceptNameMeaning,  
    int Arows,  
    int Acolumns,  
    TABLE_ROW_DEF *AtableRowDefs,  
    int AtableRowDefsCount,  
    TABLE_COL_DEF *AtableColDefs,  
    int AtableColDefsCount,  
    TABLE_CELL_VALUES *AtableCellValues,  
    int AtableCellValuesCount,  
    int *AsrChildNodeID  
)
```

- **AsrNodeID**
  An Identifier of the parent SR item to which the child is to be added.

- **Arelationship**
  Enumerated value representing a relationship.

- **AconceptNameValue**
  Code Value describing the concept represented by this Content Item.

- **AconceptNameScheme**
  Coding Scheme Designator

- **AconceptNameMeaning**
  Code Meaning

- **Arows**
  The number of rows in this TABLE item

- **Acolumns**
  The number of columns in this TABLE item

- **AtableRowDefs**
  The concepts that define the meaning of the rows of the table

- **AtableRowDefsCount**
  The number of items in the Table Row Definition Sequence

- **AtableColDefs**
  The concepts that define the meaning of the columns of the table
AtableColDefsCount

The number of items in the Table Column Definition Sequence.

AtableCellValues

The values of each populated cell in the table, identified by row and column.

AtableCellValuesCount

The number of items in the Cell Values Sequence.

AsrChildNodeID

Pointer to the variable receiving an identifier of the new SR node.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_INVALID_SR_ID</strong></td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td><strong>MC_NULL_POINTER_PARM</strong></td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td><strong>MC_VALUE_NOT_ALLOWED</strong></td>
<td>The Arelationship value is not valid.</td>
</tr>
<tr>
<td><strong>MC_SYSTEM_ERROR</strong></td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**MC_SRH_Create_TABLE_Node**

Creates a new TABLE node.

**Synopsis**

```c
MC_STATUS MC_SRH_Create_TABLE_Node (  
    const char *AconceptNameValue,  
    const char *AconceptNameScheme,  
    const char *AconceptNameMeaning,  
    int Arows,  
    int Acolumns,  
    TABLE_ROW_DEF *AtableRowDefs,  
    int AtableRowDefsCount,  
    TABLE_COL_DEF *AtableColDefs,  
    int AtableColDefsCount,  
    TABLE_CELL_VALUES *AtableCellValues,  
    int AtableCellValuesCount,  
    int *AsrNodeID)
```

**AconceptNameValue**

Code Value describing the concept represented by this Content Item.

**AconceptNameScheme**

Coding Scheme Designator.
<table>
<thead>
<tr>
<th>Concept</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arows</td>
<td>The number of rows in this TABLE item</td>
</tr>
<tr>
<td>Acolumns</td>
<td>The number of columns in this TABLE item</td>
</tr>
<tr>
<td>AtableRowDefs</td>
<td>The concepts that define the meaning of the rows of the table</td>
</tr>
<tr>
<td>AtableRowDefsCount</td>
<td>The number of items in the Table Row Definition Sequence</td>
</tr>
<tr>
<td>AtableColDefs</td>
<td>The concepts that define the meaning of the columns of the table</td>
</tr>
<tr>
<td>AtableColDefsCount</td>
<td>The number of items in the Table Column Definition Sequence</td>
</tr>
<tr>
<td>AtableCellValue</td>
<td>The values of each populated cell in the table, identified by row and column</td>
</tr>
<tr>
<td>AtableCellValueCount</td>
<td>The number of items in the Cell Values Sequence</td>
</tr>
<tr>
<td>AsrNodeID</td>
<td>Pointer to the variable receiving an identifier of the new SR node</td>
</tr>
</tbody>
</table>

### Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_VALUE_NOT_ALLOWED</td>
<td>The Arelationship value is not valid.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

### MC_SRH_Set_TABLE_Next_Row_Definition

Add a new item to the Table Row Definition Sequence in the TABLE Content Item.

#### Synopsis

```c
MC_STATUS MC_SRH_Set_TABLE_Next_Row_Definition (  
    int AsrNodeID,  
    TABLE_ROW_DEF *AtableRowDef  
)
```

*AsrNodeID* An Identifier of the TABLE SR item.
**ATableRowDef**  
Pointer to the structure representing the Table Row Definition Sequence item to be added to the TABLE Content Item

### Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_INVALID_SR_ID</strong></td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td><strong>MC_NULL_POINTER_PARM</strong></td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td><strong>MC_SYSTEM_ERROR</strong></td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

### MC_SRH_Set_TABLE_Next_Column_Definition

Add a new item to the Table Column Definition Sequence in the TABLE Content Item.

#### Synopsis

```c
MC_STATUS MC_SRH_Set_TABLE_Next_Column_Definition (  
    int AsrNodeID,  
    TABLE_COL_DEF *AtableColDef  
)
```

- **AsrNodeID**: An Identifier of the TABLE SR item.
- **AtableColDef**: Pointer to the structure representing the Table Column Definition Sequence item to be added to the TABLE Content Item

### Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_INVALID_SR_ID</strong></td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td><strong>MC_NULL_POINTER_PARM</strong></td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td><strong>MC_SYSTEM_ERROR</strong></td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

© Copyright Merge Healthcare Solutions Inc. 2023
MC_SRH_Set_TABLE_Next_Cell_Values

Creates a new CONTAINER node.

Synopsis

MC_STATUS MC_SRH_Set_TABLE_Next_Cell_Values (  
    int AsrNodeID,  
    TABLE_CELL_VALUES *AtableCellValues  
)

AsrNodeID  An Identifier of the TABLE SR item.
AtableCellValues  Pointer to the structure representing the Cell Values Sequence item to be added to the TABLE

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

MC_SRH_Add_Reference

Adds a Referenced Content Item.

Synopsis

MC_STATUS MC_SRH_Add_Reference (  
    int AsrNodeID,  
    SR_RELATIONSHIP Arelationship,  
    int AsrRefNodeID  
)

AsrNodeID  An Identifier of the parent SR item to which the child is to be added.
Arelationship  Enumerated value representing a relationship.
AsrRefNodeID  SR node that will be referenced from the current node.
Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_VALUE_NOT_ALLOWED</td>
<td>The Arelationship value is not valid.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

MC_SRH_Get_Reference

Returns a Referenced Content Item.

Synopsis

MC_STATUS MC_SRH_Get_Reference (  
    int AsrNodeID,  
    int *AsrRefNodeID  
)

AsrNodeID  An Identifier of the parent SR item to which the child is to be added.

AsrRefNodeID SR node that is referenced from the current node.

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

MC_SRH_Get_NodeType

Returns node type for the specified SR node.
**Synopsis**

MC_STATUS MC_SRH_Get_NodeType(
    int AsrNodeID,
    SR_Content_Type *AnodeType
)

- **AsrNodeID**: An Identifier of the SR node.
- **AnodeType**: Pointer to the variable that receives an enumerated node type from the list below

**Enumerated values for the Node Type**

- **SR_NODE_TEXT**: Free text, narrative description of unlimited length. May also be used to provide a label or identifier value.
- **SR_NODE_NUM**: Numeric value fully qualified by coded representation of the measurement name and unit of measurement.
- **SR_NODE_CODE**: Categorical coded value. Representation of nominal or non-numeric ordinal values.
- **SR_NODE_DATETIME**: Date and time of occurrence of the type of event denoted by the Concept Name.
- **SR_NODE_DATE**: Date of occurrence of the type of event denoted by the Concept Name.
- **SR_NODE_TIME**: Time of occurrence of the type of event denoted by the Concept Name.
- **SR_NODE_UIDREF**: Unique Identifier (UID) of the entity identified by the Concept Name.
- **SR_NODE_PNAME**: Person name of the person whose role is described by the Concept Name.
- **SR_NODE_COMPOSITE**: A reference to one Composite SOP Instance which is not an Image or Waveform.
- **SR_NODE_IMAGE**: A reference to one Image. IMAGE Content Item may convey a reference to a Softcopy Presentation State associated with the Image.
- **SR_NODE_WAVEFORM**: A reference to one Waveform.
- **SR_NODE_SCOORD**: Spatial coordinates of a geometric region of interest in the DICOM image coordinate system. The IMAGE Content Item from which spatial coordinates are selected is denoted by a SELECTED FROM relationship.
- **SR_NODE_TCOORD_D**: Temporal Coordinates (i.e. time or event based coordinates) of a region of interest in the DICOM waveform coordinate system. The WAVEFORM or IMAGE or SCOODD Content Item from which Temporal Coordinates are selected is denoted by a SELECTED
FROM relationship. The value is stored as a Referenced DateTime here.

**SR_NODE_TCOORD_O**
Temporal Coordinates (i.e. time or event based coordinates) of a region of interest in the DICOM waveform coordinate system. The WAVEFORM or IMAGE or SCOORD Content Item from which Temporal Coordinates are selected is denoted by a SELECTED FROM relationship. The value is stored as Referenced Time Offsets here.

**SR_NODE_TCOORD_R**
Temporal Coordinates (i.e. time or event based coordinates) of a region of interest in the DICOM waveform coordinate system. The WAVEFORM or IMAGE or SCOORD Content Item from which Temporal Coordinates are selected is denoted by a SELECTED FROM relationship. The value is stored as Referenced Sample Positions here.

**SR_NODE_CONTAINER**
CONTAINER groups Content Items and defines the heading or category of observation that applies to that content. The heading describes the content of the CONTAINER Content Item and may map to a document section heading in a printed or displayed document.

**SR_REFERENCE**
This item is not a child SR content node. It is a reference to another node.

### Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_INVALID_SR_ID</strong></td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td><strong>MC_NULL_POINTER_PARM</strong></td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td><strong>MC_EMPTY_VALUE</strong></td>
<td>The attribute has no value assigned to it.</td>
</tr>
<tr>
<td><strong>MC_NULL_VALUE</strong></td>
<td>The attribute’s value is NULL (i.e. its length is zero).</td>
</tr>
<tr>
<td><strong>MC_SYSTEM_ERROR</strong></td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

### MC_SRH_Get_First_Child

Returns the ID of the first child in the list of children that the parent node maintains.

**Synopsis**

```c
MC_STATUS MC_SRH_Get_First_Child (  
    int AsrNodeID,  
    int *AsrChildNodeID,  
```
AsrNodeID

The identifier assigned to this SR tree item.

Upon successful completion, the item object identifier of the first SR tree management node in the SR tree management entity AsrChildNodeID is returned here. At a given parent SRID a pointer to the next record is maintained.

Arelationship

Enumerated value representing a relationship between parent and child nodes SR_REL_CONTAINS,
SR_REL_HAS_OBS_CONTEXT,
SR_REL_HAS_ACQ_CONTEXT,
SR_REL_HAS_CONCEPT_MOD, SR_REL_HAS_PROPERTIES,
SR_REL_INFERRED_FROM, SR_REL_SELECTED_FROM

AnodeType

Returns an enumerated node type. See MC_SRH_Get_NodeType function.

AisLast

Upon successful completion, this parameter is set to true (non-zero) if the first record in the SR tree management entity AsrChildNodeID is also the last record (i.e. it is the only element)

Remarks

This function is an extension of the MC_SR_Get_First_Child function and it is using it internally.

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

MC_SRH_Get_Next_Child

Returns the ID of the next child in the list of children that the parent node maintains.

Synopsis

MC_STATUS MC_SRH_Get_Next_Child ( 

SR_RELATIONSHIP *Arelationship,
SR_CONTENT_TYPE *AnodeType,
int *AisLast
)
int AsrNodeID,  
int *AsrChildNodeID,  
SR_RELATIONSHIP *Arelationship,  
SR_CONTENT_TYPE *AnodeType,  
int *AisLast  
)

AsrNodeID The identifier assigned to this SR tree item.  
Upon successful completion, the item object identifier of  
the next SR tree management node in the SR tree  
management entity AsrChildNodeID is returned here. At a  
given parent SRID a pointer to the next record is  
maintained.

Enumerated value representing a relationship between  
parent and child nodes SR_REL_CONTAINS,  
SR_REL_HAS_OBS_CONTEXT,

Arelationship SR_REL_HAS_ACQ_CONTEXT,  
SR_REL_HAS_CONCEPT_MOD,  
SR_REL_HAS_PROPERTIES, SR_REL_INFERRED_FROM,  
SR_REL_SELECTED_FROM

AnodeType Returns an enumerated node type. See  
MC_SRH_Get_NodeType function.

AisLast Upon successful completion, this parameter is set to true  
(non-zero) if the next record in the SR tree management  
entity AsrChildNodeID is also the last record.

Remarks This function is an extension of the MC_SR_Get_Next_Child function and it is using it internally.

Return Value One of the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

MC_SRH_Get_Concept_Name

Returns the Concept Name values for the node.
Synopsis

MC_STATUS MC_SRH_Get_Concept_Name
{
    int AsrNodeID,
    char AconceptNameValue[],
    int AconceptNameValueSize,
    char AconceptNameScheme[],
    int AconceptNameSchemeSize,
    char AconceptNameMeaning[],
    int AconceptNameMeaningSize
}

AsrNodeID An Identifier of the SR item.
AconceptNameValue Returns Code Value describing the concept represented by this Content Item.
AconceptNameValueSize Code Value buffer size
AconceptNameScheme Returns Coding Scheme Designator
AconceptNameSchemeSize Coding Scheme Designator buffer size
AconceptNameMeaning Returns Code Meaning
AconceptNameMeaningSize Code Meaning buffer size

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_EMPTY_VALUE</td>
<td>The attribute has no value assigned to it.</td>
</tr>
<tr>
<td>MC_NULL_VALUE</td>
<td>The attribute’s value is NULL (i.e. its length is zero).</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

MC_SRH_Get_TEXT_Data

Returns a TEXT node data.

Synopsis

MC_STATUS MC_SRH_Get_TEXT_Data
{
    int AsrNodeID,
    char Atext[],
    int AtextSize
}
**AsrNodeID**  
An Identifier of the TEXT SR item.

**Atext**  
Returns A text value stored in this node

**AtextSize**  
Text buffer size

### Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_INVALID_SR_ID</strong></td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td><strong>MC_EMPTY_VALUE</strong></td>
<td>The attribute has no value assigned to it.</td>
</tr>
<tr>
<td><strong>MC_NULL_VALUE</strong></td>
<td>The attribute’s value is NULL (i.e. its length is zero).</td>
</tr>
<tr>
<td><strong>MC_SYSTEM_ERROR</strong></td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

### MC_SRH_Get_CODE_Data

Returns the Code Name values for the node.

#### Synopsis

```c
MC_STATUS MC_SRH_Get_CODE_Data (  
    int AsrNodeID,  
    char AconceptCodeValue[],  
    int AconceptCodeValueSize,  
    char AconceptCodingScheme[],  
    int AconceptCodingSchemeSize,  
    char AconceptCodeMeaning[],  
    int AconceptCodeMeaningSize
)
```

**AsrNodeID**  
An Identifier of the CODE SR item.

**AconceptCodeValue**  
Returns Code Value

**AconceptCodeValueSize**  
Code Value buffer size

**AconceptCodingScheme**  
Returns Coding Scheme Designator

**AconceptCodingSchemeSize**  
Coding Scheme Designator buffer size

**AconceptCodeMeaning**  
Returns Code Meaning

**AconceptCodeMeaningSize**  
Code Meaning buffer size
**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_EMPTY_VALUE</td>
<td>The attribute has no value assigned to it.</td>
</tr>
<tr>
<td>MC_NULL_VALUE</td>
<td>The attribute’s value is NULL (i.e. its length is zero).</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**MC_SRH_Get_NUM_Data**

Reads value from the NUM node.

**Synopsis**

```c
MC_STATUS MC_SRH_Get_NUM_Data (  
    int AsrNodeID,  
    char AnumericValue[],  
    int AnumericValueSize,  
    int *AvalueCount,  
    char AunitsCodeValue[],  
    int AunitsCodeValueSize,  
    char AunitsCodingScheme[],  
    int AunitsCodingSchemeSize,  
    char AunitsCodeMeaning[],  
    int AunitsCodeMeaningSize  
)
```

- **AsrNodeID** - An Identifier of the NUM SR item.
- **AnumericValue** - Numeric Value
- **AnumericValueSize** - Numeric Value buffer size
- **AvalueCount** - Number of values stored
- **AunitsCodeValue** - Coded expression of measurement units
- **AunitsCodeValueSize** - AunitsCodeValue buffer size
- **AunitsCodingScheme** - Coding Scheme Designator of the measurement units.
- **AunitsCodingSchemeSize** - AunitsCodingScheme buffer size
- **AunitsCodeMeaning** - Code Meaning of the measurement units.
The Measured Value Sequence (0040, A300) may be empty to convey the concept of a measurement whose value is unknown or missing, or a measurement or calculation failure in that case function will return the MC_EMPTY_VALUE code.

### Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_EMPTY_VALUE</td>
<td>The attribute has no value assigned to it.</td>
</tr>
<tr>
<td>MC_NULL_VALUE</td>
<td>The attribute's value is NULL (i.e. its length is zero).</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

### MC_SRH_Get_NUM_Next_Data

Reads next value from the NUM node.

#### Synopsis

```c
MC_STATUS MC_SRH_Get_NUM_Next_Data (  
    int AsrNodeID,  
    char NumericValue[],  
    int NumericValueSize
)
```

- **AsrNodeID** An Identifier of the NUM SR item.
- **NumericValue** Numeric Value
- **NumericValueSize** Numeric Value buffer size

### Remarks

The Measured Value Sequence (0040,A300) may be empty to convey the concept of a measurement whose value is unknown or missing, or a measurement or calculation failure in that case function will return the MC_EMPTY_VALUE code.
**Return Value**

One of the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_EMPTY_VALUE</td>
<td>The attribute has no value assigned to it.</td>
</tr>
<tr>
<td>MC_NULL_VALUE</td>
<td>The attribute’s value is NULL (i.e. its length is zero).</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**MC_SRH_Get_NUM_Qualifier**

Gets Qualification of Numeric Value in Measured Value Sequence, or reason for absence of Measured Value Sequence Item.

**Synopsis**

```c
MC_STATUS MC_SRH_Get_NUM_Qualifier (  
    int AsrNodeID,  
    char AqualifierCodeValue[],  
    int AqualifierCodeValueSize,  
    char AqualifierCodingScheme[],  
    int AqualifierCodingSchemeSize,  
    char AqualifierCodeMeaning[],  
    int AqualifierCodeMeaningSize
)
```

- **AsrNodeID**  
  An Identifier of the NUM SR item.

- **AqualifierCodeValue**  
  Sets Qualification of Numeric Value

- **AqualifierCodeValueSize**  
  Size of AqualifierCodeValue buffer

- **AqualifierCodingScheme**  
  Coding Scheme Designator

- **AqualifierCodingSchemeSize**  
  Size of AqualifierCodingScheme buffer

- **AqualifierCodeMeaning**  
  Code Meaning

- **AqualifierCodeMeaningSize**  
  Size of AqualifierCodeMeaning buffer
Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_EMPTY_VALUE</td>
<td>The attribute has no value assigned to it.</td>
</tr>
<tr>
<td>MC_NULL_VALUE</td>
<td>The attribute's value is NULL (i.e. its length is zero).</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

MC_SRH_Get_DATETIME_Data

Returns a DATETIME node data.

Synopsis

MC_STATUS MC_SRH_Get_DATETIME_Data (  
  int AsrNodeID,  
  char AdateTime[],  
  int AdateTimeSize  
)

AsrNodeID  
An Identifier of the DATETIME SR item.

AdateTime  
Returns a text value stored in this node

AdateTimeSize  
Datet ime buffer size

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_EMPTY_VALUE</td>
<td>The attribute has no value assigned to it.</td>
</tr>
<tr>
<td>MC_NULL_VALUE</td>
<td>The attribute's value is NULL (i.e. its length is zero).</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>
**MC_SRH_Get_DATE_Data**

Returns a DATE node data.

**Synopsis**

```c
MC_STATUS MC_SRH_Get_DATE_Data (  
    int AsrNodeID,  
    char Adate[],  
    int AdateSize
)
```

- **AsrNodeID** An Identifier of the DATE SR item.
- **Adate** Returns A text value stored in this node.
- **AdateSize** Date buffer size

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_EMPTY_VALUE</td>
<td>The attribute has no value assigned to it.</td>
</tr>
<tr>
<td>MC_NULL_VALUE</td>
<td>The attribute’s value is NULL (i.e. its length is zero).</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**MC_SRH_Get_TIME_Data**

Returns a TIME node data.

**Synopsis**

```c
MC_STATUS MC_SRH_Get_TIME_Data (  
    int AsrNodeID,  
    char Atime[],  
    int AtimeSize
)
```

- **AsrNodeID** An Identifier of the TIME SR item.
- **Atime** Returns a text value stored in this node.
- **AtimeSize** Time buffer size
## Return Value

One of the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_EMPTY_VALUE</td>
<td>The attribute has no value assigned to it.</td>
</tr>
<tr>
<td>MC_NULL_VALUE</td>
<td>The attribute’s value is NULL (i.e. its length is zero).</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

## MC_SRH_Get_UIDREF_Data

Returns a UIDREF node data.

### Synopsis

```c
MC_STATUS MC_SRH_Get_UIDREF_Data (  
    int AsrNodeID,  
    char AuidRef[],  
    int AuidRefSize  
)
```

- **AsrNodeID**  
  An Identifier of the UIDREF SR item.

- **AuidRef**  
  Returns a uid value stored in this node.

- **AuidRefSize**  
  uidRef buffer size

### Return Value

One of the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_EMPTY_VALUE</td>
<td>The attribute has no value assigned to it.</td>
</tr>
<tr>
<td>MC_NULL_VALUE</td>
<td>The attribute’s value is NULL (i.e. its length is zero).</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>
MC_SRH_Get_PNAME_Data

Returns a PNAME node data.

Synopsis

MC_STATUS MC_SRH_Get_PNAME_Data (  
  int AsrNodeID,  
  char *ApersonName[],  
  int ApersonNameSize  
)

AsrNodeID An Identifier of the PNAME SR item.
ApersonName Returns a person name value stored in this node.
ApersonNameSize ApersonName buffer size

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_EMPTY_VALUE</td>
<td>The attribute has no value assigned to it.</td>
</tr>
<tr>
<td>MC_NULL_VALUE</td>
<td>The attribute’s value is NULL (i.e. its length is zero).</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

MC_SRH_Get_SCOORD_First_Data

Returns an SCOORD node data.

Synopsis

MC_STATUS MC_SRH_Get_SCOORD_First_Data (  
  int AsrNodeID,  
  float *Acolumn,  
  float *Arow,  
  int *AvalueCount,  
  SR_GRAPHIC_TYPE *AgraphicType  
)

AsrNodeID An Identifier of the SCOORD SR item.
**Acolumn, Arow**

Spatial coordinates of a geometric region of interest in the DICOM image coordinate system. The IMAGE Content Item from which spatial coordinates are selected is denoted by a SELECTED FROM relationship. Data shall be provided as an ordered set of (column, row) pairs.

**AvalueCount**

Returns number of value pairs stored.

This attribute defines the type of geometry of the annotated region of interest. The following Enumerated Values are specified for image spatial coordinate geometries:

- `SR_GRT_POINT` = a single pixel denoted by a single (column, row) pair
- `SR_GRT_MULTIPOINT` = multiple pixels each denoted by an (column, row) pair
- `SR_GRT_POLYLINE` = a series of connected line segments with ordered vertices denoted by (column, row) pairs; if the first and last vertices are the same it is a closed polygon
- `SR_GRT_CIRCLE` = a circle defined by two (column, row) pairs. The first point is the central pixel. The second point is a pixel on the perimeter of the circle.
- `SR_GRT_ELLIPSE` = an ellipse defined by four pixel (column, row) pairs, the first two points specifying the endpoints of the major axis and the second two points specifying the endpoints of the minor axis of an ellipse.

**Return Value**

One of the enumerated `MC_STATUS` codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_EMPTY_VALUE</td>
<td>The attribute has no value assigned to it.</td>
</tr>
<tr>
<td>MC_NULL_VALUE</td>
<td>The attribute’s value is NULL (i.e. its length is zero).</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**MC_SRH_Get_SCOORD_Next_Data**

Returns an SCOORD node data.

**Synopsis**

```c
MC_STATUS MC_SRH_Get_SCOORD_Next_Data ( int AsrNodeID, ```
float *Acolumn,
float *Arow
)

AsrNodeID  An Identifier of the SCOORD SR item.

Acolumn, Arow  Spatial coordinates of a geometric region of interest in the
DICOM image coordinate system. The IMAGE Content Item
from which spatial coordinates are selected is denoted by a
SELECTED FROM relationship. Data shall be provided as an
ordered set of (column, row) pairs.

Return Value
One of the enumerated MC_STATUS codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_INCOMPATIBLE_VR</td>
<td>The attribute’s value representation cannot be derived from Value. See the table below.</td>
</tr>
<tr>
<td>MC_VALUE_OUT_OF_RANGE</td>
<td>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).</td>
</tr>
<tr>
<td>MC_TEMP_FILE_ERROR</td>
<td>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.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

MC_SRH_Get_TCOORD_D_First_Data

Returns a TCOORD node’s data that were provided as a Referenced DateTime.

Synopsis

MC_STATUS MC_SRH_Get_TCOORD_D_First_Data (  
    int AsrNodeID,
    char *dateTimeOffset[],
    int *dateTimeOffsetSize,
    int *AvalueCount,
    SR_TRANGE_TYPE *AtempRangeType
)
AsrNodeID  An Identifier of the TCOORD SR item.

AdatimeOffset  Specifies temporal point for reference by absolute time. Data is formatted as DICOM DT string.

AdatimeOffsetSize  Specifies temporal point buffer size.

AvalueCount  Returns number of values stored.

AtempRangeType  Defines the type of temporal extent of the region of interest. A temporal point (or instant of time) may be defined by a waveform sample offset (for a single waveform multiplex group only), time offset, or absolute time. Following enumerated values shall be used:

SR_TRT_POINT a single temporal point
SR_TRT_MULTIPOINT multiple temporal points
SR_TRT_SEGMENT a range between two temporal points
SR_TRT_MULTISEGMENT multiple segments, each denoted by two temporal points
SR_TRT_BEGIN a range beginning at one temporal point, and extending beyond the end of the acquired data
SR_TRT_END a range beginning before the start of the acquired data, and extending to (and including) the identified temporal point

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_EMPTY_VALUE</td>
<td>The attribute has no value assigned to it.</td>
</tr>
<tr>
<td>MC_NULL_VALUE</td>
<td>The attribute’s value is NULL (i.e. its length is zero).</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>
**MC_SRH_Get_TCOORD_D_Next_Data**

Returns a TCOORD node’s data that were provided as a Referenced DateTime.

**Synopsis**

```c
MC_STATUS MC_SRH_Get_TCOORD_D_Next_Data(
    int AsrNodeID,
    char AdateTimeOffset[],
    int AdateTimeOffsetSize
)
```

- **AsrNodeID**: An Identifier of the TCOORD SR item.
- **AdateTimeOffset**: Specifies temporal points for reference by absolute time. Data is formatted as DICOM DT string.
- **AdateTimeOffsetSize**: Specifies temporal points for buffer size. Note that it is a multi-value string that can long.
- **AvalueCount**: Returns number of values stored.
- **AtempRangeType**: Defines the type of temporal extent of the region of interest. A temporal point (or instant of time) may be defined by a waveform sample offset (for a single waveform multiplex group only), time offset, or absolute time. Following enumerated values shall be used:
  - SR_TRT_POINT a single temporal point
  - SR_TRT_MULTIPOINT multiple temporal points
  - SR_TRT_SEGMENT a range between two temporal points
  - SR_TRT_MULTISEGMENT multiple segments, each denoted by two temporal points
  - SR_TRT_BEGIN a range beginning at one temporal point, and extending beyond the end of the acquired data
  - SR_TRT_END a range beginning before the start of the acquired data, and extending to (and including) the identified temporal point

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
</tbody>
</table>
MC_NULL_POINTER_PARM
One or more of the pointer-type parameters was NULL.

MC_EMPTY_VALUE
The attribute has no value assigned to it.

MC_NULL_VALUE
The attribute’s value is NULL (i.e. its length is zero).

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

**MC_SRH_Get_TCOORD_O_First_Data**

Returns a TCOORD node’s data that were provided as Referenced Time Offsets.

**Synopsis**

```c
MC_STATUS MC_SRH_Get_TCOORD_O_First_Data (  
    int AsrNodeID,  
    char AtimeOffset[],  
    int AtimeOffsetSize,  
    int *AvalueCount,  
    SR_TRANGE_TYPE *AtempRangeType  
)
```

**AsrNodeID**
An Identifier of the TCOORD SR item.

**AtimeOffset**
Specifies temporal points for reference by number of seconds after start of data.

**AdateTimeOffsetSize**
Specifies temporal point buffer size.

**AvalueCount**
Returns number of values stored.

**AtempRangeType**
Defines the type of temporal extent of the region of interest. A temporal point (or instant of time) may be defined by a waveform sample offset (for a single waveform multiplex group only), time offset, or absolute time. Following enumerated values shall be used:

- **SR_TRT_POINT** a single temporal point
- **SR_TRT_MULTIPOINT** multiple temporal points
- **SR_TRT_SEGMENT** a range between two temporal points
- **SR_TRT_MULTISEGMENT** multiple segments, each denoted by two temporal points
- **SR_TRT_BEGIN** a range beginning at one temporal point, and extending beyond the end of the acquired data
- **SR_TRT_END** a range beginning before the start of the acquired data, and extending to (and including) the identified temporal point
Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_EMPTY_VALUE</td>
<td>The attribute has no value assigned to it.</td>
</tr>
<tr>
<td>MC_NULL_VALUE</td>
<td>The attribute’s value is NULL (i.e. its length is zero).</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

MC_SRH_Get_TCOORD_O_Next_Data

Returns a TCOORD node’s data that were provided as Referenced Time Offsets.

Synopsis

```c
MC_STATUS MC_SRH_Get_TCOORD_O_Next_Data (
    int AsrNodeID,
    char AtimeOffset[],
    int AtimeOffsetSize
)
```

AsrNodeID An Identifier of the TCOORD SR item.

AtimeOffset Specifies temporal points for reference by number of seconds after start of data.

AdateTimeOffsetSize Specifies temporal point buffer size.

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_INCOMPATIBLE_VR</td>
<td>The attribute’s value representation cannot be derived from Value. See the table below.</td>
</tr>
</tbody>
</table>
**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_SYSTEM_ERROR**
An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.

---

**MC_SRH_Get_TCOORD_R_First_Data**

Returns a TCOORD node’s data that were provided as Referenced Sample Positions.

### Synopsis

```c
MC_STATUS MC_SRH_Get_TCOORD_R_First_Data(
    int AsrNodeID,
    unsigned long *ArefSamplePosition,
    int *AvalueCount,
    SR_TRANGE_TYPE *AtempRangeType
)
```

- **AsrNodeID**
  An Identifier of the TCOORD SR item.

- **ArefSamplePosition**
  List of samples within a multiplex group specifying temporal points of the referenced data. Position of first sample is 1. Only the first value returned here.

- **AvalueCount**
  Returns number of values stored.

- **AtempRangeType**
  Defines the type of temporal extent of the region of interest. A temporal point (or instant of time) may be defined by a waveform sample offset (for a single waveform multiplex group only), time offset, or absolute time. Following enumerated values shall be used:
  - SR_TRT_POINT a single temporal point
  - SR_TRT_MULTIPOINT multiple temporal points
  - SR_TRT_SEGMENT a range between two temporal points
  - SR_TRT_MULTISEGMENT multiple segments, each denoted by two temporal points
  - SR_TRT_BEGIN a range beginning at one temporal point, and extending beyond the end of the acquired data
SR_TRT_END a range beginning before the start of the acquired data, and extending to (and including) the identified temporal point

### Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_EMPTY_VALUE</td>
<td>The attribute has no value assigned to it.</td>
</tr>
<tr>
<td>MC_NULL_VALUE</td>
<td>The attribute’s value is NULL (i.e. its length is zero).</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

### MC_SRH_Get_TCOORD_R_Next_Data

Returns a TCOORD node’s data that were provided as Referenced Sample Positions.

#### Synopsis

```c
MC_STATUS MC_SRH_Get_TCOORD_R_Next_Data (  
    int AsrNodeID,  
    unsigned long *ArefSamplePosition
)
```

- **AsrNodeID**: An Identifier of the TCOORD SR item.
- **ArefSamplePosition**: List of samples within a multiplex group specifying temporal points of the referenced data. Position of the first sample is 1. Only the first value is returned here.

#### Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_EMPTY_VALUE</td>
<td>The attribute has no value assigned to it.</td>
</tr>
<tr>
<td>MC_NULL_VALUE</td>
<td>The attribute’s value is NULL (i.e. its length is zero).</td>
</tr>
</tbody>
</table>
**MC_SYSTEM_ERROR**

An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.

---

**MC_SRH_Get_COMPOSITE_Data**

Returns a COMPOSITE node data.

**Synopsis**

```c
MC_STATUS MC_SRH_Get_COMPOSITE_Data(
    int AsrNodeID,
    char AsopClassUid[],
    int AsopClassUidSize,
    char AsopInstanceUid[],
    int AsopInstanceUidSize
)
```

- **AsrNodeID** An Identifier of the COMPOSITE SR item.
- **AsopClassUid** A SOP Class UID of a Composite object which is not an Image or Waveform.
- **AsopClassUidSize** The size of the SOP Class UID buffer.
- **AsopInstanceUid** A SOP Instance UID of a Composite object which is not an Image or Waveform.
- **AsopInstanceUidSize** The size of the SOP Instance UID buffer.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_NORMAL_COMPLETION</strong></td>
<td>The function completed normally.</td>
</tr>
<tr>
<td><strong>MC_INVALID_SR_ID</strong></td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td><strong>MC_NULL_POINTER_PARM</strong></td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td><strong>MC_EMPTY_VALUE</strong></td>
<td>The attribute has no value assigned to it.</td>
</tr>
<tr>
<td><strong>MC_NULL_VALUE</strong></td>
<td>The attribute’s value is NULL (i.e. its length is zero).</td>
</tr>
<tr>
<td><strong>MC_SYSTEM_ERROR</strong></td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

---

**MC_SRH_Get_IMAGE_Data**

Returns an IMAGE node data.
Synopsis

MC_STATUS MC_SRH_Get_IMAGE_Data
(int AsrNodeID,
 char AsopClassUid[],
 int AsopClassUidSize,
 char AsopInstanceUid[],
 int AsopInstanceUidSize,
 int *ArefFramesCount)

AsrNodeID An Identifier of the IMAGE SR item.
AsopClassUid An SOP Class UID of a Composite object which is an Image.
AsopClassUidSize The size of the SOP Class UID buffer.
AsopInstanceUid An SOP Instance UID of a Composite object which is an Image.
AsopInstanceUidSize The size of the SOP Instance UID buffer.
ArefFramesCount Returns a count of references frames within the Referenced SOP Instance to which the reference applies. Frames array can be retrieved via MC_SRH_Get_IMAGE_Frames.

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_EMPTY_VALUE</td>
<td>The attribute has no value assigned to it.</td>
</tr>
<tr>
<td>MC_NULL_VALUE</td>
<td>The attribute's value is NULL (i.e. its length is zero).</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

MC_SRH_Get_IMAGE_Frames

Gets frame numbers within a Referenced SOP Instance to which the reference applies.

Synopsis

MC_STATUS MC_SRH_Get_IMAGE_Frames
(int AsrNodeID,
 long AreferencedFrames[]),
long AframesArraySize
)

AsrNodeID  An Identifier of the IMAGE SR item.

Identifies the frame numbers within the Referenced
AreferencedFrames  SOP Instance to which the reference applies. The first
frame shall be denoted as frame number 1.

AframesArraySize  Size of the AreferencedFrames array.

Return Value
One of the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_INCOMPATIBLE_VR</td>
<td>The attribute’s value representation cannot be derived from Value. See the table below.</td>
</tr>
<tr>
<td>MC_VALUE_OUT_OF_RANGE</td>
<td>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).</td>
</tr>
<tr>
<td>MC_TEMP_FILE_ERROR</td>
<td>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.</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

MC_SRH_Get_WAVEFORM_Data

Returns an IMAGE node data.

Synopsis

MC_STATUS MC_SRH_Get_WAVEFORM_Data (  
   int AsrNodeID,  
   char AsopClassUid[],  
   int AsopClassUidSize,  
   char AsopInstanceUid[],  
   int AsopInstanceUidSize,  
   int *ArefChannelsCount
)

AsrNodeID  An Identifier of the WAVEFORM SR item.
AsopClassUid  
An SOP Class UID of a Composite object which is a Waveform.

AsopClassUidSize  
Size of the SOP Class UID buffer.

AsopInstanceUid  
An SOP Instance UID of a Composite object which is a Waveform.

AsopInstanceUidSize  
Size of the SOP Instance UID buffer.

ArefChannelsCount  
A value count for the channels referenced. Channels array can be retrieved via MC_SRH_Get_WAVEFORM_Channels.

**Return Value**
One of the enumerated MC_STATUS codes defined in “mcstatus.h”:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_EMPTY_VALUE</td>
<td>The attribute has no value assigned to it.</td>
</tr>
<tr>
<td>MC_NULL_VALUE</td>
<td>The attribute’s value is NULL (i.e. its length is zero).</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

**MC_SRH_Get_WAVEFORM_Channels**

Gets channels referenced within the Referenced SOP Instance to which the reference applies.

**Synopsis**

```c
MC_STATUS MC_SRH_Get_WAVEFORM_Channels (  
    int AsrNodeID,  
    short ArefChannels[],  
    long ArefChannelsSize
)
```

AsrNodeID  
An Identifier of the WAVEFORM SR item.
ArefChannels
A multi-value attribute which lists the channels referenced. Each channel is specified as a pair of values (M,C), where the first value is the sequence item number of the Waveform Sequence (5400,0100) attribute in the referenced object (i.e., the Multiplex Group Number), and the second value is the sequence item number of the Channel Definition Sequence (003A, 0200) attribute (i.e., the Channel Number) within the multiplex group.

ArefChannelsSize
Size of the ArefChannels array that shall be enough to fit all data.

Return Value
One of the enumerated MC_STATUS codes defined in “mcstatus.h”

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_EMPTY_VALUE</td>
<td>The attribute has no value assigned to it.</td>
</tr>
<tr>
<td>MC_NULL_VALUE</td>
<td>The attribute’s value is NULL (i.e. its length is zero).</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

MC_SRH_Get_TABLE_First_Cell_Values_Counts
Retrieve the value counts of the internal attributes of the first item in the Cell Values Sequence in the TABLE Content Item. These value counts will be necessary for allocating the memory space for the subsequent retrieval of those attributes.

Synopsis
MC_STATUS MC_SRH_Get_TABLE_First_Cell_Values_Counts (  
    int AsrNodeID,  
    int *AvalueCount,  
    int *ArefContentItemIdCount,  
    int *AselectorAttrValueCount,  
    unsigned long *AselectorAttrTag,  
    int *AcellValuesID  
)  

AsrNodeID
An Identifier of the TABLE content item.

AvalueCount
Returning the number of items in the Cell Values Sequence
ArefContentItemIdCount
Returning the number of values in the Referenced Content Item Identifier.

AselectorAttrValueCount
Returning the number of values in the Selector XX Value data element denoted by the Selector Attribute VR.

AselectorAttrTag
Returning the Selector XX Value data element denoted by the Selector Attribute VR.

AcellValuesID
Returning the numeric handle of the first item in the Cell Value Sequence.

Return Value
One of the enumerated MC_STATUS codes defined in "mcstatus.h":

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_EMPTY_VALUE</td>
<td>The attribute has no value assigned to it.</td>
</tr>
<tr>
<td>MC_NULL_VALUE</td>
<td>The attribute’s value is NULL (i.e. its length is zero).</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

MC_SRH_Get_TABLE_First_Column_Definition
Retrieve the data of the first item of the Table Column Definition Sequence in the TABLE Content Item.

Synopsis
MC_STATUS MC_SRH_Get_TABLE_First_Column_Definition (  
  int AsrNodeID,  
  TABLE_ROW_DEF *AtableColDef,  
  int *AvalueCount  
)

AsrNodeID An Identifier of the TIME SR item.

AtableColDef Pointer to a supplied structure in which the next Table Column Definition Sequence item is to be retrieved.
AvalueCount

Returns the number of items stored in the Table Column Definition Sequence

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_EMPTY_VALUE</td>
<td>The attribute has no value assigned to it.</td>
</tr>
<tr>
<td>MC_NULL_VALUE</td>
<td>The attribute’s value is NULL (i.e. its length is zero).</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

MC_SRH_Get_TABLE_First_Row_Definition

Retrieve the data of the first item of the Table Row Definition Sequence in the TABLE Content Item.

Synopsis

MC_STATUS MC_SRH_Get_TABLE_First_Row_Definition (  
    int AsrNodeID,  
    TABLE_ROW_DEF *AtableRowDef,  
    int *AvalueCount  
)

AsrNodeID

An Identifier of the TIME SR item.

AtableRowDef

Pointer to a supplied structure in which the next Table Row Definition Sequence item is to be retrieved.

AvalueCount

Returns the number of items stored in the Table Row Definition Sequence

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
</tbody>
</table>
MC_NULL_POINTER_PARM  One or more of the pointer-type parameters was NULL.
MC_EMPTY_VALUE The attribute has no value assigned to it.
MC_NULL_VALUE The attribute’s value is NULL (i.e. its length is zero).
MC_SYSTEM_ERROR An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.

**MC_SRH_Get_TABLE_Next_Cell_Values_Counts**

Retrieve the value counts of the internal attributes of the next Cell Values Sequence item in the TABLE Content Item. These value counts will be necessary for allocating the memory space for the subsequent retrieval of those attributes.

**Synopsis**

```c
MC_STATUS MC_SRH_Get_TABLE_Next_Cell_Values_Counts (  
    int AsrNodeID,  
    int *ArefContentItemIdCount,  
    int *AselectorAttrValueCount,  
    unsigned long *AselectorAttrTag,  
    int *AcellValuesID  
)
```

- **AsrNodeID** An Identifier of the TABLE content item.
- **ArefContentItemIdCount** Returning the number of values in the Referenced Content Item Identifier.
- **AselectorAttrValueCount** Returning the number of values in the Selector XX Value data element denoted by the Selector Attribute VR.
- **AselectorAttrTag** Returning the Selector XX Value data element denoted by the Selector Attribute VR.
- **AcellValuesID** Returning the numeric handle of the first item in the Cell Value Sequence.

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_EMPTY_VALUE</td>
<td>The attribute has no value assigned to it.</td>
</tr>
</tbody>
</table>
MC_NULL_VALUE
The attribute's value is NULL (i.e. its length is zero).

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

**MC_SRH_Get_TABLE_Next_Column_Definition**

Retrieve the data in the next item of the Table Column Definition Sequence in the TABLE Content Item.

**Synopsis**

```c
MC_STATUS MC_SRH_Get_TABLE_Next_Column_Definition (  
    int AsrNodeID,  
    TABLE_COL_DEF *AtableColDef  
)
```

- **AsrNodeID** An Identifier of the TABLE content item.
- **AtableColDef** Pointer to a supplied structure in which the next Table Column Definition Sequence item is to be retrieved.
- **AtimeSize** Time buffer size

**Return Value**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_EMPTY_VALUE</td>
<td>The attribute has no value assigned to it.</td>
</tr>
<tr>
<td>MC_NULL_VALUE</td>
<td>The attribute's value is NULL (i.e. its length is zero).</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>
MC_SRH_Get_TABLE_Next_Row_Definition

Retrieve the data in the next item of the Table Row Definition Sequence in the TABLE Content Item.

Synopsis

```c
MC_STATUS MC_SRH_Get_TABLE_Next_Row_Definition (
    int AsrNodeID,
    TABLE_ROW_DEF *AtableRowDef
)
```

- **AsrNodeID**: An Identifier of the TABLE content item.
- **AtableRowDef**: Pointer to a supplied structure in which the next Table Row Definition Sequence item is to be retrieved.

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_EMPTY_VALUE</td>
<td>The attribute has no value assigned to it.</td>
</tr>
<tr>
<td>MC_NULL_VALUE</td>
<td>The attribute’s value is NULL (i.e. its length is zero).</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

MC_SRH_Get_TABLE_Cell_Values_Item

Retrieve the data in a Cell Value Sequence item in a TABLE Content item.

Synopsis

```c
MC_STATUS MC_SRH_Get_TABLE_Cell_Values_Item (  
    int AcellValuesID,
    TABLE_CELL_VALUES* AtableCellValues
)
```

- **AcellValuesID**: Numeric handle pointing to the Cell Value Sequence Item to be retrieved.
- **AtableCellValues**: Pointer to a supplied structure in which the attributes of the Cell Value Sequence item are to be retrieved.
Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_EMPTY_VALUE</td>
<td>The attribute has no value assigned to it.</td>
</tr>
<tr>
<td>MC_NULL_VALUE</td>
<td>The attribute’s value is NULL (i.e. its length is zero).</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>

MC_SRH_Get_CONTAINER_Data

Returns a CONTAINER node data.

Synopsis

MC_STATUS MC_SRH_Get_CONTAINER_Data (  
    int AsrNodeID,  
    SR_CONT_CONTINUITY *Acontinuity,  
    char AtemplateId[],  
    int AtemplateIdSize  
)

AsrNodeID
An Identifier of the TIME SR item.

Acontinuity
Enumerated return value that specifies whether or not its contained Content Items are logically linked in a continuous textual flow, or are separate items.

AtemplateId
DICOM Template Identifier that describes the content of this Content Item and its subsidiary Content Items. Can be NULL.

AtemplateIdSize
The size in bytes of the template Identifier array.

Return Value

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_NORMAL_COMPLETION</td>
<td>The function completed normally.</td>
</tr>
<tr>
<td>MC_INVALID_SR_ID</td>
<td>The AsrNodeID value is not a valid SR tree management object ID.</td>
</tr>
<tr>
<td>ErrorCode</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MC_NULL_POINTER_PARM</td>
<td>One or more of the pointer-type parameters was NULL.</td>
</tr>
<tr>
<td>MC_INCOMPATIBLE_VALUE</td>
<td>The Acontinuity value is not valid.</td>
</tr>
<tr>
<td>MC_EMPTY_VALUE</td>
<td>The attribute has no value assigned to it.</td>
</tr>
<tr>
<td>MC_NULL_VALUE</td>
<td>The attribute’s value is NULL (i.e. its length is zero).</td>
</tr>
<tr>
<td>MC_SYSTEM_ERROR</td>
<td>An unexpected, potentially serious, problem was detected in the operating environment. A message describing the error has been written to the Merge DICOM Toolkit log file.</td>
</tr>
</tbody>
</table>
Appendix A: Writing a DICOM Conformance Statement

Detailed below is a guideline for writing a conformance statement for your application. Since the Toolkit is not an application, this section 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 Toolkit. 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 toolkit functionality.

Conformance Statement Sections

Implementation Model

The Implementation model consists of three sections:
- the Application Data Flow Diagram which specifies the relationship between the Application Entities and the “external world” or Real-World activities
- a functional description of each Application Entity
- the sequencing constraints among them.

Application Data Flow

As part of the Implementation model, an Application Data Flow Diagram shall be included. This diagram represents all of the Application Entities present in an implementation, and graphically depicts the relationship of the AE's use of DICOM to Real-World Activities as well as any applicable User interaction.

The Merge DICOM Toolkit 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.

![Data Flow Diagram](image-url)
Functional Definition of Application Entities (AE)

This part shall contain a functional definition for each individual, local Application Entity. It will describe in general terms the functions to be performed by the AE, and the DICOM services used to accomplish these functions. In this sense, “DICOM services” refers not only to DICOM Service Classes, but also to lower level DICOM services, such as Association Services.

Application conformance statements will fill in this section with a general specification of functions to be performed by SCU or SCP.

Sequencing of Real World Activities

If applicable, this section shall contain a description of sequencing as well as potential constraints on real-world activities. These include any applicable user interaction as performed by all the AEs. A UML sequence diagram that depicts the real-world activities as vertical bars, and shows events exchanged between them as arrows, is strongly recommended.

Application conformance statements fill this section in 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

The next section in the DICOM Conformance Statement is a set of Application Entity specifications. There shall be one specification for AE. Each individual AE specification has a subsection. There are as many of these subsections as there are different AE’s in the implementation. That is, if there are two distinct AEs, then there will be two subsections. The Merge DICOM Toolkit is using the mergecom.app configuration file to read configuration parameters for each AE. The following subsections are filled in for each AE:

- Application Entity
  - SOP Classes
  - Association Policies
    - General
    - Number of Associations
    - Asynchronous Nature
    - Implementation Identifying Information
  - Association Initiation Policy
    - Activity
• Description and Sequencing of Activities
• Proposed Presentation Contexts
• SOP Specific Conformance for SOP Class(es)

• Association Acceptance Policy
  o Activity
    • Description and sequencing of Activities
    • Accepted Presentation Contexts
    • SOP Specific Conformance for SOP Class(es)

**SOP Classes**

Application conformance statements specify the DICOM SOPs which are supported by each Application Entity. For SCP Entities, the initiation of associations. Please see the “System Profile” section in ANNEX B: CONFIGURATION PARAMETERS and the “MC_Wait_For_Association” or “MC_Wait_For_Secure_Association” definition in this Reference Manual. For SCU Entities, 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.

**Number of Associations**

The Merge DICOM Toolkit does not impose any limit on the number of simultaneous associations that can be requested or accepted. The only limitation on the number of simultaneous associations is imposed by the operating system and available resources. However, if your application enforces that limit, it is defined here.

The MAX_PENDING_CONNECTIONS setting in the “mergecom.pro” file refers to the maximum number of outstanding connection requests per listener socket. It does not limit the maximum number of simultaneous associations.

**Asynchronous Nature**

Merge DICOM Toolkit 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, as well as the Implementation version name. These identifiers are taken from the mergecom.pro configuration file under the following keys:

  IMPLEMENTATION_CLASS_UID
  IMPLEMENTATION_VERSION

This UID must follow the syntax rules specified in Part 5 of the DICOM standard.
**Proposed or Accepted Presentation Contexts**

Application conformance statements specify all presentation contexts that are 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 that are specified in the "mergecom.pro" file. Please see the "System Profile" section in Appendix B: Configuration Parameters.
- 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 "MC_Get_Negotiation_Info" function in this Reference Manual.
- any rules that govern the acceptance of presentation contexts for the AE. This includes rules for which combinations of Abstract/Transfer Syntaxes are acceptable, and rules for prioritization of presentation contexts. Rules that govern selection of transfer syntax within a presentation context shall be stated here. Please see the "Application Profile" section in the Appendix B: Configuration Parameters. Also, see the "MC_Get_Association_Info" function in this Reference Manual to learn about the presentation contexts that are queryable by an application program.

Refer to Table A-1 for an example.

*Table A-1: Example Presentation Context*

<table>
<thead>
<tr>
<th>Name</th>
<th>Abstract Syntax</th>
<th>Transfer Syntax</th>
<th>Role</th>
<th>Extended Negotiation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UID</td>
<td>Name List</td>
<td>UID List</td>
<td></td>
</tr>
<tr>
<td>Computed Radiography</td>
<td>1.2.840.10008.5.1.4.1.1.1</td>
<td>DICOM Implicit VR Little Endian</td>
<td>1.2.840.10008.1.2</td>
<td>SCP</td>
</tr>
<tr>
<td>Image Storage</td>
<td></td>
<td></td>
<td>1.2.840.10008.1.2.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DICOM Explicit VR Little Endian</td>
<td></td>
<td>SCP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DICOM Explicit VR Big Endian</td>
<td></td>
<td>SCP</td>
</tr>
</tbody>
</table>

Merge DICOM Toolkit uses mergecom.app configuration settings to specify presentation contexts shown above.

**SOP Specific Conformance**

This section includes the SOP specific behavior, i.e. error codes, error and exception handling and time-outs, etc. The information is described in the SOP specific Conformance Statement section of PS 3.4 (or relevant private SOP definition).

**Transfer Syntax Selection Policies**

Merge DICOM Toolkit uses the following policy when selecting transfer syntax:

- An SCU offers any transfer syntaxes which are defined in its mergecom.pro file.
- The SCP prefers its native byte ordering, and will prefer explicit over implicit VR.
Network Interfaces

Physical Network Interface
Merge DICOM Toolkit runs over the TCP/IP protocol stack on any physical interconnection media supporting the TCP/IP stack.

IPv4 and IPv6 Support
Merge DICOM Toolkit supports both IPv4 and IPv6 protocols and is configurable in the system profile.

Configuration
Refer to the Appendix B: Configuration Parameters 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.
- mergeserv.pro — Specifies run-time parameters for the application
- mergeserv.app — Defines service lists and applications on other network nodes to which connections are possible.
- mergesrv.srv — Service and sequence definitions.

AE Title/Presentation Address Mapping
Presentation address mapping is configured in the mergeserv.app file. The Presentation Address of an SCU/SCP application is specified by configuring the Listen Port in the mergeserv.pro file, and specifying the AE title for the SCU/SCP within the application itself.

Configurable Parameters
The mergeserv.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 mergeserv.pro file. Also, before modifying other parameters, you should consider contacting Merge Healthcare for advice.

PDU size
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 this Reference Manual for further information about configuring the PDU size.

Extensions/Specializations/Privatizations
Standard Extended/Specialized/Private SOPs
Application conformance statements list extended, specialized, or private SOPs that are supported.
Private Transfer Syntaxes
This section describes private transfer syntaxes that are listed in the Transfer Syntax Tables. See the System Profile section in Appendix B: Configuration Parameters for details.
Appendix B: Configuration Parameters

This appendix describes each configuration parameter in detail.

Merge DICOM Toolkit is a highly configurable toolkit, and understanding its configuration is critical to using the library effectively.

Related parameters are grouped into sections in a configuration file as follows:

```
[SECTION_1]
PARAMETER_1 = value1
PARAMETER_2 = value2

[SECTION_2]
PARAMETER_3 = value3
```

Information contained in these tables are parameter names including descriptions and sections where they are 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 “Configuration” section in this Reference Manual.

Initialization File

The Merge DICOM Toolkit Initialization file (usually called merge.ini) provides the DICOM Toolkit with its top-level configuration. It specifies the location of the other three configuration files, along with message and error logging characteristics.

There are two ways of accessing the merge.ini file in your runtime environment. The functions `MC_Set_MergeINI()` or `MC_Set_MergeINI_Unicode()` can be used to assign the path where the merge.ini file is located. You can also set the `MERGE_INI` environment variable to point to the Merge Initialization File. This variable can be set within a command shell. For example:

In Unix C-shell:
```
setenv MERGE_INI /users/mc3adv/merge.ini
```

In Unix Bourne, Korn, or Bash shell:
```
MERGE_INI=/users/mc3adv/merge.ini; export MERGE_INI
```

In DOS command shell:
```
set MERGE_INI=\mc3adv\merge.ini
```

See the Merge DICOM Toolkit Platform Notes for your platform if none of these methods apply.

The initialization file contains one `[MergeCOM3]` section that points to the other three Merge DICOM Toolkit initialization files. It specifies characteristics of the message/error log kept by the DICOM Toolkit library, turns particular types of logging on and off, and specifies where the messages are logged (file, screen, both or none). In most cases the INFO, WARNING and ERROR messages are sufficient. The `Tn_MESSAGE` settings (where `n` is an integer between 1 and 9) turns on lower-level protocol tracing capabilities. These capabilities can prove useful when running into...
difficulties communicating with other implementations of DICOM over a network and can be used by Merge Healthcare service engineers in diagnosing lower-level network problems.

The following parameters are recognized by Merge DICOM Toolkit in the initialization file.

**Table B-1: Initialization file parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALLOW_LIBRARY_EXCEPTION_HANDLER</td>
<td>MergeCOM3</td>
<td>This parameter, if set to YES, directs the toolkit to call the internal (and the user defined, if present) exception handler in the case System exception or System signal is raised. This allows for the (limited) recovery of the user program by at least giving it a chance to cleanup and exit gracefully and avoid a straight crash. DEFAULT: NO</td>
</tr>
<tr>
<td>BLANK_FILL_LOG_FILE</td>
<td>MergeCOM3</td>
<td>This parameter informs the toolkit 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</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>MergeCOM3</td>
<td>This parameter instructs the toolkit to which destination (File, Screen and/or Memory) to log error messages.</td>
</tr>
<tr>
<td>INFO_MESSAGE</td>
<td>MergeCOM3</td>
<td>This parameter instructs the toolkit to which destination (File, Screen and/or Memory or none) to log information messages.</td>
</tr>
<tr>
<td>LOG_FILE</td>
<td>MergeCOM3</td>
<td>This is the name of the Merge DICOM Toolkit message log. The file will be [re-]created by Merge DICOM Toolkit. This parameter is ignored by embedded toolkits. The path to the file can be specified using environment variables (including the pseudo environment variable MC3INIDIR which does not need to be set as the toolkit will resolve it internally to the directory where the merge.ini file resides). Unicode paths can now be specified through the environment variables. Both Windows style (%) and Unix style ($) notations for the environment variables are accepted. DEFAULT: ./merge.log</td>
</tr>
<tr>
<td>LOG_FILE_BACKUP</td>
<td>MergeCOM3</td>
<td>This parameter tells the Merge DICOM Toolkit to create a backup of the log file before starting a new log. If &quot;ON&quot;, any existing log file is renamed with a file extension of .Lnn where nn is an integer number between 01 and 99. DEFAULT: OFF.</td>
</tr>
<tr>
<td>Name</td>
<td>Section</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------</td>
<td>------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LOG_FILE_LINE_LENGTH</td>
<td>MergeCOM3</td>
<td>This option specifies the number of characters that occur on a line within the merge.log file.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DEFAULT: 78</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MINIMUM: 16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MAXIMUM: 254</td>
</tr>
<tr>
<td>LOG_FILE_SIZE</td>
<td>MergeCOM3</td>
<td>This is the number of lines which will be created for the log file. If BLANK_FILL_LOG_FILE is set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>to YES, the file is initialized to all binary zeros before the first message is logged.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DEFAULT: 100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MINIMUM: 100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MAXIMUM: 30720 (30 * 1024)</td>
</tr>
<tr>
<td>LOG_MEMORY_SIZE</td>
<td>MergeCOM3</td>
<td>This is the number of lines of length equal to LOG_FILE_LINE_LENGTH which will be created for</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the memory log. Used when logging is set to “Memory” in merge.ini.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DEFAULT: 1</td>
</tr>
<tr>
<td>MERGECOM_3_APPLICATIONS</td>
<td>MergeCOM3</td>
<td>File containing the Merge DICOM Toolkit application configurations.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The path to the file can be specified using environment variables (including the pseudo</td>
</tr>
<tr>
<td></td>
<td></td>
<td>environment variable MC3INIDIR which does not need to be set as the toolkit will resolve it</td>
</tr>
<tr>
<td></td>
<td></td>
<td>internally to the directory where the merge.ini file resides).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unicode paths can now be specified through the environment variables. Both Windows style (%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and Unix style ($) notations for the environment variables are accepted.</td>
</tr>
<tr>
<td>MERGECOM_3_PROFILE</td>
<td>MergeCOM3</td>
<td>File containing the Merge DICOM Toolkit system profile parameters.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The path to the file can be specified using environment variables (including the pseudo</td>
</tr>
<tr>
<td></td>
<td></td>
<td>environment variable MC3INIDIR which does not need to be set as the toolkit will resolve it</td>
</tr>
<tr>
<td></td>
<td></td>
<td>internally to the directory where the merge.ini file resides).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unicode paths can now be specified through the environment variables. Both Windows style (%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and Unix style ($) notations for the environment variables are accepted.</td>
</tr>
<tr>
<td>Name</td>
<td>Section</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MERGECOM_3_SERVICES</td>
<td>MergeCOM3</td>
<td>File containing the Merge DICOM Toolkit system service and message definitions. The path to the file can be specified using environment variables (including the pseudo environment variable MC3INIDIR which does not need to be set as the toolkit will resolve it internally to the directory where the merge.ini file resides). Unicode paths can now be specified through the environment variables. Both Windows style (%) and Unix style ($) notations for the environment variables are accepted.</td>
</tr>
<tr>
<td>NUM_HISTORICAL_LOG_FILES</td>
<td>MergeCOM3</td>
<td>This parameter informs the toolkit 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.LXX where basename.LXX 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.</td>
</tr>
<tr>
<td>T1_MESSAGE</td>
<td>MergeCOM3</td>
<td>Not used (internal tracing).</td>
</tr>
<tr>
<td>T2_MESSAGE</td>
<td>MergeCOM3</td>
<td>This logging level parameter instructs the toolkit to log the entire contents of messages sent or received over the network. The format is similar to MC_List_Message's output.</td>
</tr>
<tr>
<td>T3_MESSAGE</td>
<td>MergeCOM3</td>
<td>This logging level parameter instructs the toolkit to log messages relating to association negotiation.</td>
</tr>
<tr>
<td>T4_MESSAGE</td>
<td>MergeCOM3</td>
<td>This logging level parameter instructs the toolkit to log messages when incoming associations are automatically rejected.</td>
</tr>
<tr>
<td>T5_MESSAGE</td>
<td>MergeCOM3</td>
<td>This logging level parameter instructs the toolkit to log messages relating to regular and extended validation.</td>
</tr>
<tr>
<td>T6_MESSAGE</td>
<td>MergeCOM3</td>
<td>This logging level parameter instructs the toolkit to log messages relating to configuration.</td>
</tr>
<tr>
<td>T7_MESSAGE</td>
<td>MergeCOM3</td>
<td>This logging level parameter instructs the toolkit to log messages relating to logging of command level attributes in messages sent or received.</td>
</tr>
<tr>
<td>T8_MESSAGE</td>
<td>MergeCOM3</td>
<td>This logging level parameter instructs the toolkit to log messages relating to the streaming in and out of messages and file objects.</td>
</tr>
<tr>
<td>Name</td>
<td>Section</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>T9_MESSAGE</td>
<td>MergeCOM3</td>
<td>This logging level parameter instructs the toolkit 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.</td>
</tr>
<tr>
<td>WARNING_MESSAGE</td>
<td>MergeCOM3</td>
<td>This parameter instructs the toolkit to which destination (File, Screen and/or Memory or none) to log warning messages.</td>
</tr>
</tbody>
</table>

**NOTE:** The destination for the logging from the T2_MESSAGE to T9_MESSAGE trace levels can be one or more of File, Screen and Memory or none.

**Application Profile**

The Merge DICOM Toolkit application profile (usually called mergecom.app) specifies the characteristics of your application entity and the Application Entities (AE) your application connects with over a network. The name and location of this file is specified in the [MergeCOM3] section of the Merge DICOM Toolkit initialization file.

When your application acts as a client (SCU), the application profile must specify the network address of the server (SCP) and the application entities you wish to connect (open an association) to. Your client refers to the Application Entity using a DICOM AE title, which is the same way it is referred to in the application profile. The AE title consists of a string of characters containing no spaces and having a length of 16 characters or less. A section of the profile exists for each Server AE that you wish to connect to.

For example, if your application is an image source and also performs query and retrieval of images from two separate DICOM AE’s, it might contain sections like the following:

```
[Acme_Store_SCP]
  PORT_NUMBER     = 104
  HOST_NAME       = acme_sun1
  SERVICE_LIST    = Storage_Service_List

[Acme_QR_SCP]
  PORT_NUMBER     = 104
  HOST_NAME       = acme_hp2
  SERVICE_LIST    = Query_Service_List
```

Acme_Store_SCP and Acme_QR_SCP are the AE titles for the applications you wish to connect to. The storage server runs on a SUN Solaris computer with the host name acme_sun1, while the query/retrieve server runs on an HP workstation with the host name acme_hp2. Both servers listen on port 104 (the standard DICOM listen port). The host name and port combined make up the TCP/IP network address for a listening server application. See Figure 9.

Besides entering a hostname for the HOST_NAME parameter, it is also possible to simply enter an IP address. Both IPv4 addresses and IPv6 addresses are allowed in this field.
Sections

Table B-2: Application profile section headings

The application profile contains the following sections:

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;remote_application_title&gt;</td>
<td>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.</td>
</tr>
<tr>
<td>&lt;service_list_name&gt;</td>
<td>List(s) of DICOM services that will be provided by the Application Entities listed in the [{&lt;remote_application_title&gt;}] 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.</td>
</tr>
<tr>
<td>&lt;syntax_list_name&gt;</td>
<td>List(s) of DICOM transfer syntaxes that will be supported by the services listed in the [{&lt;service_list_name&gt;}] sections. The transfer syntaxes must be one of those listed in Table 5.</td>
</tr>
</tbody>
</table>

Parameters

Table B-3: Application profile section headers.

The application profile contains the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PORT_NUMBER</td>
<td>&lt;remote_application_title&gt;</td>
<td>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.</td>
</tr>
<tr>
<td>HOST_NAME</td>
<td>&lt;remote_application_title&gt;</td>
<td>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 parameter's value must be 1 to 19 bytes in length with no embedded spaces. Note: A numeric internet address may be used: e.g. 192.204.32.1</td>
</tr>
<tr>
<td>SERVICE_LIST</td>
<td>&lt;remote_application_title&gt;</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------</td>
<td></td>
</tr>
<tr>
<td>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 <code>MC_Open_Association</code> or <code>MC_Open_Secure_Association</code> function call. The parameter’s value names must be 1 to 33 bytes in length with no embedded spaces.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Service List**

The `SERVICE_LIST` section of the application profile is used to describe the DICOM services that will be negotiated by the listed Application Entity.

For example, these sections might look like the following:

```
[Storage_Service_List]
SERVICES_SUPPORTED = 11  # Number of Services in list
SERVICE_1 = STANDARD_MR
SERVICE_2 = STANDARD_CR
SERVICE_3 = STANDARD_CT
SERVICE_4 = STANDARD_CURVE
SERVICE_5 = STANDARD_MODALITY_LUT
SERVICE_6 = STANDARD_OVERLAY
SERVICE_7 = STANDARD_SEC_CAPTURE
SERVICE_8 = STANDARD_US
SERVICE_9 = STANDARD_US_MF
SERVICE_10 = STANDARD_VOI_LUT
SERVICE_11 = STANDARD_NM

[Query_Service_List]
SERVICES_SUPPORTED = 2  # Number of Services in list
SERVICE_1 = STUDY_ROOT_FIND
SERVICE_2 = STUDY_ROOT_MOVE
```

`[Storage_Service_List]` lists the storage services that are requested, while `[Query_Service_List]` lists the type of query/retrieve that are requested. These service names are the strings used in Merge DICOM Toolkit to identify standard DICOM services, and are discussed further in specific application guides. Any services listed must be defined in the service profile, discussed below.
A service list also needs to be defined for each of your own server AE’s. Even though you do not need a section for your server AE title (since it is running on your local machine), you do need to specify a service list that your application supports as an SCP. If your application also acts as a storage server, it might use `Storage_Service_List`. You also need to specify a listen port for your server AE in the system profile, which is discussed below.

The parameter values are text strings recognizable by the Merge DICOM Toolkit. These strings are defined in `mergecom.srv` in the toolkit distribution. They are also made available to the application writer at compile time in the header file `mc3services.h` and source file `mc3services.c` as symbolic constants. The following is a list of currently supported services:

**Table B-4: Application profile parameters**

<table>
<thead>
<tr>
<th>Merge DICOM Toolkit Service Parameter</th>
<th>DICOM Service Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACQUISITION_CONTEXT_SR</td>
<td>Storage</td>
</tr>
<tr>
<td>ADVANCED_BLENDING_PRESENTATION_STATE</td>
<td>Storage</td>
</tr>
<tr>
<td>ARTERIAL_PULSE_WAVEFORM</td>
<td>Storage</td>
</tr>
<tr>
<td>AUDIO_WAVEFORM_REAL_TIME_COMMUNICATION</td>
<td>Storage</td>
</tr>
<tr>
<td>AUTOREFRACTION_MEASUREMENTS</td>
<td>Storage</td>
</tr>
<tr>
<td>BASIC_ANNOTATION_BOX</td>
<td>Print Management</td>
</tr>
<tr>
<td>BASIC_Color_IMAGE_BOX</td>
<td>Print Management</td>
</tr>
<tr>
<td>BASIC_FILM_BOX</td>
<td>Print Management</td>
</tr>
</tbody>
</table>

*Figure B-1: A sample configuration of DICOM applications*
<table>
<thead>
<tr>
<th>Merge DICOM Toolkit Service Parameter</th>
<th>DICOM Service Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASIC_FILM_SESSION</td>
<td>Print Management</td>
</tr>
<tr>
<td>BASIC_GRAYSCALE_IMAGE_BOX</td>
<td>Print Management</td>
</tr>
<tr>
<td>BASIC_PRINT_IMAGE_OVERLAY_BOX</td>
<td>Print Management</td>
</tr>
<tr>
<td>BASIC_STRUCTURED_DISPLAY</td>
<td>Storage</td>
</tr>
<tr>
<td>BODY_POSITION_WAVEFORM</td>
<td>Storage</td>
</tr>
<tr>
<td>BREAST_IMAGING_RPI_QUERY</td>
<td>Relevant Patient Information Query</td>
</tr>
<tr>
<td>BREAST_PROJ_PRESENT</td>
<td>Storage</td>
</tr>
<tr>
<td>BREAST_PROJ_PROCESS</td>
<td>Storage</td>
</tr>
<tr>
<td>BREAST_TOMO_IMAGE_STORAGE</td>
<td>Storage</td>
</tr>
<tr>
<td>C_ARM_PHOTON_ELECTRON_RADIATION</td>
<td>Storage</td>
</tr>
<tr>
<td>C_ARM_PHOTON_ELECTRON_RADIATION_RECORD</td>
<td>Storage</td>
</tr>
<tr>
<td>CARDIAC_RPI_QUERY</td>
<td>Relevant Patient Information Query</td>
</tr>
<tr>
<td>CHEST_CAD_SR</td>
<td>Storage</td>
</tr>
<tr>
<td>COLON_CAD_SR</td>
<td>Storage</td>
</tr>
<tr>
<td>COLOR_PALETTE_FIND</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>COLOR_PALETTE_GET</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>COLOR_PALETTE_MOVE</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>COLOR_PALETTE_STORAGE</td>
<td>Storage</td>
</tr>
<tr>
<td>COMPOSITE_INST_RET_NO_BULK_GET</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>COMPOSITE_INSTANCE_ROOT_RET_GET</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>COMPOSITE_INSTANCE_ROOT_RET_MOVE</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>COMPOSITING_PLANAR_MPR_VOLUMETRIC_PS</td>
<td>Storage</td>
</tr>
<tr>
<td>COMPREHENSIVE_3D_SR</td>
<td>Storage</td>
</tr>
<tr>
<td>CONTENT_ASSESSMENT_RESULTS</td>
<td>Storage</td>
</tr>
<tr>
<td>Merge DICOM Toolkit Service Parameter</td>
<td>DICOM Service Class</td>
</tr>
<tr>
<td>-------------------------------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>CORNEAL_TOPOGRAPHY_MAP</td>
<td>Storage</td>
</tr>
<tr>
<td>CT_DEFINED_PROCEDURE_PROTOCOL</td>
<td>Storage</td>
</tr>
<tr>
<td>CT_PERFORMED_PROCEDURE_PROTOCOL</td>
<td>Storage</td>
</tr>
<tr>
<td>DEFINED_PROCEDURE_PROTOCOL_FIND</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>DEFINED_PROCEDURE_PROTOCOL_GET</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>DEFINED_PROCEDURE_PROTOCOL_MOVE</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>DEFORMABLE_SPATIAL_REGISTRATION</td>
<td>Storage</td>
</tr>
<tr>
<td>DERMOSCOPIC_PHOTOGRAPHY_IMAGE</td>
<td>Storage</td>
</tr>
<tr>
<td>DETACHED_INTERP_MANAGEMENT</td>
<td>Results Management</td>
</tr>
<tr>
<td>DETACHED_PATIENT_MANAGEMENT</td>
<td>Patient Management</td>
</tr>
<tr>
<td>DETACHED_RESULTS_MANAGEMENT</td>
<td>Results Management</td>
</tr>
<tr>
<td>DETACHED_STUDY_MANAGEMENT</td>
<td>Study Management</td>
</tr>
<tr>
<td>DETACHED_VISIT_MANAGEMENT</td>
<td>Patient Management</td>
</tr>
<tr>
<td>DICOMDIR</td>
<td>Media Storage</td>
</tr>
<tr>
<td>DISPLAY_SYSTEM</td>
<td>Display System Management</td>
</tr>
<tr>
<td>ELECTROMYOGRAM_WAVEFORM</td>
<td>Storage</td>
</tr>
<tr>
<td>ELECTROOCULOGRAM_WAVEFORM</td>
<td>Storage</td>
</tr>
<tr>
<td>ENCAPSULATED_CDA</td>
<td>Storage</td>
</tr>
<tr>
<td>ENCAPSULATED_MTL</td>
<td>Storage</td>
</tr>
<tr>
<td>ENCAPSULATED_OBJ</td>
<td>Storage</td>
</tr>
<tr>
<td>ENCAPSULATED_STL</td>
<td>Storage</td>
</tr>
<tr>
<td>ENHANCED_CT_IMAGE</td>
<td>Storage</td>
</tr>
<tr>
<td>ENHANCED_MR_COLOR_IMAGE</td>
<td>Storage</td>
</tr>
<tr>
<td>ENHANCED_MR_IMAGE</td>
<td>Storage</td>
</tr>
<tr>
<td>ENHANCED_PET_IMAGE</td>
<td>Storage</td>
</tr>
<tr>
<td>Merge DICOM Toolkit Service Parameter</td>
<td>DICOM Service Class</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>ENHANCED_US_VOLUME</td>
<td>Storage</td>
</tr>
<tr>
<td>ENHANCED_XA_IMAGE</td>
<td>Storage</td>
</tr>
<tr>
<td>ENHANCED_XRAY_RADIATION_DOSE_SR</td>
<td>Storage</td>
</tr>
<tr>
<td>ENHANCED_XRF_IMAGE</td>
<td>Storage</td>
</tr>
<tr>
<td>EXTENSIBLE_SR</td>
<td>Storage</td>
</tr>
<tr>
<td>G_P_PERFORMED_PROCEDURE_STEP_RETIRED</td>
<td>Study Management</td>
</tr>
<tr>
<td>G_P_SCHEDULED_PROCEDURE_STEP_RETIRED</td>
<td>Study Management</td>
</tr>
<tr>
<td>G_P_WORKLIST_RETIRED</td>
<td>Basic Worklist Management</td>
</tr>
<tr>
<td>GENERAL_AUDIO_WAVEFORM</td>
<td>Storage</td>
</tr>
<tr>
<td>GENERAL_RPI_QUERY</td>
<td>Relevant Patient Information Query</td>
</tr>
<tr>
<td>GENERIC_IMPLANT_TEMPLATE</td>
<td>Storage</td>
</tr>
<tr>
<td>GENERIC_IMPLANT_TEMPLATE_FIND</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>GENERIC_IMPLANT_TEMPLATE_GET</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>GENERIC_IMPLANT_TEMPLATE_MOVE</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>GRAYSCALE_PLANAR_MPR_VOLUMETRIC_PS</td>
<td>Storage</td>
</tr>
<tr>
<td>HANGING_PROTOCOL</td>
<td>Hanging Protocol Storage</td>
</tr>
<tr>
<td>HANGING_PROTOCOL_FIND</td>
<td>Hanging Protocol Query/Retrieve</td>
</tr>
<tr>
<td>HANGING_PROTOCOL_GET</td>
<td>Hanging Protocol Query/Retrieve</td>
</tr>
<tr>
<td>HANGING_PROTOCOL.Move</td>
<td>Hanging Protocol Query/Retrieve</td>
</tr>
<tr>
<td>IMAGE_OVERLAY_BOX_RETIRED</td>
<td>Print Management</td>
</tr>
<tr>
<td>IMPLANT_ASSEMBLY_TEMPLATE</td>
<td>Storage</td>
</tr>
<tr>
<td>IMPLANT_ASSEMBLY_TEMPLATE_FIND</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>IMPLANT_ASSEMBLY_TEMPLATE_GET</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>Merge DICOM Toolkit Service Parameter</td>
<td>DICOM Service Class</td>
</tr>
<tr>
<td>-------------------------------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>IMPLANT_ASSEMBLY_TEMPLATE_MOVE</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>IMPLANT_TEMPLATE_GROUP</td>
<td>Storage</td>
</tr>
<tr>
<td>IMPLANT_TEMPLATE_GROUP_FIND</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>IMPLANT_TEMPLATE_GROUP_GET</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>IMPLANT_TEMPLATE_GROUP_MOVE</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>IMPLANTATION_PLAN_SR_DOCUMENT</td>
<td>Storage</td>
</tr>
<tr>
<td>INSTANCE_AVAIL_NOTIFICATION</td>
<td>Instance Availability Notification</td>
</tr>
<tr>
<td>INTRAOCULAR_LENS_CALCULATIONS</td>
<td>Storage</td>
</tr>
<tr>
<td>INVENTORY</td>
<td>Storage</td>
</tr>
<tr>
<td>INVENTORY_CREATION</td>
<td>Storage Management</td>
</tr>
<tr>
<td>INVENTORY_FIND</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>INVENTORY_GET</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>INVENTORY_MOVE</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>KERATOMETRY_MEASUREMENTS</td>
<td>Storage</td>
</tr>
<tr>
<td>KEY_OBJECT_SELECTION_DOC</td>
<td>Storage</td>
</tr>
<tr>
<td>LEGACY_CONVERTED_ENHANCED_CT_IMAGE</td>
<td>Storage</td>
</tr>
<tr>
<td>LEGACY_CONVERTED_ENHANCED_MR_IMAGE</td>
<td>Storage</td>
</tr>
<tr>
<td>LEGACY_CONVERTED_ENHANCED_PET_IMAGE</td>
<td>Storage</td>
</tr>
<tr>
<td>LENSOMETRY_MEASUREMENTS</td>
<td>Storage</td>
</tr>
<tr>
<td>MACULAR_GRID_THICKNESS_VOLUME</td>
<td>Storage</td>
</tr>
<tr>
<td>MAMMOGRAPHY_CAD_SR</td>
<td>Storage</td>
</tr>
<tr>
<td>MEDIA_CREATION_MANAGEMENT</td>
<td>Media Creation Management</td>
</tr>
<tr>
<td>MICROSCOPY_BULK_SIMPLE_ANNOTATIONS</td>
<td>Storage</td>
</tr>
<tr>
<td>MODALITY_WORKLIST_FIND</td>
<td>Modality Work list</td>
</tr>
<tr>
<td>MR_SPECTROSCOPY</td>
<td>Storage</td>
</tr>
<tr>
<td>Merge DICOM Toolkit Service Parameter</td>
<td>DICOM Service Class</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>MULTI_CHANNEL_RESPIRATORY_WAVEFORM</td>
<td>Storage</td>
</tr>
<tr>
<td>MULTIPLE_VOLUME_RENDERING_VOLUMETRICPRESENTATION_STATE</td>
<td>Storage</td>
</tr>
<tr>
<td>OPHT_VIS_FIELD_STATIC_PERIM_MEAS</td>
<td>Storage</td>
</tr>
<tr>
<td>OPHTHALMIC_AXIAL_MEASUREMENTS</td>
<td>Storage</td>
</tr>
<tr>
<td>OPHTHALMIC_OCT_BSCAN_VOLUME_ANALYSIS</td>
<td>Storage</td>
</tr>
<tr>
<td>OPHTHALMIC_OCT_EN_FACE_IMAGE</td>
<td>Storage</td>
</tr>
<tr>
<td>OPHTHALMIC_TOMOGRAPHY_IMAGE</td>
<td>Storage</td>
</tr>
<tr>
<td>OPM_THICKNESS_MAP</td>
<td>Storage</td>
</tr>
<tr>
<td>PARAMETRIC_MAP</td>
<td>Storage</td>
</tr>
<tr>
<td>PATIENT_RADIATION_DOSE_SR</td>
<td>Storage</td>
</tr>
<tr>
<td>PATIENT_ROOT_QR_FIND</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>PATIENT_ROOT_QR_GET</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>PATIENT_ROOT_QR_MOVE</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>PATIENT_STUDY_ONLY_QR_FIND_RETIRED</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>PATIENT_STUDY_ONLY_QR_GET_RETIRED</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>PATIENT_STUDY_ONLY_QR_MOVE_RETIRED</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>PERFORMED_IMAGING_AGENT_ADMINISTRATION_SR</td>
<td>Storage</td>
</tr>
<tr>
<td>PERFORMED_PROCEDURE_STEP</td>
<td>Study Management</td>
</tr>
<tr>
<td>PERFORMED_PROCEDURE_STEP_NOTIFY</td>
<td>Study Management</td>
</tr>
<tr>
<td>PERFORMED_PROCEDURE_STEP_RETRIEVE</td>
<td>Study Management</td>
</tr>
<tr>
<td>PLANNED_IMAGING_AGENT_ADMINISTRATION_SR</td>
<td>Storage</td>
</tr>
<tr>
<td>PRESENTATION_LUT</td>
<td>Print Management</td>
</tr>
<tr>
<td>PRINT_JOB</td>
<td>Print Management</td>
</tr>
<tr>
<td>PRINT_QUEUE_MANAGEMENT</td>
<td>Print Management</td>
</tr>
<tr>
<td>PRINTER</td>
<td>Print Management</td>
</tr>
<tr>
<td>Merge DICOM Toolkit Service Parameter</td>
<td>DICOM Service Class</td>
</tr>
<tr>
<td>--------------------------------------------------------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>PRINTER_CONFIGURATION</td>
<td>Print Management</td>
</tr>
<tr>
<td>PROCEDURAL_EVENT_LOGGING</td>
<td>Application Event Logging</td>
</tr>
<tr>
<td>PROCEDURE_LOG</td>
<td>Storage</td>
</tr>
<tr>
<td>PRODUCT_CHARACTERISTICS_QUERY</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>PROTOCOL_APPROVAL</td>
<td>Storage</td>
</tr>
<tr>
<td>PROTOCOL_APPROVAL_FIND</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>PROTOCOL_APPROVAL_GET</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>PROTOCOL_APPROVAL_MOVE</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>PULL_PRINT_REQUEST</td>
<td>Print Management</td>
</tr>
<tr>
<td>RADIOPHARMACEUTICAL_RADIATION_DOSE_SR</td>
<td>Storage</td>
</tr>
<tr>
<td>RAW_DATA</td>
<td>Storage</td>
</tr>
<tr>
<td>REAL_WORLD_VALUE_MAPPING</td>
<td>Storage</td>
</tr>
<tr>
<td>REFERENCED_IMAGE_BOX</td>
<td>Print Management</td>
</tr>
<tr>
<td>RENDITION_SELECTION_DOCUMENT_REAL_TIME_COMMUNICATION</td>
<td>Storage</td>
</tr>
<tr>
<td>REPOSITORY_QUERY</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>RESPIRATORY_WAVEFORM</td>
<td>Storage</td>
</tr>
<tr>
<td>ROBOTIC_ARM_RADIATION</td>
<td>Storage</td>
</tr>
<tr>
<td>ROBOTIC_ARM_RADIATION_RECORD</td>
<td>Storage</td>
</tr>
<tr>
<td>ROUTINE_SCALP_ELECTROENCEPHALOGRAM_WAVEFORM</td>
<td>Storage</td>
</tr>
<tr>
<td>RT_BEAMS_DELIVERY_INSTRUCTION</td>
<td>Storage</td>
</tr>
<tr>
<td>RT_BRACHY_APP_SETUP_DELIVERY_INSTR</td>
<td>Storage</td>
</tr>
<tr>
<td>RT_CONVENTIONAL_MACHINE_VERIFICATION</td>
<td>Verification</td>
</tr>
<tr>
<td>RT_ION_MACHINE_VERIFICATION</td>
<td>Verification</td>
</tr>
<tr>
<td>RT_PHYSICIAN_INTENT</td>
<td>Storage</td>
</tr>
<tr>
<td>RT_RADIATION_RECORD_SET</td>
<td>Storage</td>
</tr>
<tr>
<td>Merge DICOM Toolkit Service Parameter</td>
<td>DICOM Service Class</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>RT_RADIATION_SALVAGE_RECORD</td>
<td>Storage</td>
</tr>
<tr>
<td>RT_RADIATION_SET</td>
<td>Storage</td>
</tr>
<tr>
<td>RT_RADIATION_SET_DELIVERY_INSTRUCTION</td>
<td>Storage</td>
</tr>
<tr>
<td>RT_SEGMENT_ANNOTATION</td>
<td>Storage</td>
</tr>
<tr>
<td>RT_TREATMENT_PREPARATION</td>
<td>Storage</td>
</tr>
<tr>
<td>SC_MULTIFRAME_GRAYSCALE_BYTE</td>
<td>Storage</td>
</tr>
<tr>
<td>SC_MULTIFRAME_GRAYSCALE_WORD</td>
<td>Storage</td>
</tr>
<tr>
<td>SC_MULTIFRAME_SINGLE_BIT</td>
<td>Storage</td>
</tr>
<tr>
<td>SC_MULTIFRAME_TRUE_COLOR</td>
<td>Storage</td>
</tr>
<tr>
<td>SEGMENTATION</td>
<td>Storage</td>
</tr>
<tr>
<td>SEGMENTED_VOLUME_RENDERING_VOLUMETRIC_PRESENTATION_STATE</td>
<td>Storage</td>
</tr>
<tr>
<td>SIMPLIFIED_ADULT_ECHO_SR</td>
<td>Storage</td>
</tr>
<tr>
<td>SLEEP_ELECTROENCEPHALOGRAM_WAVEFORM</td>
<td>Storage</td>
</tr>
<tr>
<td>SPATIAL_FIDUCIALS</td>
<td>Storage</td>
</tr>
<tr>
<td>SPATIAL_REGISTRATION</td>
<td>Storage</td>
</tr>
<tr>
<td>SPECTACLE_PRESCRIPTION_REPORT</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_BASIC_TEXT_SR</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_BLENDING_SOFTCOPY_PS</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_COLOR_SOFTCOPY_PS</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_COMPREHENSIVE_SR</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_CR</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_CT</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_CURVE</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_DX_PRESENT</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_DX_PROCESS</td>
<td>Storage</td>
</tr>
<tr>
<td>Merge DICOM Toolkit Service Parameter</td>
<td>DICOM Service Class</td>
</tr>
<tr>
<td>----------------------------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>STANDARD_ECHO</td>
<td>Verification</td>
</tr>
<tr>
<td>STANDARD_ENCAPSULATED_PDF</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_ENHANCED_SR</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_GRAYSCALE_SOFTCOPY_PS</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_HARDCOPY_COLOR</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_HARDCOPY_GRAYSCALE</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_IO_PRESENT</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_IO_PROCESS</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_IVOCT_PRESENT</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_IVOCT_PROCESS</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_MG_PRESENT</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_MG_PROCESS</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_MODALITY_LUT</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_MR</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_NM</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_NM_RETIRED</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_OPHTHALMIC_16_BIT</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_OPHTHALMIC_8_BIT</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_OVERLAY</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_PET</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_PET_CURVE</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_PRINT_STORAGE</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_PSEUDOCOLOR_SOFTCOPY_PS</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_RT_BEAMS_TREAT</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_RT_BRACHY_TREAT</td>
<td>Storage</td>
</tr>
<tr>
<td>Merge DICOM Toolkit Service Parameter</td>
<td>DICOM Service Class</td>
</tr>
<tr>
<td>-----------------------------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>STANDARD_RT_DOSE</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_RT_IMAGE</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_RT_ION_BEAMS_TREAT</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_RT_ION_PLAN</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_RT_PLAN</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_RT_STRUCTURE_SET</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_RT_TREAT_SUM</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_SEC_CAPTURE</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_US</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_US_MF</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_US_MF_RETIRED</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_US_RETIRED</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_VIDEO_ENDOSCOPIC</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_VIDEO_MICROSCOPIC</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_VIDEO_PHOTOGRAPHIC</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_VL_ENDOSCOPIC</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_VL_MICROSCOPIC</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_VL_PHOTOGRAPHIC</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_VL_SLIDE_MICROSCOPIC</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_VOI_LUT</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_WAVEFORM_12_LEAD_ECG</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_WAVEFORM_AMBULATORY_ECG</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_WAVEFORM_BASIC_VOICE_AUDIO</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_WAVEFORM_CARDIAC_EP</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_WAVEFORM_GENERAL_ECG</td>
<td>Storage</td>
</tr>
<tr>
<td>Merge DICOM Toolkit Service Parameter</td>
<td>DICOM Service Class</td>
</tr>
<tr>
<td>----------------------------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>STANDARD_WAVEFORM_HEMODYNAMIC</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_XRAY_ANGIO</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_XRAY_ANGIO_BIPLANE</td>
<td>Storage</td>
</tr>
<tr>
<td>STANDARD_XRAY_RF</td>
<td>Storage</td>
</tr>
<tr>
<td>STEREOMETRIC_RELATIONSHIP</td>
<td>Storage</td>
</tr>
<tr>
<td>STORAGE_COMMITMENT_PULL</td>
<td>Storage Commitment</td>
</tr>
<tr>
<td>STORAGE_COMMITMENT_PUSH</td>
<td>Storage Commitment</td>
</tr>
<tr>
<td>STUDY_COMPONENT_MANAGEMENT</td>
<td>Study Management</td>
</tr>
<tr>
<td>STUDY_CONTENT_NOTIFICATION</td>
<td>Study Content Notification</td>
</tr>
<tr>
<td>STUDY_ROOT_QR_FIND</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>STUDY_ROOT_QR_GET</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>STUDY_ROOT_QR_MOVE</td>
<td>Query/Retrieve</td>
</tr>
<tr>
<td>SUBJ_REFRACTION_MEASUREMENTS</td>
<td>Storage</td>
</tr>
<tr>
<td>SUBSTANCE_ADMIN_LOGGING</td>
<td>Storage</td>
</tr>
<tr>
<td>SUBSTANCE_APPROVAL_QUERY</td>
<td>Storage</td>
</tr>
<tr>
<td>SURFACE_SCAN_MESH</td>
<td>Storage</td>
</tr>
<tr>
<td>SURFACE_SCAN_POINT_CLOUD</td>
<td>Storage</td>
</tr>
<tr>
<td>SURFACE_SEGMENTATION</td>
<td>Storage</td>
</tr>
<tr>
<td>TOMOTHERAPEUTIC_RADIATION</td>
<td>Storage</td>
</tr>
<tr>
<td>TOMOTHERAPEUTIC_RADIATION_RECORD</td>
<td>Storage</td>
</tr>
<tr>
<td>TRACTOGRAPHY_RESULTS</td>
<td>Storage</td>
</tr>
<tr>
<td>UPS_EVENT_SOP</td>
<td>Unified Procedure Step Management</td>
</tr>
<tr>
<td>UPS_EVENT_SOP_TRIAL_RETIRED</td>
<td>Unified Procedure Step Management</td>
</tr>
<tr>
<td>UPS_PULL_SOP</td>
<td>Unified Procedure Step Management</td>
</tr>
<tr>
<td>Merge DICOM Toolkit Service Parameter</td>
<td>DICOM Service Class</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>UPS_PULL_SOP_TRIAL_RETIRED</td>
<td>Unified Procedure Step Management</td>
</tr>
<tr>
<td>UPS_PUSH_SOP</td>
<td>Unified Procedure Step Management</td>
</tr>
<tr>
<td>UPS_PUSH_SOP_TRIAL_RETIRED</td>
<td>Unified Procedure Step Management</td>
</tr>
<tr>
<td>UPS_QUERY_SOP</td>
<td>Unified Procedure Step Management</td>
</tr>
<tr>
<td>UPS_WATCH_SOP</td>
<td>Unified Procedure Step Management</td>
</tr>
<tr>
<td>UPS_WATCH_SOP_TRIAL_RETIRED</td>
<td>Unified Procedure Step Management</td>
</tr>
<tr>
<td>VIDEO_ENDOSCOPIC_IMAGE_REAL_TIME_COMMUNICATION</td>
<td>Storage</td>
</tr>
<tr>
<td>VIDEO_PHOTOGRAPHIC_IMAGE_REAL_TIME_COMMUNICATION</td>
<td>Storage</td>
</tr>
<tr>
<td>VISUAL_ACUITY_MEASUREMENTS</td>
<td>Storage</td>
</tr>
<tr>
<td>VL_WHOLE_SLIDE_MICROSCOPY_IMAGE</td>
<td>Storage</td>
</tr>
<tr>
<td>VOI_LUT_BOX</td>
<td>Print Management</td>
</tr>
<tr>
<td>VOLUME_RENDERING_VOLUMETRIC_PRESENTATION_STATE</td>
<td>Storage</td>
</tr>
<tr>
<td>WIDE_FIELD_OPHTHALMIC_PHOTO_3D_COORDINATES</td>
<td>Storage</td>
</tr>
<tr>
<td>WIDE_FIELD_OPHTHALMIC_PHOTO_STEREOGRAPHIC_PROJ</td>
<td>Storage</td>
</tr>
<tr>
<td>XA_DEFINED_PROCEDURE_PROTOCOL</td>
<td>Storage</td>
</tr>
<tr>
<td>XA_PERFORMED_PROCEDURE_PROTOCOL</td>
<td>Storage</td>
</tr>
<tr>
<td>XA_XRF_GRAYSCALE_SOFTCOPY_PS</td>
<td>Storage</td>
</tr>
<tr>
<td>XRAY_3D_ANGIO_IMAGE</td>
<td>Storage</td>
</tr>
<tr>
<td>XRAY_3D_CRANIO_IMAGE</td>
<td>Storage</td>
</tr>
<tr>
<td>XRAY_RADIATION_DOSE_SR</td>
<td>Storage</td>
</tr>
<tr>
<td>BASIC_COLOR_PRINT_MANAGEMENT (META_SOP)</td>
<td>Print Management</td>
</tr>
<tr>
<td>BASIC_GRAYSCALE_PRINT_MANAGEMENT (META_SOP)</td>
<td>Print Management</td>
</tr>
</tbody>
</table>
### Transfer Syntax Lists

For DICOM Toolkit users, Merge DICOM Toolkit allows you to define transfer syntaxes supported for each service in a service list. This functionality is implemented through the use of transfer syntax lists. The basic service lists discussed above can be modified to include these transfer syntax lists. The following is an example service list that has transfer syntaxes specified for each service:

```
[Storage_Service_List]
SERVICES_SUPPORTED  = 3 # Number of Services
SERVICE_1           = STANDARD_MR
SYNTAX_LIST_1       = MR_Syntax_List
SERVICE_2           = STANDARD_US
SYNTAX_LIST_2       = US_Syntax_List
SERVICE_3           = STANDARD_CT
SYNTAX_LIST_3       = CT_Syntax_List

[MR_Syntax_List]
SYNTAXES_SUPPORTED  = 4 # Number of Syntaxes
SYNTAX_1            = JPEG_BASELINE
SYNTAX_2            = EXPLICIT_BIG_ENDIAN
SYNTAX_3            = EXPLICIT_LITTLE_ENDIAN
SYNTAX_4            = IMPLICIT_LITTLE_ENDIAN

[US_Syntax_List]
SYNTAXES_SUPPORTED  = 2 # Number of Syntaxes
SYNTAX_1            = RLE
SYNTAX_2            = IMPLICIT_BIG_ENDIAN

[CT_Syntax_List]
SYNTAXES_SUPPORTED  = 2 # Number of Syntaxes
SYNTAX_1            = EXPLICIT_LITTLE_ENDIAN
SYNTAX_2            = IMPLICIT_LITTLE_ENDIAN
```
[Storage_Service_List] lists various standard storage service class services used by Merge DICOM Toolkit. The SYNTAX_LIST_N parameter has been added to this example to specify a transfer syntax list for each service. This optional parameter is set to the name of another section in the application profile which lists DICOM transfer syntaxes to be negotiated. If this parameter is not set, the default non-compressed transfers syntaxes (implicit VR little endian, explicit VR little endian, and explicit VR big endian) are negotiated.

The [MR_Syntax_List], [US_Syntax_List] and [CT_Syntax_List] sections each define a separate transfer syntax list for the MR, US, and CT services, respectively. Merge DICOM Toolkit currently supports all transfer syntaxes specified in the DICOM standard. The names used for these transfer syntaxes are defined below in this Reference Manual.

For server (SCP) applications, the order in which transfer syntaxes are specified in a transfer syntax list dictates the priority Merge DICOM Toolkit places on them during association negotiation. For example, using the [US_Syntax_List] specified above, if a client (SCU) proposes the Ultrasound storage service with the RLE compressed transfer syntax and the implicit VR little endian transfer syntax, Merge DICOM Toolkit selects the RLE transfer syntax because it is listed first in the transfer syntax list.

When a transfer syntax list is not specified in a service list, the priority Merge DICOM Toolkit places on transfer syntaxes during association negotiation is dependent on the hardware platform. On little endian machines (Intel and DEC Alpha based systems) the priority order is: Explicit VR Little Endian, Implicit VR Little Endian, and Explicit VR Big Endian. On big endian machines the priority order is: Explicit VR Big Endian, Explicit VR Little Endian, and Implicit VR Little Endian.

The following is a list of the currently supported transfer syntaxes:

*Table B-5: Transfer Syntax List Parameters*

<table>
<thead>
<tr>
<th>Merge DICOM Toolkit Transfer Syntax Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMPLICIT_LITTLE_ENDIAN</td>
<td>Implicit VR Little Endian: Default Transfer Syntax for DICOM</td>
</tr>
<tr>
<td>IMPLICIT_BIG_ENDIAN</td>
<td>Implicit VR Big Endian</td>
</tr>
<tr>
<td>EXPLICIT_LITTLE_ENDIAN</td>
<td>Explicit VR Little Endian</td>
</tr>
<tr>
<td>EXPLICIT_BIG_ENDIAN</td>
<td>Explicit VR Big Endian</td>
</tr>
<tr>
<td>DEFLATED_EXPLICIT_LITTLE_ENDIAN</td>
<td>Deflated Explicit VR Little Endian</td>
</tr>
<tr>
<td>ENCAPSULATED_UNCOMPRESSED_ELE</td>
<td>Encapsulated Explicit VR Little Endian</td>
</tr>
<tr>
<td>RLE</td>
<td>Run length Encoding</td>
</tr>
<tr>
<td>HEVC_H265_M10P_LEVEL_5_1</td>
<td>HEVC/H.265 Main 10 Profile / Level 5.1</td>
</tr>
<tr>
<td>HEVC_H265_MP_LEVEL_5_1</td>
<td>HEVC/H.265 Main Profile / Level 5.1</td>
</tr>
<tr>
<td>Merge DICOM Toolkit Transfer Syntax Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>JPEG_BASELINE</td>
<td>JPEG Baseline (Process 1): Default Transfer Syntax for Lossy JPEG 8 Bit Image Compression</td>
</tr>
<tr>
<td>JPEG_EXTENDED_2_4</td>
<td>JPEG Extended (Process 2 &amp; 4): Default Transfer Syntax for Lossy JPEG 12 Bit Image Compression (Process 4 only)</td>
</tr>
<tr>
<td>JPEG_EXTENDED_3_5</td>
<td>JPEG Extended (Process 3 &amp; 5)</td>
</tr>
<tr>
<td>JPEG_SPEC_NON_HIER_6_8</td>
<td>JPEG Spectral Selection, Non-Hierarchical (Process 6 &amp; 8)</td>
</tr>
<tr>
<td>JPEG_SPEC_NON_HIER_7_9</td>
<td>JPEG Spectral Selection, Non-Hierarchical (Process 7 &amp; 9)</td>
</tr>
<tr>
<td>JPEG_FULL_PROG_NON_HIER_10_12</td>
<td>JPEG Full Progression, Non-Hierarchical (Process 10 &amp; 12)</td>
</tr>
<tr>
<td>JPEG_FULL_PROG_NON_HIER_11_13</td>
<td>JPEG Full Progression, Non-Hierarchical (Process 11 &amp; 13)</td>
</tr>
<tr>
<td>JPEG_LOSSLESS_NON_HIER_14</td>
<td>JPEG Lossless, Non-Hierarchical (Process 14)</td>
</tr>
<tr>
<td>JPEG_LOSSLESS_NON_HIER_15</td>
<td>JPEG Lossless, Non-Hierarchical (Process 15)</td>
</tr>
<tr>
<td>JPEG_EXTENDED_HIER_16_18</td>
<td>JPEG Extended, Hierarchical (Process 16 &amp; 18)</td>
</tr>
<tr>
<td>JPEG_EXTENDED_HIER_17_19</td>
<td>JPEG Extended, Hierarchical (Process 17 &amp; 19)</td>
</tr>
<tr>
<td>JPEG_SPEC_HIER_20_22</td>
<td>JPEG Spectral Selection, Hierarchical (Process 20 &amp; 22)</td>
</tr>
<tr>
<td>JPEG_SPEC_HIER_21_23</td>
<td>JPEG Spectral Selection, Hierarchical (Process 21 &amp; 23)</td>
</tr>
<tr>
<td>JPEG_FULL_PROG_HIER_24_26</td>
<td>JPEG Full Progression, Hierarchical (Process 24 &amp; 26)</td>
</tr>
<tr>
<td>JPEG_FULL_PROG_HIER_25_27</td>
<td>JPEG Full Progression, Hierarchical (Process 25 &amp; 27)</td>
</tr>
<tr>
<td>JPEG_LOSSLESS_HIER_28</td>
<td>JPEG Lossless, Hierarchical (Process 28)</td>
</tr>
<tr>
<td>JPEG_LOSSLESS_HIER_29</td>
<td>JPEG Lossless, Hierarchical (Process 29)</td>
</tr>
<tr>
<td>Merge DICOM Toolkit Transfer Syntax Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>JPEG_LOSSLESS_HIER_14</td>
<td>JPEG Lossless, Hierarchical, First-Order Prediction (Process 14 [Selection Value 1]): Default Transfer Syntax for Lossless JPEG Image Compression</td>
</tr>
<tr>
<td>JPEG_2000_LOSSLESS_ONLY</td>
<td>JPEG 2000, Lossless</td>
</tr>
<tr>
<td>JPEG_2000</td>
<td>JPEG 2000, Lossless or Lossy</td>
</tr>
<tr>
<td>JPEG_2000_MC_LOSSLESS_ONLY</td>
<td>JPEG 2000 Part 2 Multi-component Image Compression (Lossless Only)</td>
</tr>
<tr>
<td>JPEG_2000_MC</td>
<td>JPEG 2000 Part 2 Multi-component Image Compression</td>
</tr>
<tr>
<td>JPEG_LS_LOSSLESS</td>
<td>JPEG LS Lossless</td>
</tr>
<tr>
<td>JPEG_LS_LOSSY</td>
<td>JPEG LS Lossy (Near-Lossless)</td>
</tr>
<tr>
<td>JPIP_REFERENCED</td>
<td>JPIP Referenced</td>
</tr>
<tr>
<td>JPIP_REFERENCED_DEFLATE</td>
<td>JPIP Referenced Deflate</td>
</tr>
<tr>
<td>MPEG2_MPHL</td>
<td>MPEG2 Main Profile / High Level</td>
</tr>
<tr>
<td>MPEG2_MPML</td>
<td>MPEG2 Main Profile @ Main Level</td>
</tr>
<tr>
<td>MPEG4_AVC_H264_BDC_HP_LEVEL_4_1</td>
<td>MPEG-4 AVC/H.264 BD-compatible High Profile / Level 4.1</td>
</tr>
<tr>
<td>MPEG4_AVC_H264_HP_LEVEL_4_1</td>
<td>MPEG-4 AVC/H.264 High Profile / Level 4.1</td>
</tr>
<tr>
<td>MPEG4_AVC_H264_HP_LEVEL_4_2_2D</td>
<td>MPEG-4 AVC/H.264 High Profile / Level 4.2 For 2D Video</td>
</tr>
<tr>
<td>MPEG4_AVC_H264_HP_LEVEL_4_2_3D</td>
<td>MPEG-4 AVC/H.264 High Profile / Level 4.2 For 3D Video</td>
</tr>
<tr>
<td>MPEG4_AVC_H264_STEREO_HP_LEVEL_4_2</td>
<td>MPEG-4 AVC/H.264 Stereo High Profile / Level 4.2</td>
</tr>
<tr>
<td>SMPTE_ST_2110_20_UNCOMPRESSED_INTERLACED_ACTIVE_VIDEO</td>
<td>SMPTE ST 2110-20 Uncompressed Interlaced Active Video</td>
</tr>
<tr>
<td>SMPTE_ST_2110_20_UNCOMPRESSED_PROGRESSIVE_ACTIVE_VIDEO</td>
<td>SMPTE ST 2110-20 Uncompressed Progressive Active Video</td>
</tr>
<tr>
<td>SMPTE_ST_2110_30_PCM_DIGITAL_AUDIO</td>
<td>SMPTE ST 2110-30 PCM Digital Audio</td>
</tr>
</tbody>
</table>
Merge DICOM Toolkit™ Transfer Syntax

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRIVATE_SYNTAX_1</td>
<td>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.</td>
</tr>
<tr>
<td>PRIVATE_SYNTAX_2</td>
<td>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.</td>
</tr>
</tbody>
</table>

### Role Negotiation

Merge DICOM Toolkit also supports DICOM role negotiation through its service lists. Whereas in previous examples, the same service list could be used for both client (SCU) and server (SCP), these service lists are specific to the role to be negotiated for each service.

```
[SCU_Service_List]
SERVICES_SUPPORTED     = 1 # Number of Services
SERVICE_1              = STORAGE_COMMITMENT_PUSH
ROLE_1                 = SCU

[SCP_Service_List]
SERVICES_SUPPORTED     = 1 # Number of Services
SERVICE_1              = STORAGE_COMMITMENT_PUSH
ROLE_1                 = SCP
```

In this case, the [SCU_Service_List] supports the Storage Commitment Push SOP class as an SCU, and the [SCP_Service_List] supports the Storage Commitment Push SOP class as an SCP. Merge DICOM Toolkit negotiates the association based on the settings for these roles.

The role for a service can be defined as SCU, SCP, BOTH or remain undefined. Table 9 contains a complete list of configurable roles for both requestors and acceptors along with the resultant negotiated roles.

**NOTE:** In some cases, a service will be rejected because the roles being negotiated do not match.
Table B-6: Negotiated Roles

<table>
<thead>
<tr>
<th>Requestor's Configured Role</th>
<th>Acceptor's Configured Role</th>
<th>Requestor's Negotiated Role</th>
<th>Acceptor's Negotiated Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCU</td>
<td>SCP</td>
<td>SCU</td>
<td>SCP</td>
</tr>
<tr>
<td>SCU</td>
<td>Rejected</td>
<td>Rejected</td>
<td>Rejected</td>
</tr>
<tr>
<td>BOTH</td>
<td>SCU</td>
<td>SCP</td>
<td>SCP</td>
</tr>
<tr>
<td>NOT DEFINED</td>
<td>SCU</td>
<td>SCP</td>
<td>SCP</td>
</tr>
<tr>
<td>SCP</td>
<td>SCP</td>
<td>Rejected</td>
<td>Rejected</td>
</tr>
<tr>
<td>SCU</td>
<td>SCP</td>
<td>SCU</td>
<td>SCU</td>
</tr>
<tr>
<td>BOTH</td>
<td>SCP</td>
<td>SCU</td>
<td>SCU</td>
</tr>
<tr>
<td>NOT DEFINED</td>
<td>Rejected</td>
<td>Rejected</td>
<td>Rejected</td>
</tr>
<tr>
<td>BOTH</td>
<td>SCP</td>
<td>SCU</td>
<td>SCP</td>
</tr>
<tr>
<td>SCU</td>
<td>SCP</td>
<td>SCU</td>
<td>SCP</td>
</tr>
<tr>
<td>BOTH</td>
<td>BOTH</td>
<td>BOTH</td>
<td>BOTH</td>
</tr>
<tr>
<td>NOT DEFINED</td>
<td>SCP</td>
<td>SCP</td>
<td>SCP</td>
</tr>
<tr>
<td>NOT DEFINED</td>
<td>SCP</td>
<td>SCU</td>
<td>SCP</td>
</tr>
<tr>
<td>SCU</td>
<td>Rejected</td>
<td>Rejected</td>
<td>Rejected</td>
</tr>
<tr>
<td>BOTH</td>
<td>SCU</td>
<td>SCP</td>
<td>SCP</td>
</tr>
<tr>
<td>NOT DEFINED</td>
<td>SCU</td>
<td>SCP</td>
<td>SCP</td>
</tr>
</tbody>
</table>

DICOM Asynchronous Communication

Merge DICOM Toolkit optionally supports DICOM Asynchronous Operations Window Negotiation through service lists. The service list in this case can be used for both the client (SCU) and server (SCP). The following is a sample service list that configures DICOM asynchronous communication negotiation:

```
[SCU_OrSCP_Service_List]
SERVICES_SUPPORTED = 1 # Number of Services
MAX_OPERATIONS_INVOKED = 10
MAX_OPERATIONS_PERFORMED = 10
SERVICE_1 = STANDARD_MR
```
In this case, the [SCU_Or_SCP_Service_List] supports the Standard MR SOP Class. For all services, it supports 10 maximum operations invoked and 10 maximum operations performed. When MAX_OPERATIONS_INVOKED and MAX_OPERATIONS_PERFORMED are not included in the service list, asynchronous communications are not negotiated. See a subsequent section for details on implementing DICOM asynchronous communications with Merge DICOM Toolkit.

**Extended Negotiation**

Merge DICOM Toolkit optionally supports configuration of DICOM extended negotiation information in service lists. Currently, the DICOM standard allows extended negotiation information for Storage and Query/Retrieve Service classes as defined in PS3.4 of the standard. The extended negotiation information can be set for only the client (SCU). Server applications utilizing extended negotiation must set this information with a call to MC_Set_Negotiation_Info_For_Association.

```
[SCU_Service_List]
SERVICES_SUPPORTED = 2 # Number of Services
SERVICE_1          = STUDY_ROOT_QR_FIND
EXT_NEG_INFO_1     = 0x01
SERVICE_2          = STUDY_ROOT_QR_MOVE
EXT_NEG_INFO_2     = 0x01
```

In this case, the [SCU_Service_List] supports the Study Root Q/R Find and Move services. Both services have set a single byte of extended negotiation information set to hexadecimal 0x01 (this implies the Client supports relational Queries and Moves).

Multiple hexadecimal bytes can be set in the service list by listing each byte in the format "0x00 0x01 0x02".

**Related General SOP Classes and Service Classes**

DICOM Supplement 90 defines a mechanism in association negotiation for identifying when an SOP Class is a customization of a generalized SOP Class. It also defines a method to identify the service class of an SOP Class that is proposed by an SCU. This allows flexibility in an SCP to support service classes for which it supports the generalized version of an SOP class, but does not explicitly support the customized SOP class. It also allows a mechanism to easily make an SCP that supports all storage service class SOP classes that are proposed to it.

Related General SOP classes can be supported in the application profile by defining a service list containing the related general SOP classes for a given SOP class, then assigning the service list to the SOP class. The following example demonstrates how this is done:

```
[SCU_SR_General]
SERVICES_SUPPORTED = 2
SERVICE_1          = STANDARD_ENHANCED_SR
SERVICE_2          = STANDARD_COMPREHENSIVE_SR

[SCU_DX_General]
SERVICES_SUPPORTED = 1
SERVICE_1          = STANDARD_DX_PRESENT
```
In this case, the \texttt{SCU\_SR\_General} service list contains the related general SOP Classes for the \texttt{STANDARD\_BASIC\_TEXT\_SR} service. The \texttt{REL\_GENERAL\_1} option points to the service list to use as the related general services for \texttt{SERVICE\_1}.

The example above also shows how the service class can be defined for each SOP Class within a service list. For instance, \texttt{SERVICE\_CLASS\_3} in the example specifies the service class for \texttt{SERVICE\_3}. The UID for the Storage Service Class as defined in Supplement 90 is used.

The service lists above are only utilized by SCU applications. For SCP applications, there are several configuration options that define how Merge DICOM Toolkit negotiates an association when related general SOP Classes are included, or the Service Class is included for an SOP Class.

When the \texttt{ACCEPT\_STORAGE\_SERVICE\_CONTEXTS} configuration option is set to Yes, Merge DICOM Toolkit accepts any proposed SOP Class that is defined as supporting the Storage Service Class.

When the \texttt{ACCEPT\_RELATED\_GENERAL\_SERVICES} configuration option is set to Yes, Merge DICOM Toolkit accepts any SOP Class proposed if the SCP lists any of the related general SOP Classes defined for the SOP Class proposed.

\section*{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 \texttt{MERGECOM\_3\_PROFILE} parameter of the \texttt{[MergeCOM3]} section of the \texttt{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 default TCP/IP listen port for a DICOM server (SCP) application.
Most importantly, you must place the license number you received when you purchased the Merge DICOM Toolkit in the [ASSOC_PARMS] section of the system profile. If the license you received with your Toolkit was DA53-31D34, you would set it in the [ASSOC_PARMS] section as follows:

```
[ASSOC_PARMS]
LICENSE = DA53-31D34
IMPLEMENTATION_CLASS_UID = 2.16.840.1.113669.2.1.2
IMPLEMENTATION_VERSION = MergeCOM3_340
ACCEPT_MULTIPLE_PRES_CONTEXTS = Yes
```

Both Toolkit sample applications and your own applications which use the DICOM Toolkit Library will not work without a valid license number.

The example of the [ASSOC_PARMS] section of the system profile also contains a sample implementation class UID and implementation version configuration values. The implementation class UID is intended by the DICOM standard to be unique for major revisions of an Application Entity. The implementation version is intended to be unique for minor revisions of an Application Entity. These configuration values are used during association negotiation by Merge DICOM Toolkit, and are intended to aid in tracking versions of applications in the field.

The ACCEPT_MULTIPLE_PRES_CONTEXTS configuration value is used by server (SCP) applications. This value determines whether multiple presentation contexts can be negotiated for a single DICOM Service.

As mentioned earlier, a listen port must be identified for your server AE. Port 104 is the standard DICOM listen port. This, along with the number of simultaneous TCP connection requests that can be queued up for acceptance (pending) for Merge DICOM Toolkit, is specified in the [TRANSPORT_PARMS] section.

The MAX_PENDING_CONNECTIONS setting in the “mergecom.pro” file refers to the maximum number of outstanding connection requests per listener socket. Note that this is not the "Maximum number of simultaneous associations". The value of this configuration is passed by the toolkit to the listen() call on the socket as the backlog parameter and it specifies how many pending connections can be queued at any given time.

The MAX_PENDING_CONNECTIONS configuration option affects the accepting of associations but not the requesting of associations and it affects the behavior at the TCP level. In the default case, if more than 5 association requests arrive at once then only the first 5 will be accepted by TCP and passed to Merge DICOM Toolkit, the others would be refused at the TCP level.

```
[TRANSPORT_PARMS]
TCPIP_LISTEN_PORT = 104
# Max number of open listen channels
MAX_PENDING_CONNECTIONS = 5
```

An important section of the System Profile is the [MESSAGE_PARMS] section:

```
[MESSAGE_PARMS]
LARGE_DATA_STORE = FILE | MEM Default = FILE
LARGE_DATA_SIZE = 200
OBOW_BUFFER_SIZE = 4096
DICTIONARY_ACCESS = MEM | FILE Default = FILE
```
DICTIONARY_FILE         = /users/mc3adv/mrgcom3.dct
TEMP_FILE_DIRECTORY     = /users/mc3adv/tmp_files/
MSG_INFO_FILE           = /users/mc3adv/mrgcom3.msg

The LARGE_DATA_STORE parameter informs the toolkit where it should store large data: either in memory, or in temporary files on disk. Large data is defined as a value for an attribute larger than LARGE_DATA_SIZE bytes. Pixel data associated with a medical image would most certainly be considered large data. If you are running your process on a resource rich system that supplies plenty of physical and virtual memory, you should select LARGE_DATA_STORE = MEM to improve your performance. If your process is not so fortunate or you are dealing with messages with very large data values, you will want to use LARGE_DATA_STORE = FILE. In this case, the DICOM Toolkit will manage the large data in temporary files located in the TEMP_FILE_DIRECTORY you specify.

Large data that is of value representation OB (Other Byte), OW (Other Word), or OF (Other Float) is treated specially by the toolkit. Pixel Data, Curves, and Overlays are composed of this type of data. You can let the toolkit manage OB/OW data for you like any other large data, or register your own Callback Function in your applications to deal with such data as it is being received or transmitted over the network. The use of Callbacks will be covered later when we discuss developing DICOM applications with the toolkit.

The OBOW_BUFFER_SIZE is used to tell the toolkit what size ‘chunks’ in bytes of OB/OW data it should read in before either writing the data to a temporary file or passing it to your Callback Function. Choosing a large number for OBOW_BUFFER_SIZE means less time spent by your application process writing to temporary files or making callbacks, but results in a larger process size.

You must also take into account that there is a 2^16 limit in how many “chunks” can be stored in the temporary files. For example, when the chunk size is 16KB (as set by OBOW_BUFFER_SIZE), the maximum value for pixel data is only 1GB (i.e., 16K * 64K). According to the DICOM standard, the maximum pixel data size can be 4GB. In order to support storing the maximum possible pixel data, the OBOW_BUFFER_SIZE must not be less than 64KB. If you need to use temporary files, you should tune this parameter to maximize performance within the constraints of your runtime environment.

The DICTIONARY_ACCESS parameter specifies whether the DICOM Data Dictionary should be loaded and stored into memory the first time it is accessed, or be accessed from a file every time it is referenced. The dictionary file is a binary file supplied with your toolkit with the default name of mrgcom3.dct. This file is accessed very frequently by the toolkit and you should set this parameter to MEM in all but the most extreme cases of memory limitation. You also specify the location and name of the data dictionary file using the DICTIONARY_FILE parameter.

Another binary file supplied with the Toolkit is the message info file. This file contains binary encoded message objects and is accessed when an application opens a message. Once open, these objects reside in memory, and are “filled in” by your application. They then become a message object instance that can be exchanged over the network. The message info file along with the data dictionary file, make possible the powerful message validation capabilities of the DICOM Toolkit. The message info file is a binary file supplied by your Toolkit with the default name of
You also specify the location and name of the message info file using the `MSG_INFO_FILE` parameter.

Tables B-7 through B-12 define how each parameter should be defined within each section of the system profile.

### Table B-7 [ASSOC_PARMS] section of system profile parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCEPT_ANY_APPLICATION_TITLE †</td>
<td>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 toolkit will report the remote application as connecting to the first application registered. DEFAULT: NO</td>
</tr>
<tr>
<td>ACCEPT_ANY_CONTEXT_NAME †</td>
<td>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</td>
</tr>
<tr>
<td>ACCEPT_ANY_HOSTNAME</td>
<td>If set to YES, the toolkit 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 toolkit will automatically reject associations from unknown hosts. DEFAULT: NO</td>
</tr>
<tr>
<td>ACCEPT_ANY_PRESENTATION_CONTEXT †</td>
<td>If set to YES, the toolkit 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 toolkit will abort associations when these values do not match. DEFAULT: NO</td>
</tr>
<tr>
<td>ACCEPT_DIFFERENT_IC_UID †</td>
<td>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</td>
</tr>
<tr>
<td>ACCEPT_DIFFERENT_VERSION †</td>
<td>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</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ACCEPT_LIST_OF_APPLICATION_TITLES</td>
<td>List of AE titles which the remote system might use when requesting an association. The parameters line should contain all AE titles separated by one of predefined delimiters: ';' '&quot;' '/' ';'. The length of each AE title can not exceed 16 characters. Example: ACCEPT_LIST_OF_APPLICATION_TITLES = MERGE_STORE_SCP/MERGE_STORE_SCU/MERGE_STORE_RQ</td>
</tr>
<tr>
<td>DEFAULT: &lt;none&gt;</td>
<td></td>
</tr>
<tr>
<td>ACCEPT_MULTIPLE_PRES_CONTEXTS</td>
<td>If set to YES, SCP applications will allow multiple presentation contexts to be negotiated for a single DICOM service. If set to NO, an SCP will only accept a single presentation context for a DICOM service. DEFAULT: YES</td>
</tr>
<tr>
<td>ACCEPTRELATEDGENERALSERVICES</td>
<td>This parameter sets the Merge DICOM Toolkit behavior in regards to support for DICOM Supplement 90. Supplement 90 defines a method for association requestors to specify the generalized version of a SOP Class. When set to YES, Merge DICOM Toolkit will allow association acceptors to accept a presentation context whose generalized SOP Class is supported; however, the customized SOP Class is not specifically supported. DEFAULT: NO</td>
</tr>
<tr>
<td>ACCEPT_STORAGE_SERVICE_CONTEXTS</td>
<td>This parameter sets the Merge DICOM Toolkit behavior in regards to support for DICOM Supplement 90. When set to YES, Merge DICOM Toolkit will accept any presentation context which is defined as a Storage Service Class SOP Class. DEFAULT: NO</td>
</tr>
<tr>
<td>ALLOW EMPTY PDV LENGTH</td>
<td>The DICOM standard specifies that PDVs shall not be sent without any content in the fragment. The toolkit however can send and accept empty PDVs. To enforce the standard requirement, this setting should be set to No. DEFAULT: YES</td>
</tr>
<tr>
<td>AUTO ECHO SUPPORT</td>
<td>If set to YES, the toolkit automatically handles C-ECHO requests when the application doesn't explicitly include STANDARD_ECHO in its supported service list. If set to NO, the toolkit rejects C-ECHO requests when the application doesn't explicitly include STANDARD_ECHO in its supported service list. DEFAULT: YES</td>
</tr>
<tr>
<td>DEFLATED EXPLICIT LITTLE_ENDIAN_SYNTAX</td>
<td>This value defines the UID of the Deflated Explicit VR Little Endian transfer syntax. DEFAULT: 1.2.840.10008.1.2.1.99</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ENCAPSULATED_UNCOMPRESSED_ELE_SYNTAX</td>
<td>This value defines the UID of the Encapsulated Uncompressed Explicit VR LittleEndian transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.1.98</td>
</tr>
<tr>
<td>EXPLICIT_BIG_ENDIAN_SYNTAX</td>
<td>This value defines the UID of the Explicit VR Big Endian transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.2</td>
</tr>
<tr>
<td>EXPLICIT_LITTLE_ENDIAN_SYNTAX</td>
<td>This value defines the UID of the Explicit VR Little Endian transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.1</td>
</tr>
<tr>
<td>HARD_CLOSE_TCP_IP_CONNECTION</td>
<td>This parameter specifies how TCP/IP connections are closed by the toolkit. When set to YES, TCP/IP connections are instantaneously closed with an RST packet. When set to NO, TCP/IP connections are closed gracefully with a FIN packet. Note, that in the NO case the toolkit must wait for an operating system dependent amount of time for the response to the FIN packet.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: YES</td>
</tr>
<tr>
<td>IMPLEMENTATION_CLASS_UID</td>
<td>The DICOM Implementation Class UID (as specified in your DICOM conformance statement).</td>
</tr>
<tr>
<td>IMPLEMENTATION_VERSION</td>
<td>The Implementation Version Number (as specified in your DICOM conformance statement).</td>
</tr>
<tr>
<td>IMPLICIT_BIG_ENDIAN_SYNTAX</td>
<td>The Implicit VR Big Endian transfer syntax is not defined by the DICOM standard. This value is provided to supply compatibility with private implementations.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: &lt;none&gt;</td>
</tr>
<tr>
<td>IMPLICIT_LITTLE_ENDIAN_SYNTAX</td>
<td>The Implicit VR Little Endian transfer syntax is the default network transfer syntax of the DICOM standard. The Implicit VR Little Endian transfer syntax must always be defined.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2</td>
</tr>
<tr>
<td>JPEG_2000_LOSSLESS_ONLY_SYNTAX</td>
<td>This value defines the UID for JPEG 2000, Lossless transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.4.90</td>
</tr>
<tr>
<td>JPEG_2000_MC_LOSSLESS_ONLY_SYNTAX</td>
<td>This value defines the UID for JPEG 2000 Part 2 Multi-component Image Compression (Lossless Only) transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.4.92</td>
</tr>
<tr>
<td>JPEG_2000_MC_SYNTAX</td>
<td>This value defines the UID for JPEG 2000 Part 2 Multi-component Image Compression transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.4.93</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>JPEG_2000_SYNTAX</td>
<td>This value defines the UID for JPEG 2000 transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.4.91</td>
</tr>
<tr>
<td>JPEG_BASELINE_SYNTAX</td>
<td>This value defines the UID for JPEG Baseline (Process 1) transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.4.50</td>
</tr>
<tr>
<td>JPEG_EXTENDED_2_4_SYNTAX</td>
<td>This value defines the UID for JPEG Extended (Process 2 &amp; 4) transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.4.51</td>
</tr>
<tr>
<td>JPEG_EXTENDED_3_5_SYNTAX</td>
<td>This value defines the UID for JPEG Extended (Process 3 &amp; 5) transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.4.52</td>
</tr>
<tr>
<td>JPEG_EXTENDED_HIER_16_18_SYNTAX</td>
<td>This value defines the UID for JPEG Extended, Hierarchical (Process 16 &amp; 18) transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.4.59</td>
</tr>
<tr>
<td>JPEG_EXTENDED_HIER_17_19_SYNTAX</td>
<td>This value defines the UID for JPEG Extended, Hierarchical (Process 17 &amp; 19) transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.4.60</td>
</tr>
<tr>
<td>JPEG_FULL_PROG_HIER_24_26_SYNTAX</td>
<td>This value defines the UID for JPEG Full Progression, Hierarchical (Process 24 &amp; 26) transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.4.63</td>
</tr>
<tr>
<td>JPEG_FULL_PROG_HIER_25_27_SYNTAX</td>
<td>This value defines the UID for JPEG Full Progression, Hierarchical (Process 25 &amp; 27) transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.4.64</td>
</tr>
<tr>
<td>JPEG_FULL_PROG_NON_HIER_10_12_SYNTAX</td>
<td>This value defines the UID for JPEG Full Progression, Non-Hierarchical (Process 10 &amp; 12) transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.4.55</td>
</tr>
<tr>
<td>JPEG_FULL_PROG_NON_HIER_11_13_SYNTAX</td>
<td>This value defines the UID for JPEG Full Progression, Non-Hierarchical (Process 11 &amp; 13) transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.4.56</td>
</tr>
<tr>
<td>JPEG_LOSSLESS_HIER_14_SYNTAX</td>
<td>This value defines the UID for JPEG Lossless, Non-Hierarchical, First-Order Prediction (Process 14, Selection Value 1) transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.4.70</td>
</tr>
<tr>
<td>JPEG_LOSSLESS_HIER_28_SYNTAX</td>
<td>This value defines the UID for JPEG Lossless, Hierarchical (Process 28) transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.4.65</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>JPEG_LOSSLESS_HIER_29_SYNTAX</td>
<td>This value defines the UID for JPEG Lossless, Hierarchical (Process 29) transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.4.66</td>
</tr>
<tr>
<td>JPEGLOSSLESS_NON_HIER_14_SYNTAX</td>
<td>This value defines the UID for JPEG Lossless, Non-Hierarchical (Process 14) transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.4.57</td>
</tr>
<tr>
<td>JPEGLOSSLESS_NON_HIER_15_SYNTAX</td>
<td>This value defines the UID for JPEG Lossless, Non-Hierarchical (Process 15) transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.4.58</td>
</tr>
<tr>
<td>JPEG_LS_LOSSLESS_SYNTAX</td>
<td>This value defines the UID for JPEG LS Lossless transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.4.80</td>
</tr>
<tr>
<td>JPEG_LS_LOSSY_SYNTAX</td>
<td>This value defines the UID for JPEG LS Lossy (Near Lossless) transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.4.81</td>
</tr>
<tr>
<td>JPEG_SPEC_HIER_20_22_SYNTAX</td>
<td>This value defines the UID for JPEG Spectral Selection, Hierarchical (Process 20 &amp; 22) transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.4.61</td>
</tr>
<tr>
<td>JPEG_SPEC_HIER_21_23_SYNTAX</td>
<td>This value defines the UID for JPEG Spectral Selection, Hierarchical (Process 21 &amp; 23) transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.4.62</td>
</tr>
<tr>
<td>JPEG_SPEC_NON_HIER_6_8_SYNTAX</td>
<td>This value defines the UID for JPEG Spectral Selection, Non-Hierarchical (Process 6 &amp; 8) transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.4.53</td>
</tr>
<tr>
<td>JPEG_SPEC_NON_HIER_7_9_SYNTAX</td>
<td>This value defines the UID for JPEG Spectral Selection, Non-Hierarchical (Process 7 &amp; 9) transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.4.54</td>
</tr>
<tr>
<td>JPIPREFERENCED_DEFLATE_SYNTAX</td>
<td>This value defines the UID for JPIP Referenced Deflate transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.4.95</td>
</tr>
<tr>
<td>JPIPREFERENCED_SYNTAX</td>
<td>This value defines the UID for JPIP Referenced transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.4.94</td>
</tr>
<tr>
<td>LICENSE</td>
<td>The Merge DICOM Toolkit license number that was supplied when the toolkit was purchased.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LOCAL_APPL_CONTEXT_NAME</td>
<td>The DICOM Application Context Name (UID) (as specified in the DICOM Standard). DEFAULT: 1.2.840.10008.3.1.1.1</td>
</tr>
<tr>
<td>MPEG2_MPHL_SYNTAX</td>
<td>This value defines the UID for MPEG2 Main Profile @ High Level transfer syntax. DEFAULT: 1.2.840.10008.1.2.4.101</td>
</tr>
<tr>
<td>MPEG2_MPML_SYNTAX</td>
<td>This value defines the UID for MPEG2 Main Profile @ Main Level transfer syntax. DEFAULT: 1.2.840.10008.1.2.4.100</td>
</tr>
<tr>
<td>MPEG4_AVC_H264_BDC_HP_LEVEL_4_1_SYNTAX</td>
<td>This value defines the UID for MPEG-4 AVC/H.264 BD-compatible High Profile / Level 4.1 transfer syntax. DEFAULT: 1.2.840.10008.1.2.4.103</td>
</tr>
<tr>
<td>MPEG4_AVC_H264_HP_LEVEL_4_1_SYNTAX</td>
<td>This value defines the UID for MPEG-4 AVC/H.264 High Profile / Level 4.1 transfer syntax. DEFAULT: 1.2.840.10008.1.2.4.102</td>
</tr>
<tr>
<td>MPEG4_AVC_H264_HP_LEVEL_4_2_2D_SYNTAX</td>
<td>This value defines the UID for MPEG-4 AVC/H.264 High Profile / Level 4.2For 2D Video transfer syntax. DEFAULT: 1.2.840.10008.1.2.4.104</td>
</tr>
<tr>
<td>MPEG4_AVC_H264_HP_LEVEL_4_2_3D_SYNTAX</td>
<td>This value defines the UID for MPEG-4 AVC/H.264 High Profile / Level 4.2For 3D Video transfer syntax. DEFAULT: 1.2.840.10008.1.2.4.105</td>
</tr>
<tr>
<td>MPEG4_AVC_H264_STEREO_HP_LEVEL_4_2_SYNTAX</td>
<td>This value defines the UID for MPEG-4 AVC/H.264 Stereo High Profile /Level 4.2 transfer syntax. DEFAULT: 1.2.840.10008.1.2.4.106</td>
</tr>
<tr>
<td>PDU_MAXIMUM_LENGTH °</td>
<td>The maximum size of Protocol Data Units that can be received by this Merge DICOM Toolkit implementation. This value will also place a limit on how large PDU values being sent can be. Setting this so that a PDU fits within an even multiple of the default TCP/IP MSS (Maximum Segment Size) of 1460 will optimize network performance. Note that 6 bytes for the PDU header must be added to the configured maximum PDU size when calculating a multiple of the MSS. Note also to see the TCPIP_SEND_BUFFER_SIZE and TCPIP_RECEIVE_BUFFER_SIZE configuration values for improving performance. Example: (1460*44)-6 = 64234 PDU Size DEFAULT: 64234 MINIMUM: 4K MAXIMUM: NONE</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PRIVATE_SYNTAX_1_ENCAPSULATED</td>
<td>When set to YES, Merge DICOM Toolkit 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.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: NO</td>
</tr>
<tr>
<td>PRIVATE_SYNTAX_1_EXPLICIT_VR</td>
<td>When set to YES, Merge DICOM Toolkit will interpret private transfer syntax 1 as being encoded in explicit VR format.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: YES</td>
</tr>
<tr>
<td>PRIVATE_SYNTAX_1_LITTLE_ENDIAN</td>
<td>When set to YES, Merge DICOM Toolkit will interpret private transfer syntax 1 as being encoded in little endian format.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: YES</td>
</tr>
<tr>
<td>PRIVATE_SYNTAX_1_SYNTAX</td>
<td>The unique identifier (UID) Merge DICOM Toolkit will use to identify private transfer syntax 1. When this value is set to &quot;&lt;none&gt;&quot;, private transfer syntax support is shut off.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: &lt;none&gt;</td>
</tr>
<tr>
<td>PRIVATE_SYNTAX_2_ENCAPSULATED</td>
<td>When set to YES, Merge DICOM Toolkit 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.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: NO</td>
</tr>
<tr>
<td>PRIVATE_SYNTAX_2_EXPLICIT_VR</td>
<td>When set to YES, Merge DICOM Toolkit will interpret private transfer syntax 2 as being encoded in explicit VR format.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: YES</td>
</tr>
<tr>
<td>PRIVATE_SYNTAX_2_LITTLE_ENDIAN</td>
<td>When set to YES, Merge DICOM Toolkit will interpret private transfer syntax 2 as being encoded in little endian format.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: YES</td>
</tr>
<tr>
<td>PRIVATE_SYNTAX_2_SYNTAX</td>
<td>The unique identifier (UID) Merge DICOM Toolkit will use to identify private transfer syntax 2. When this value is set to &quot;&lt;none&gt;&quot;, private transfer syntax support is shut off.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: &lt;none&gt;</td>
</tr>
<tr>
<td>RLE_SYNTAX</td>
<td>This value defines the UID of the RLE Lossless transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.5</td>
</tr>
<tr>
<td>SMPTE_ST_2110_20_UNCOMPRESSED_INTERLACED_ACTIVE_VIDEO_SYNTAX</td>
<td>This value defines the UID for SMPTE ST 2110-20 Uncompressed Interlaced Active Video transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.7.2</td>
</tr>
<tr>
<td>SMPTE_ST_2110_20_UNCOMPRESSED_PROGRESSIVE_ACTIVE_VIDEO_SYNTAX</td>
<td>This value defines the UID for SMPTE ST 2110-20 Uncompressed Progressive Active Video transfer syntax.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 1.2.840.10008.1.2.7.1</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SMPTE_ST_2110_30_PCM_DIGITAL_AUDIO_SYN TAX</td>
<td>This value defines the UID for SMPTE ST 2110-30 PCM Digital Audio transfer syntax. DEFAULT: 1.2.840.10008.1.2.7.3</td>
</tr>
</tbody>
</table>

† These options allow for non-standard DICOM operations. Such exceptions, if used, should be noted in your DICOM conformance statement.

* Performance tuning.

**Table B-8: [DIMSE_PARMS] section of system profile parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INITIATOR_NAME †</td>
<td>The DICOM standard has retired the old ACR/NEMA Initiator Name attribute in command messages. To generate such an attribute in command messages, specify an initiator name. &lt;none&gt; means do not put initiator name in messages. DEFAULT: &lt;none&gt;</td>
</tr>
<tr>
<td>RECEIVER_NAME †</td>
<td>The DICOM standard has retired the old ACR/NEMA Receiver Name attribute in command messages. To generate such an attribute in command messages, specify a receiver name. &lt;none&gt; means do not put receiver name in messages. DEFAULT: &lt;none&gt;</td>
</tr>
<tr>
<td>SEND_ECHO_PRIORITY †</td>
<td>The DICOM standard has retired the message priority attribute in echo command messages. To generate such an attribute in command messages, specify YES. To NOT use message priority in echo messages, specify NO. DEFAULT: NO</td>
</tr>
<tr>
<td>SEND_LENGTH_TO_END †</td>
<td>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</td>
</tr>
<tr>
<td>SEND_MSG_ID_RESPONSE †</td>
<td>The DICOM standard has retired the message ID attribute in response command messages. To generate such an attribute in command messages, specify YES. To NOT use message ID in response messages, specify NO. DEFAULT: NO</td>
</tr>
<tr>
<td>SEND_RECOGNITION_CODE †</td>
<td>The DICOM standard has retired the old Recognition Code attribute in command messages. To generate such an attribute in command messages, specify YES. If you do not want to generate such an attribute, specify NO. DEFAULT: NO</td>
</tr>
</tbody>
</table>
Table B-9: [DUL_PARMS] section of system profile parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| SEND_RESPONSE_PRIORITY † | The DICOM standard has retired the message priority attribute in response messages. To generate such an attribute in response messages, specify YES. To NOT use message priority in response messages, specify NO.  
 DEFAULT: NO |
| SEND_SOP_CLASS_UID †     | Certain DICOM service classes demand that the affected SOP class UID be present in the message. To prevent the library from ensuring that this is done, specify NO. To ensure that Affected SOP class UID is present, specify YES.  
 DEFAULT: YES |
| SEND_SOP_INSTANCE_UID †  | Certain DICOM service classes demand that the affected SOP instance UID be present in the message. To prevent the library from ensuring that this is done, specify NO. To ensure that Affected SOP instance UID is present, specify YES.  
 DEFAULT: YES |
| ARTIM_TIMEOUT            | 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      | The number of seconds to wait for a reply to an associate request.  
 DEFAULT: 15. |
| CONNECT_TIMEOUT          | The number of seconds to wait for a network connect to be accepted.  
 DEFAULT: 15. |
| INACTIVITY_TIMEOUT       | 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() and MC_Read_To_Stream functions.  
 DEFAULT: 15. |
| INSURE_EVEN_UID_LENGTH † | Set to NO, if odd-length UIDs in PDU’s should NOT be padded with a NULL to ensure even length unique Ids. Set to YES to ensure even length UIDs in PDUs.  
 DEFAULT: NO |
### Configuration Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELEASE_TIMEOUT</td>
<td>The number of seconds to wait for a reply to an associate release.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 15.</td>
</tr>
<tr>
<td>WRITE_TIMEOUT</td>
<td>The number of seconds to wait for a network write to be accepted.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: 15.</td>
</tr>
</tbody>
</table>

† These options allow for non-standard DICOM operations. Such exceptions, if used, should be noted in your DICOM conformance statement.

#### Table B-10: [MEDIA_PARMS] section of system profile parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DICOMDIR_STREAM_STORAGE</td>
<td>When set to yes, DICOMDIRs read in leave their directory records internally in &quot;stream&quot; format and are not parsed until the directory record is referenced. This can greatly reduce memory usage when reading in large DICOMDIRs when the entire DICOMDIR is not referenced. Default: NO</td>
</tr>
<tr>
<td>EXPORT_GROUP_LENGTHS_TO_MEDIA *</td>
<td>When set to NO, do not write group length attributes with MC_Write_File() and MC_Write_File_By_Callback(). DEFAULT: YES</td>
</tr>
<tr>
<td>EXPORT_PRIVATE_ATTRIBUTES_TO_MEDIA</td>
<td>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</td>
</tr>
<tr>
<td>EXPORT_UN_VR_TO_MEDIA</td>
<td>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</td>
</tr>
<tr>
<td>EXPORT_UNDEFINED_LENGTH_SQ_IN_DICOMDIR *</td>
<td>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</td>
</tr>
</tbody>
</table>

* Performance tuning.

#### Table B-11: [MESSAGE_PARMS] section of system profile parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALLOW_COMMA_IN_DS_FL_FDgetStrings</td>
<td>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 a decimal separator. Note that</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ALLOW_INVALID_LENGTH_FOR_VR</td>
<td>When set to 'No', attribute data values with incorrect length according to DICOM, i.e., more than the maximum allowed length or not a multiple of the unit size for fixed length VRs (SS, US, AT, SL, UL, SV, UV, FL, FD) will cause an MC_INVALID_LENGTH_FOR_VR error on DICOM data reading. DEFAULT: YES</td>
</tr>
<tr>
<td>ALLOW_INVALID_PRIVATE_ATTRIBUTES</td>
<td>When reading messages or file objects, this parameter specifies if private attributes encoded in an invalid format should be ignored or parsed. DEFAULT: NO</td>
</tr>
<tr>
<td>ALLOW_INVALID_PRIVATE_CREATOR_CODES</td>
<td>When reading messages or file objects, this parameter specifies if private creator codes encoded with invalid characters should be ignored or parsed. DEFAULT: NO</td>
</tr>
<tr>
<td>ALLOW_OUT_OF_RANGE_BITS_JPEG_LOSSLESS</td>
<td>During decompression of JPEG lossless images, the Pegasus decompressor may discover that the original compressor had failed to mask off the out-of-range bits for the image bit depth. However, if all other lossless JPEG computations are correct, the original image, including such incorrect out-of-range bits, can be losslessly recovered. The Pegasus decompressor will return a warning status, along with the fully decoded image. If this flag is set by the application, the out-of-range bits in output pixels will not be masked off, but returned in the decoded image. Without this flag, out-of-range bits will be masked off to keep pixel values in range. DEFAULT: NO</td>
</tr>
<tr>
<td>ATT_00081190_USE_UT_VR</td>
<td>In the 2014b edition of the DICOM Standard, the value representation of attribute (0008,1190) Retrieve URL was changed from UT to the newly introduced UR. For backward compatibility, this parameter specifies that, when reading messages or file objects, the attribute is expected to have the old UT value representation. DEFAULT: NO</td>
</tr>
<tr>
<td>ATT_00287FE0_USE_UT_VR</td>
<td>In the 2014b edition of the DICOM Standard, the value representation of attribute (0028,7FE0) Pixel Data Provider URL was changed from UT to the newly introduced UR. For backward compatibility, this parameter specifies that, when reading messages or file objects, the attribute is expected to have the old UT value representation.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| ATT_0040E010_USE_UT_VR                    | In the 2014b edition of the DICOM Standard, the value representation of attribute (0040,E010) Retrieve URI was changed from UT to the newly introduced UR. For backward compatibility, this parameter specifies that, when reading messages or file objects, the attribute is expected to have the old UT value representation.  
DEFAULT: NO                                                                                           |
| ATT_0074100A_USE_ST_VR                    | In the 2014b edition of the DICOM Standard, the value representation of attribute (0074,100A) Contact URI was changed from ST to the newly introduced UR. For backward compatibility, this parameter specifies that, when reading messages or file objects, the attribute is expected to have the old ST value representation.  
DEFAULT: NO                                                                                           |
| CALCULATE_DEFINED_LENGTH_FOR_CB           | This parameter is applied when a registered callback function expects the data length to be provided to it and the data length is undefined. If the parameter is set to No, the undefined length will be passed as is to the callback function. If the parameter value is Yes, the toolkit will calculate the actual value of the data length before passing it to the callback function.  
DEFAULT: NO                                                                                           |
| CALLBACK_MIN_DATA_SIZE                    | 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_ALLOW_FRAGS                   | 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.  
DEFAULT: YES                                                                                           |
| COMPRESSION_CHROM_FACTOR                  | Configuration Parameter for MC_Standard_Compressor. Values 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                                                                                             |
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPRESSION_J2K_LOSSY_QUALITY</td>
<td>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. DEFAULT: 1</td>
</tr>
<tr>
<td>COMPRESSION_J2K_LOSSY_RATIO</td>
<td>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. DEFAULT: 10</td>
</tr>
<tr>
<td>COMPRESSION_J2K_LOSSY_USE_QUALITY</td>
<td>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. DEFAULT: YES</td>
</tr>
<tr>
<td>COMPRESSION_LUM_FACTOR</td>
<td>Configuration Parameter for MC_Standard_Compressor. Values 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</td>
</tr>
<tr>
<td>COMPRESSION_RGB_TRANSFORM_FORMAT</td>
<td>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 specify what photometric interpretation Merge DICOM Toolkit should compress into when compressing RGB images. DEFAULT: YBR_FULL_422</td>
</tr>
<tr>
<td>COMPRESSION_USE_HEADER_QUERY</td>
<td>If set to YES, it instructs the toolkit to give precedence to the image parameters (rows, columns, etc.) from the JPEG header, in case disagreement is suspected between the the DICOM header the JPEG header. If set to NO, the DICOM header will be used. DEFAULT: NO</td>
</tr>
<tr>
<td>COMPRESSION_WHEN_J2K_USE_LOSSY</td>
<td>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.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CREATE_OFFSET_TABLE</td>
<td>This parameter specifies if an offset table is created when <code>MC_Duplicate_Message()</code> is used to compress a DICOM message or file. It also specifies if an offset table is created when the <code>MC_Set_Encapsulated_Value_From_Function()</code> and <code>MC_Set_Next_Encapsulated_Value_From_Function()</code> routines are used. DEFAULT: Yes</td>
</tr>
<tr>
<td>DECODER_TAG_FILTER</td>
<td>Specifies the list of tags to be ignored when reading DICOM files or messages. The values are separated by commas and can be specified in different formats:</td>
</tr>
<tr>
<td></td>
<td>• Single tag, e.g.: 00080020</td>
</tr>
<tr>
<td></td>
<td>• Tag range, e.g.: 00080020-000800FF</td>
</tr>
<tr>
<td></td>
<td>• Single group, e.g.: G0020</td>
</tr>
<tr>
<td></td>
<td>• Group range, e.g: G0020-G0022</td>
</tr>
<tr>
<td></td>
<td>• All private as: PRIVATE</td>
</tr>
<tr>
<td></td>
<td>All ranges are inclusive, meaning that G0020-G0022 will filter groups 20 and 22.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: (empty)</td>
</tr>
<tr>
<td>DEFLATE_ALLOW_FLUSH</td>
<td>Allows deflate to flush data occasionally to limit buffering.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: Yes</td>
</tr>
<tr>
<td>DEFLATE_COMPRESSION_LEVEL</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: -1</td>
</tr>
<tr>
<td>DESIRED_LAST_PDU_SIZE</td>
<td>This parameter allows 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.</td>
</tr>
<tr>
<td></td>
<td>Note: Starting with release 3.5.1, this configuration option has a limited effect.</td>
</tr>
<tr>
<td>DICTIONARY_ACCESS</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>Note: Starting with the 3.5.1 Merge DICOM Toolkit release, dictionary access is always memory based and can no longer be file based. This option is now ignored.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT: MEM</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DICTIONARY_FILE</td>
<td>This parameter specifies the name (path) of the DICOM dictionary. An absolute or relative path may be specified. The path to the file can also be specified using environment variables (including the pseudo environment variable MC3INIDIR which does not need to be set as the toolkit will resolve it internally to the directory where the merge.ini file resides). Unicode paths can now be specified through the environment variables. Both Windows style (%) and Unix style ($) notations for the environment variables are accepted. Note: This parameter is ignored if the dictionary has been pre-compiled. DEFAULT: ../mc3msg/mrgcom3.dct</td>
</tr>
<tr>
<td>DUPLICATE_ENCAPSULATED_ICON</td>
<td>When duplicating to an encapsulated transfer syntax, this configuration value specifies whether an ICON IMAGE SEQUENCE should also be encapsulated. DEFAULT: NO</td>
</tr>
<tr>
<td>ELIMINATE_ITEM_REFERENCES *</td>
<td>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: NO.</td>
</tr>
<tr>
<td>EMPTY_PRIVATE_CREATOR_CODES</td>
<td>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</td>
</tr>
<tr>
<td>EXPLICIT_VR_TO_UN_FOR_LENGTH_GT_64K</td>
<td>If set to YES, the toolkit will allow encoding in explicit VR of data elements whose VR is none of OB, OW, OD, OF, SQ or UT and whose value length exceeds 65534 bytes by effectively changing the VR to UN (as per CP-1066). If set to NO, the attempt to encode such data elements will result in an MC_INVALID_LENGTH_FOR_VR error. DEFAULT: NO</td>
</tr>
<tr>
<td>EXPORT_EMPTY_PRIVATE_CREATOR_CODES</td>
<td>If set to NO it prevents the toolkit from exporting private creator data elements which don't have any private attributes in the private block. If set to YES, exporting private creator data elements with empty private blocks is allowed. DEFAULT: YES</td>
</tr>
<tr>
<td>EXPORT_GROUP_LENGTHS_TO_NETWORK *</td>
<td>When set to NO, do not export group length attributes when using the MC_Send_Request_Message(),</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>MC_Send_Request(), MC_Send_Response_Message() and MC_Send_Response() functions</td>
<td><strong>EXPORT_PRIVATE_ATTRIBUTES_TO_NETWORK</strong> When set to NO, disable the exporting of private attributes in messages written to the network with the MC_Send_Request_Message(), MC_Send_Request(), MC_Send_Response_Message() and MC_Send_Response() functions. <strong>DEFAULT: YES</strong></td>
</tr>
<tr>
<td>EXPORT_PRIVATE_ATTRIBUTES_TO_NETWORK</td>
<td>When set to NO, disable the exporting of private attributes in messages written to the network with the MC_Send_Request_Message(), MC_Send_Request(), MC_Send_Response_Message() and MC_Send_Response() functions. <strong>DEFAULT: YES</strong></td>
</tr>
<tr>
<td>EXPORT_UNVR_TO_NETWORK</td>
<td>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(), MC_Send_Request(), MC_Send_Response_Message() and MC_Send_Response() functions. <strong>DEFAULT: YES</strong></td>
</tr>
<tr>
<td>EXPORT_UNDEFINED_LENGTH_SQ *</td>
<td>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. <strong>DEFAULT: NO</strong></td>
</tr>
<tr>
<td>FLATE_GROW_OUTPUT_BUF_SIZE *</td>
<td>The size that the output buffer of deflate or inflate should grow to when its size is insufficient. An Info message is logged each time the buffer grows. <strong>DEFAULT: 1024</strong></td>
</tr>
<tr>
<td>FORCE_OPEN_EMPTY_ITEM *</td>
<td>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. <strong>DEFAULT: NO</strong></td>
</tr>
<tr>
<td>IGNORE_JPEG_BAD_SUFFIX</td>
<td>Configuration 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</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LARGE_DATA_SIZE</td>
<td>Defines “Large Data” to the toolkit. “Large Data” is defined as an attribute value which has a length of LARGE_DATA_SIZE or more. DEFAULT: 200.</td>
</tr>
<tr>
<td>LARGE_DATA_STORE</td>
<td>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</td>
</tr>
<tr>
<td>LIST_SQ_DEPTH_LIMIT</td>
<td>Limit the depth of sequences listing. This parameter should be set to the maximum number of levels any sequence should be listed. DEFAULT: is 0 - means do not limit the listing of sequences</td>
</tr>
<tr>
<td>LIST_UN_ATTRIBUTES</td>
<td>If No, attributes with Unknown VR will not be listed by MC_List_Message() and T2 logging option. DEFAULT: Yes</td>
</tr>
<tr>
<td>LIST_VALUE_LIMIT</td>
<td>Limit the size of listed values by MC_List_Message() or T2 logging option. This parameter should be set to the maximum number of lines to be printed for any attribute in the list. DEFAULT: 0 - means show the whole value.</td>
</tr>
<tr>
<td>MSG_FILE_ITEM_OBJ_TRACE</td>
<td>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</td>
</tr>
<tr>
<td>MSG_INFO_FILE</td>
<td>This parameter specifies the name (path) of the DICOM message information file. An absolute or relative path may be specified. The path to the file can also be specified using environment variables (including the pseudo environment variable MC3INIDIR which does not need to be set as the toolkit will resolve it internally to the directory where the merge.ini file resides). Unicode paths can now be specified through the environment variables. Both Windows style (%) and Unix style ($) notations for the environment variables are accepted. Note: This parameter is ignored if the message database has been pre-compiled. DEFAULT: ../mc3msg/mrgcom3.msg</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>NULL_TYPE3_VALIDATION</td>
<td>This parameter specifies how the toolkit will validate a single NULL value in a type 3 attribute with VM &gt; 1. Valid values are ERR, WARN and INFO. DEFAULT: ERR</td>
</tr>
<tr>
<td>O Bow BUFFER_SIZE</td>
<td>This parameter specifies the number of bytes of &quot;Large Data&quot; that should be buffered before they are written to disk. This value is only used when the parameter LARGE_DATA_STORE is set to FILE. DEFAULT: 4096</td>
</tr>
<tr>
<td>PEGASUS_DISP_REG_NAME</td>
<td>When using your own Pegasus license to remove the 3 frames/second limitation, this should have the company name that was used to generate your Pegasus license.</td>
</tr>
<tr>
<td>PEGASUS_DISP_REGISTRATION</td>
<td>When using your own Pegasus license to remove the 3 frames/second limitation, this should have the registration code that goes with the Pegasus dispatcher.</td>
</tr>
<tr>
<td>PEGASUS_NUMBER_OF_THREADS</td>
<td>Certain Pegasus opcodes can operate in a multithreaded manner. Use this setting to specify the number of threads to be used by the opcode. DEFAULT: 1</td>
</tr>
<tr>
<td>PEGASUS_OP_*_NAME</td>
<td>When using your own Pegasus license to remove the 3 frames/second limitation, this should have the company name that was used to generate your Pegasus license.</td>
</tr>
<tr>
<td>PEGASUS_OP_*_REGISTRATION</td>
<td>When using your own Pegasus license to remove the 3 frames/second limitation, this should have the registration code that goes with its respective PEGASUS_OP_*_NAME.</td>
</tr>
<tr>
<td>PEGASUS_OPCODE_PATH</td>
<td>This parameter specifies the directory where Pegasus opcode DLLs are to be loaded from. The opcode DLL refers to files like picn6220 and not the dispatcher DLL picn20. If the option is empty, the SSM/DLL is loaded from the same directory as the dispatcher DLL. If these files are not found, opocode SSM/DLL is loaded using the directory order Windows uses when loading DLLs. The SSM/DLL is loaded from the current directory if '.' is specified. DEFAULT: (empty)</td>
</tr>
<tr>
<td>RELEASE SQ_ITEMS</td>
<td>If set to NO, existing item IDs will not be freed when setting a null value or an empty value or a new value to a sequence attribute. Setting it to YES will allow sequence items that have no other references to be freed. DEFAULT: No</td>
</tr>
<tr>
<td>REJECT INVALID VR</td>
<td>This parameter specifies whether or not to reject invalid VR values in DICOM messages. If set to Yes, the parsing is aborted and the data set is rejected with a status of MC_INVALID_VR. This is useful in some scenarios when</td>
</tr>
</tbody>
</table>

© Copyright Merge Healthcare Solutions Inc. 2023 519
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID_ATTRIBUTE VR and length can result in runaway read/copy operations which may lead to crashes.</td>
<td>DEFAULT: No</td>
</tr>
<tr>
<td>REMOVE_PADDING_CHARS</td>
<td>When set to Yes, Merge DICOM Toolkit 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</td>
</tr>
<tr>
<td>REMOVE_SINGLE_TRAILING_SPACE</td>
<td>If set to YES, the toolkit will strip a single trailing padding space character from an attribute value of string type. Otherwise it will not.                                                                                                                                                                                                                                           DEFAULT: YES</td>
</tr>
<tr>
<td>RETURN_COMMA_IN_DS_FL_FD_STRINGS</td>
<td>When set to Yes, Merge DICOM Toolkit will return a comma character as a decimal separator 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 decimal separator. Note that DS values will always be properly encoded with a period in DICOM message objects.                                                                 DEFAULT: No</td>
</tr>
<tr>
<td>TEMP_FILE_DIRECTORY</td>
<td>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. The path to the directory can also be specified using environment variables (including the pseudo environment variable MC3INIDIR which does not need to be set as the toolkit will resolve it internally to the directory where the merge.ini file resides). Unicode paths can now be specified through the environment variables. Both Windows style (%) and Unix style ($) notations for the environment variables are accepted. DEFAULT: /</td>
</tr>
<tr>
<td>TOLERATE_INVALID_IN_DEFAULT_CHARSET</td>
<td>This parameter specifies if non-ASCII characters are to be tolerated in the default repertoire. When set to Yes, the validation of the attribute/message will not be enforced, but a warning message will still be logged.                                                                 DEFAULT: Yes</td>
</tr>
<tr>
<td>UN_VR_CODE</td>
<td>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</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>UPDATE_GROUP_0028_ON_DUPLICATE</strong></td>
<td>When set to Yes, the group 0028 attributes within a message will be updated when duplicating a message or file with <code>MC_Duplicate_Message()</code> and the standard compressor or decompressor. The Photometric Interpretation will be updated as appropriate, and the Lossy Image Compression, Lossy Image Compression Ratio and Lossy Image Compression Method tags will be updated if Lossy Image Compression was applied to the image.</td>
</tr>
<tr>
<td><strong>DEFAULT</strong>: No</td>
<td></td>
</tr>
<tr>
<td><strong>USE_FREE_DATA_CALLBACK</strong></td>
<td>When set to Yes, all registered callback functions registered with <code>MC_Register_Callback_Function</code> are called with the FREE_DATA callback type when the memory associated with the callback is to be freed, because the enclosing message, file, or item is being freed.</td>
</tr>
<tr>
<td><strong>DEFAULT</strong>: No</td>
<td></td>
</tr>
<tr>
<td>**WORK_BUFFER_SIZE *</td>
<td>This parameter specifies the amount of data that is buffered in the toolkit before being stored internally or passed to a user’s callback function. This option impacts the <code>MC_Message_To_Stream()</code>, <code>MC_Stream_To_Message()</code>, <code>MC_Send_Request_Message()</code>, <code>MC_Send_Request()</code>, <code>MC_SEND_Response_Message()</code>, <code>MC_SEND_Response()</code>, <code>MC_Read_Message()</code>, <code>MC_Read_To_Stream()</code>, <code>MC_Open_File()</code>, <code>MC_Open_File_Bypass_OBOW()</code>, <code>MC_Open_File_Upto_Tag()</code>, <code>MC_Write_File()</code> and <code>MC_Write_File_By_Callback()</code> functions.</td>
</tr>
<tr>
<td></td>
<td>Setting this option to values larger than 28K will in most cases cause the toolkit to use the operating system’s memory management scheme instead of the toolkit’s internal mechanism.</td>
</tr>
<tr>
<td><strong>DEFAULT</strong>: 28K</td>
<td>* Performance tuning.</td>
</tr>
</tbody>
</table>
### Table B-12: [TRANSPORT_PARMS] section of system profile parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| **CAPTURE_FILE**      | 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** | 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                                                                                                                            |
| **IP_TYPE**           | This parameter specifies the preferred IP type for network communications. When set to IPV4, Merge DICOM Toolkit will attempt to utilize only IPV4 network connections.  
  When set to IPV6, Merge DICOM Toolkit will attempt to use only IPV6 network connections.  
  When set to AVAILABLE in an SCP, Merge DICOM Toolkit will prefer IPV6 if it is enabled in the operating system over IPV4. If IPV6 is used, the socket is put into dual stack mode, if supported by the operating system, to accept connections from both IPV4 and IPV6.  
  When set to AVAILABLE in an SCU, Merge DICOM Toolkit will use the available type of IP networking.  
  DEFAULT: AVAILABLE  
  VALID VALUES: AVAILABLE, IPV4, IPV6                                                                                                   |
| **MAX_PENDING_CONNECTIONS** | This parameter specifies the maximum number of open listen channels. Its value is used as the second argument of a TCP listen() call.  
  DEFAULT: 5                                                                                                                            |
| **NETWORK_CAPTURE**   | 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                                                                                                                           |

**Note:** Use of this parameter is deprecated.
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMBER_OF_CAP_FILES</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Use of this parameter is deprecated.</td>
</tr>
<tr>
<td>REWRITE_CAPTURE_FILES</td>
<td>This parameter specifies whether or not the 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 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.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Use of this parameter is deprecated.</td>
</tr>
<tr>
<td>TCPIP_DISABLE_NAGLE</td>
<td>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.</td>
</tr>
<tr>
<td>TCPIP_LISTEN_PORT</td>
<td>This parameter specifies the TCP/IP port on which server applications are to listen for associate requests.</td>
</tr>
<tr>
<td>TCPIP_RECEIVE_BUFFER_SIZE *</td>
<td>This parameter specifies the TCP/IP receive buffer size for each connection. Note that the maximum values for this constant and TCPIP_SEND_BUFFER_SIZE are operating system dependent. If the values of these options are set too high, a message will be logged to the toolkit's log files, although no errors will be returned through the toolkit's API. Larger values for these constants will greatly improve network performance on networks with minimal network activity. Note that for optimum performance, these values should be at least slightly larger than the PDU_MAXIMUM_LENGTH configuration value. Note also that setting these values to an even multiple of the TCP/IP MSS (Maximum Segment Size) of 1460 bytes can help increase performance. Note, also that some operating systems such as Linux have auto-tuning of TCP/IP buffer sizes implemented when an explicit TCP/IP Send and Receive buffer size are not set. These options can be set to zero to disable Merge DICOM Toolkit's setting of each buffer size.</td>
</tr>
<tr>
<td></td>
<td><strong>MAXIMUM:</strong> Operating System dependent</td>
</tr>
</tbody>
</table>
**Name** | **Description**
--- | ---
**TCPIP_SEND_BUFFER_SIZE** * | This parameter specifies the TCP/IP send buffer size for each connection. Note that the maximum values for this constant and **TCPIP_RECEIVE_BUFFER_SIZE** are operating system dependent. If the values of these options are set too high, a message will be logged to the toolkit's log files, although no errors will be returned through the toolkit's API.

Larger values for these constants will greatly improve network performance on networks with minimal network activity. Note that for optimum performance, these values should be at least slightly larger than the **PDU_MAXIMUM_LENGTH** configuration value.

Note also that setting these values to an even multiple of the TCP/IP MSS (Maximum Segment Size) of 1460 bytes can help increase performance.

Note, also that some operating systems such as Linux have auto-tuning of TCP/IP buffer sizes implemented when an explicit TCP/IP Send and Receive buffer size are not set. These options can be set to zero to disable Merge DICOM Toolkit's setting of each buffer size.

**DEFAULT:** 131400

**MAXIMUM:** Operating System dependent

* Performance tuning.

**Service Profile**

The Service Profile (usually called **mergecom.srv**) contains DICOM standard services and commands and is a useful reference (along with the **message.txt** file) to find the Merge DICOM Toolkit 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 toolkit to create your own private services, you will need to add specifications for these private services to the Service Profile. See the **Merge DICOM Toolkit: Database 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 Merge DICOM Toolkit Profile Database Utilities. Unless you are absolutely confident about changes being made, DO NOT CHANGE THE CONTENTS OF THIS FILE.

The Service Profile contains the following sections:
### Table B-13: Service profile parameters.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[SERVICE_TABLE]</td>
<td>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.</td>
</tr>
<tr>
<td>[&lt;service_number&gt;]</td>
<td>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.</td>
</tr>
<tr>
<td>[ITEM_TABLE]</td>
<td>One item name and number for each DICOM item that can be encoded in an attribute of Value representation SQ (Sequence of Items).</td>
</tr>
</tbody>
</table>