Programmer's Guide to the MSG Facility

A Facility for Interpreting DICOM Messages

Stephen M. Moore

Mallinckrodt Institute of Radiology
Electronic Radiology Laboratory
510 South Kingshighway Boulevard
St. Louis, Missouri 63110
314/362-6965 (Voice)
314/362-6971 (FAX)

Version 2.10.0

August 3, 1998

This manual describes a facility which is used to interpret DICOM Messages. These routines build and parse the COMMAND group of a DICOM Message.

1 Introduction

The MSG facility provides a translation mechanism between DICOM Objects (constructed with the DCM facility) and structures with fixed fields that are easily accessible to the user. This translation facility is provided for the COMMAND group of each of the possible DICOM Messages as defined in Part 7 of the DICOM V3.0 Standard. Any identifiers or other data sets that may be present in a DICOM Message are not interpreted by this facility.

Part 7 of the DICOM Standard defines the parameters that are used to create request and response messages for each of the DIMSE-C and DIMSE-N services. This facility defines a structure for each type of message (request, response) and each type of service. These structures contain fields for each of the attributes in the COMMAND part of the message. These structures also contain fields for any additional data sets that might be present in the request or response message. These additional data sets are maintained as DCM_OBJECTs and are not interpreted by this facility.

Each parameter in a message is classified as mandatory or optional in Part 7 of the Standard. The MSG routines will assume that all mandatory parameters have been provided and will generate errors if any mandatory parameters are missing. Conditional parameters are labeled with an extra bit which is stored in a flag in the structure. When the caller wants the MSG routine to include a conditional parameter in an object, the caller enters the data in the structure and sets the appropriate bit for that parameter. Similarly, if an MSG routine senses that conditional parameter is present on a DICOM object, it will set the appropriate bit in the message structure.

Figure 1 shows the structure definition for the C-STORE request and includes the definition of bits which mark conditional parameters. The bits define as `MSG_K_C_STORE_MOVEMESSAGEID` and `MSG_K_C_STORE_MOVEAETITLE` are set if the parameters moveMessageID and moveAETitle are present in the structure. (These parameters are included when the STORE command is generated in response to a MOVE request.)

```
#define MSG_K_C_STORE_MOVEMESSAGEID     0x01
#define MSG_K_C_STORE_MOVEAETITLE       0x02

typedef struct {
    MSG_TYPE type;
    long conditionalFields;                /* Flag */
    unsigned short messageID;              /* Mandatory */
    unsigned short dataSetType;            /* Mandatory */
    unsigned short priority;               /* Mandatory */
    unsigned short moveMessageID;          /* Optional (if move) */
    DCM_OBJECT *dataSet;                   /* Mandatory */
    char *fileName;                        /* Can replace dataSet */
    char classUID[DICOM_UI_LENGTH + 1];    /* Mandatory */
    char instanceUID[DICOM_UI_LENGTH + 1]; /* Mandatory */
    char moveAETitle[20];                  /* Optional (if move) */
}   MSG_C_STORE_REQ;
```

FIGURE 1. Structure Definition for C-STORE Request

Table 1 shows the correspondence between the DIMSE-C parameter names for the C-STORE service (Part 7, Table 9.1.1-1) and the field definitions above.

TABLE 1. Relation BEtween DIMSE Parameters and Structre Fields

DIMSE-C Parameter Name	MSG Field Name	Comment
	Type	required for MSG
	Conditional Fields	Bit Mask for Conditional Parameters
	Data Set Type	Required for MSG
Message ID	Message ID	Mandatory
Affected SOP Class UID	Class UID	Mandatory
Affected SOP Instance UID	Instance UID	Mandatory
Priority	Priority	Mandatory
Move Originator Application Entity Title	MoveAETitle	Conditional
Move Originator Message ID	Move Message ID	Conditional
Message Set ID		Conditional
End Message Set		Conditional
Data Set	Data Set, FileName	Required for MSG

The MSG routines do not perform any semantic checking. For example, in the C-STORE Request, one would normally include the "Move Originator Message ID" parameter if one included "Move Originator Application Entity Title". The MSG routines do not perform a consistency check, so an application could request that one of the optional parameters be included without the other.

All ASCII data are stored as NULL-terminated ASCII strings. Leading and trailing blanks are stripped as appropriate. Thus, ASCII fields which have been padded for encoding according to Part 5 of the standard will be stripped before the data are passed to the caller.

This facility provides four functions for manipulating the MSG structures. *MSG_BuildObject* reads the data stored in an MSG structure in the caller's memory and creates a DICOM Object (DCM facility) containing the COMMAND data. This object can then be transmitted by one of the DIMSE service routines to a peer application. Once the Object has been used it should be released via *DCM_CloseObject*.

MSG_ParseObject parses a DICOM object which contains data in the COMMAND group. *MSG_ParseObject* can place the output in memory provided by the caller or can allocate memory for the data. If the caller instructs *MSG_ParseObject* to allocate a structure, that structure should be released with *MSG_Free*.

The last function of interest is *MSG_Dump*. This function will examine an MSG structure and print the contents to an ASCII file.

2 Data Structures

This facility defines data structures for each of the messages defined in Part 7 of the Standard. Figure 1 provides an example of the structure defined for the C-STORE request. The other structures are not repeated here and are found in the include file for this facility. The names of the individual fields closely approximate the parameter names for the DIMSE-C and DIMSE-N services.

3 Include Files

To use MSG functions, applications need to include these files in the order given below:

```
#include “dicom.h”
#include “lst.h”
#include “dicom_objects.h”
#include “dicom_messages.h”
```

4 Return Values

The following returns are possible from the MSG facility:

`MSG_NORMAL`	Normal return from MSG routine.
`MSG_PARSEFAILED`	MSG routine failed to parse a DICOM Object.
`MSG_ZEROLENGTHCLASSUID`	Caller’s SOP Class UID string was of zero length.
`MSG_ZEROLENGTHINSTANCEUID`	Caller’s SOP Instance UID string was of zero length.
`MSG_ILLEGALMESSAGETYPE`	MSG facility detected an illegal message. (structure) type. This usually indicates the caller passed the wrong type of structure to an MSG routine or did not properly initialize the type field.
`MSG_NOCOMMANDELEMENT`	MSG routine was unable to extract the command element from an Information Object to determine the type of DICOM Message.
`MSG_UNSUPPORTEDCOMMAND`	MSG routine encountered a DICOM command that is not supported by the facility.
`MSG_MALLOCFAILURE`	MSG routine failed to allocate memory.
`MSG_OBJECTACCESSERROR`	MSG routine failed to access a DICOM Information Object (via DCM facility).
`MSG_OBJECTCREATEFAILED`	MSG routine failed to create a new DICOM Information Object (via DCM facility).

`MSG_MODIFICATIONFAILURE`	MSG routine failed to modify the attribute of a DICOM Information Object.
`MSG_LISTFAILURE`	MSG function encountered an error when trying to access a list (via LST facility).

5 MSG Routines

This section provides detailed documentation for each MSG facility routine.

MSG_BuildCommand

Name

MSG_BuildCommand - build a DICOM Object containing the COMMAND group.

Synopsis

CONDITION MSG_BuildCommand(void *message, DCM_OBJECT **object)

message	Pointer to one of the defined MSG structures supported by this facility. Caller has filled in data fields before calling this function and requests that the function translate the structure into a DICOM Object.
object	Address of caller's pointer to a DICOM Object. The function will create a new object and add the approrpriate elements.

Description

MSG_BuildCommand builds the COMMAND group of a DICOM message. The caller passes the address of one of the defined MSG structures and the address of a DICOM Object. The caller is expected to have filled in the MSG structure with all of the required attributes. If the structure includes any conditional attributes, these are noted with the appropriate flag.

This routine checks the message type against its list of known messages. When the proper message type is found via a table lookup, the DICOM Object is created and populated with the appropriate required and conditional attributes.

Notes

Return Values

```
MSG_NORMAL
MSG_ILLEGALMESSAGETYPE
MSG_OBJECTCREATEFAILED
MSG_MODIFICATIONFAILURE
```

Name

MSG_DumpMessage - dump an ASCII representation of a message structure to a file.

Synopsis

```
void MSG_DumpMesssage(void *message, FILE *f)
```

message	Pointer to message which is to be dumped to a file. A void pointer is specified so it can point to any of the message types.
f	File pointer for an open file. In many instances, this is stderr or stdout.

Description

MSG_DumpMessage dumps an MSG message structure in ASCII to a file which has been opened by the caller. The caller passes a pointer to one of the MSG structures which are defined by this facility. This function looks at the "type" field in the message and calls an appropriate private function which will interpret the data in the structure and dump it to a file. If the function does not recognize the "type" of the message, it returns to the caller with no error indication.

The caller's second arguement is a FILE pointer to an open file. Many times, this is stderr or stdout.

Notes

Return Values

None

MSG_Free

Name

MSG_Free - Free any structure created by the MSG facility.

Synopsis

CONDITION MSG_Free(void **msg)

msg	Address of a pointer to a MSG structure

Description

MSG_Free frees a MSG structure that was allocated by this facility. This requires freeing any lists or DICOM Objects that were created with the structure and freeing the memory for the structure itself.

The caller passes the address of a pointer to a structure.

This function:

- determines the type of a structure
- rees any subelement of the structure as necessary
- frees the structure, and
- writes a NULL into the caller’s pointer

The last step eliminates the caller’s reference to the structure.

Notes

Return Values

```
MSG_NORMAL
MSG_ILLEGALMESSAGETYPE
```

MSG_ParseCommand

Name

MSG_ParseCommand - parse the elements in a DICOM Object and place the data in a fixed structure.

Synopsis

CONDITION MSG_ParseCommand(DCM_OBJECT **object, void ** msg)

object	Address of a DICOM Object pointer which holds the DICOM Command to be parsed.
message	Address of a pointer to a MSG structure. The pointer should be point to pre-existing memory or NULL.

Description

MSG_ParseCommand parses a DICOM Object and places the attribute values in fixed structures. The caller passes a DICOM Object which is assumed to contain a DICOM Command (data in the command group). This function extracts the DICOM Command value from the DICOM Object and performs a table lookup to determine if the command can be parsed by this function. If the command is supported by this function, a private funcation is called which parses the DICOM Object and places the data values in the MSG structure.

The caller's second arguement is the address of a pointer to a MSG structure. If the caller's pointer is NULL, this function allocates a structure of the proper size and writes the address of the structure in the caller's memory. If the pointer is not NULL, the function assumes the caller has allocated memory for the structure.

Notes

The size and type of structure depends on the command value which is extracted from the COMMAND group. If the caller asks this routine to allocate memory for a MSG structure, the structure should be freed with the MSG_Free routine.

Return Values

```
MSG_NORMAL
MSG_NOCOMMANDELEMENT
MSG_UNSUPPORTEDCOMMAND
MSG_MALLOCFAILURE
MSG_PARSEFAILED
MSG_OBJECTACCESSERROR
```