Programmer's Guide to the DMAN Facility

A Facility for Managing and Configuring Applications in a DICOM Environment

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

1 Introduction

The DMAN facility provides access to a number of database tables that are used to manage and configure CTN applications. This facility will serve as a replacement for the CFG facility included with the 1993 CTN software. It is given a different name (not CFG) because it provides an entirely different access mechanism, uses a relational database (Sybase) and is easily extendable as new configuration requirements are identified.

2 Data Structures

manage.h is the public include file for the DMAN facility. It defines a set of structures which correspond to records in a control database.

The table *ApplicationEntity* is used to link applications (by their application title) to a node name. The table also allows a port number to be defined for applications that accept Associations using the TCP/IP stack. The following structure is defined for operations on the *ApplicationEntity* table. Beneath the structure definition are the bit values that can be set in the Flag field.

```
typedef struct {
    void *reserved[2];
    DMAN_DATATYPE Type;
    int Flag;
    char Title[];
    char Node[];
    char Comment[];
int Port;
} DMAN_APPLICATIONENTITY;

DMAN_K_APPLICATION_TITLE
DMAN_K_APPLICATION_NODE
DMAN_K_APPLICATION_COMMENT
DMAN_K_APPLICATION_PORT
```

The table *GroupNames* is used to place external (non-CTN) applications into groups to allow a set of applications to share privileges. This is much like the group names that exist in the Unix file system. Each entry in the *GroupNames* table consists of a tuple which contains the name of a group and the name of one application. The structure used to access the *GroupNames* table and the bits defined for the Flag field in that structure are defined below.

```
typedef struct {
    void *reserved[2];
    int Flag;
    char Title[];
    char GroupName[];
} DMAN_GROUPNAMES;

DMAN_K_GROUP_TITLE
DMAN_K_GROUP_GROUP
```

The *StorageAccess* table is used by image server programs to define how storage areas are accessed and what access rights exist (for the owner, group and other users). The structure used for the *StorageAccess* table and bits defined for the Flag field are defined below.

```
typedef struct {
    void *reserved[2];
    DMAN_DATATYPE Type;
    int Flag;
    S32 Access;
    char Title[];
    char DbKey[];
    char Owner[];
    char GroupName[];
    char Comment[];
} DMAN_STORAGEACCESS;

DMAN_K_STORAGEACCESS_ACCESS
DMAN_K_STORAGEACCESS_TITLE
DMAN_K_STORAGEACCESS_DBKEY
DMAN_K_STORAGEACCESS_OWNER
DMAN_K_STORAGEACCESS_GROUPNAME
DMAN_K_STORAGEACCESS_COMMENT
```

The *StorageControl* table is used to control the storage location when an external (non-CTN) application sends image data to an image server. The structure used for the *StorageControl* table and bits defined for the Flag field are defined below.

```
typedef struct {
    void *reserved[2];
    DMAN_DATATYPE Type;
    char RequestingTitle[];
    char RespondingTitle[];
    char Medium[];
    char Root[];
} DMAN_STORAGECONTROL;

DMAN_K_STORAGECONTROL_REQUESTING
DMAN_K_STORAGECONTROL_RESPONDING
DMAN_K_STORAGECONTROL_MEDIUM
DMAN_K_STORAGECONTROL_ROOT
```

The *SecurityMatrix* table defines which external (non-CTN) applications are allowed to establish Associations with CTN applications. The structure used for the *SecurityMatrix* table and bits defined for the Flag field are defined below.

```
typedef struct {
    void *reserved[2];
    DMAN_DATATYPE Type;
    int Flag;
    char RequestingTitle[];
    char RespondingTitle[];
} DMAN_SECURITYMATRIX;
```

```
DMAN_K_SECURITY_REQUESTING
DMAN_K_SECURITY_RESPONDING
```

The *FISAccess* table is used to define the database tables for different Fake Information Systems and to define access rights for applications that wish to access the FIS's. The structure used for the *FISAccess* table and bits defined for the Flag field are defined below.

```
typedef struct {
    void *reserved[2];
    DMAN_DATATYPE Type;
    int Flag;
    S32 Access;
    char Title[];
    char DbKey[];
    char Owner[];
    char GroupName[];
    char Comment[];
} DMAN_FISACCESS;

DMAN_K_FISACCESS_ACCESS
DMAN_K_FISACCESS_TITLE
DMAN_K_FISACCESS_DBKEY
DMAN_K_FISACCESS_OWNER
DMAN_K_FISACCESS_GROUPNAME
DMAN_K_FISACCESS_COMMENT
```

3 Include Files

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

```
#include "dicom.h"
#include "lst.h"
#include "manage.h"
```

4 Return Values

The following returns are possible from the DMAN facility:

`DMAN_NORMAL`	Normal return from DMAN function.
`DMAN_UNIMPLEMENTED`	This DMAN function is not implemented.
`DMAN_MALLOCFAILED`	A DMAN function failed to malloc memory.
`DMAN_TABLEOPENFAILED`	DMAN_Open failed to open one of the control tables.
`DMAN_APPLICATIONVERIFIATIONFAILED`	Function was unable to verify an applicaton.

DMAN_APPLICATIONNODEMISMATCH	An application was found in the control database, but it is defined for a different node than identified by
DMAN_TITLENOTFOUND	When searching for an application in the control database, this title was not found.
DMAN_ILLEGALCONNECTION	The DMAN facility determined that the connection requested by the caller was illegal (the calling application does not have the right to connect to this CTN application).
DMAN_STORAGEACCESSDENIED	The calling application does not have access privileges for the requested storage area.
DMAN_FILEGENRATIONFAILED	A DMAN function failed to generate a requested file name.
DMAN_FILENAMETOOLONG	A DMAN function generated a file name that is too long for the system (this indicates an algorithm problem in the DMAN facility or that the root used for a storage area was too long.
DMAN_PATHNOTDIR	A file path does not point to a directory as assumed. Make certain that the root value in the StorageControl table corresponds to a directory name.
DMAN_FILECREATEFAILED	Failed to create a new file.
DMAN_APPLICATIONLOOUPFAILED	DMAN_LookupApplication failed to lookup an application as requested.
DMAN_STORAGELOOKUPFAILED	DMAN_LookupStorage failed to lookup a storage access record as requested.

5 DMAN Routines

This section provides detailed documentation for each DMAN facility routine.

DMAN_ApplicationAccess

Name

DMAN_ApplicationAccess - determine if an external application is allowed to access (make an Association with) a CTN application.

Synopsis

CONDITION DMAN_ApplicationAccess(DMAN_HANDLE **handle, char *requestingTitle, char *respondingTitle, BOOLEAN *accessFlag)

handle	The DMAN handle opened by calling DMAN_Open.
requestingTitle	The title of the application that is requesting a connection to a server application.
respondingTitle	The title of the application that is the server application (that is calling this function to validate the access)
accessFlag	Address of a BOOLEAN variable in the caller's space. The flag will be set to TRUE if access is allowed and FALSE otherwise.

Description

DMAN_ApplicationAccess is run by CTN server programs to determine if external (non-CTN) applications are allowed to make Associations. This test is accomplished by examaning the SecurityMatrix table to determine if there is an entry that allows the requested connection to take place.

Notes

Return Values

DMAN_NORMAL

DMAN_ClearList

Name

DMAN_ClearList - clear a list of DMAN structures returned by a call to DMAN_Select

Synopsis

CONDITION DMAN_ClearList(LST_HEAD *l)

l Caller's list pointer.

Description

DMAN_ClearList clears a list of existing DMAN structures by removing the structures from the list and freeing the memory allocated for each structure.

Notes

DMAN_ClearList is called by DMAN_Select before the select function is processed. This means that the user does not need to call DMAN_ClearList before each call to DMAN_Select. The user may wish to call DMAN_ClearList before exiting to free any remaining lists.

Return Values

`DMAN_NORMAL`

DMAN_Close

Name

DMAN_Close - close the open DMAN control tables..

Synopsis

CONDITION DMAN_Close(DMAN_HANDLE **handle)

handle Caller's handle for a set of open control tables.

Description

DMAN_Close closes the set of open control tables and clears memory associated with the caller's handle. DMAN_Close does not free any existing lists of structures that had been returned by DMAN_Select (because it is the caller's responsibility to free the lists through DMAN_ClearList).

Notes

Return Values

DMAN_NORMAL

DMAN_Delete

Name

DMAN_Delete - delete one or more entries from a DMAN control table.

Synopsis

CONDITION DMAN_Delete(DMAN_HANDLE **handle, DMAN_GENERICRECORD *record)

handle The DMAN handle.
record Pointer to a DMAN record in the caller's address space. The record contains data elements that will be used as the criteria in the deletion process. Caller's should set bits in the Flag variable of the record to indicate which elements in the structure should be used to form the delete statement.

Description

DMAN_Delete uses the data in the caller's record to form a delete statement to delete one or more rows from a DMAN control table. The Type field of record will indicate which table is to be modified. The bits that are set in the Flag field of record will determine the criteria for the delete statement.

Notes

DMAN_Delete may not use the data from all attributes passed in record. This is done for efficiency purposes as well as for safety purposes (to keep someone from deleting all of the records when all of the bits are turned off in Flag).

Return Values

DMAN_NORMAL

DMAN_Insert

Name

DMAN_Insert - insert one record into a DMAN control table.

Synopsis

CONDITION DMAN_Insert(DMAN_HANDLE **handle, DMAN_GENERICRECORD *record)

handle	The DMAN handle.
record	Pointer to a DMAN record in the caller's space that will be inserted into a DMAN control table.

Description

DMAN_Insert inserts one record into a DMAN control table. The Type field in record determines which table is to be updated. The caller fills in the data in record and sets the appropriate bits in the Flag field of record. *DMAN_InsertRecord* will attempt to do the insert.

Notes

Many times a failure to complete an insert means that the caller has not included a value for an attribute that the database has marked as "must not be null."

Return Values

DMAN_NORMAL

DMAN_LookupApplication

Name

DMAN_LookupApplication - a convenience function which provides a simple mechanism for retrieving the database record for one application.

Synopsis

```
CONDITION DMAN_LookupApplication(DMAN_HANDLE **handle, char *title,
                DMAN_APPLICATIONENTITY *ae)
```

handle	The DMAN handle.
title	The title of the application entity that is the subject of the lookup perfomed by this function.
ae	Pointer to a structure in the caller's space that will be loaded with the data about this application from the ApplicationEntity table.

Description

DMAN_LookupApplication provides a simple mechanism for an application to retrieve information about an application using the application title as the search criterion.

Notes

The same functionality can be accomplished via *DMAN_Select*, but this mechanism is used often and is easier to use.

Return Values

`DMAN_NORMAL`

DMAN_LookupStorage

Name

DMAN_LookupStorage - a convenience function which provides a simple mechanism for retrieving the storage access record for an application running on a CTN.

Synopsis

CONDITION DMAN_LookupStorage(DMAN_HANDLE **handle, char *applicationTitle,
DMAN_STORAGEACCESS *storage)

handle	The DMAN handle.
applicationTitle	The application title of an application that is running on a CTN.
storage	Pointer to a structure in the caller's space that will be filled in with a record from the StorageAccess table.

Description

DMAN_LookupStorage provides a simple interface for an application to find a record in the StorageAccess control table using applicationTitle as the criterion for the search.

Notes

The same functionality can be accomplished via *DMAN_Select*, but this mechanism is used often and is easier to use.

Return Values

DMAN_NORMAL

DMAN_Open

Name

DMAN_Open - open all of the DMAN control tables.

Synopsis

CONDITION DMAN_Open(char *databaseName, char *requestingTitle, char *respondingTitle,
DMAN_HANDLE **handle)

databaseName	The name of the control database.
requestingTitle	If this function is called by a program that accepts DICOM associations, the name of the requesting application. Otherwise, “”.
respondingTitle	If this function is called by a program that accepts DICOM associations, the name of the responding application. Otherwise, “”.
handle	Pointer to a DMAN handle in the caller’s address space. This function will allocate memory for a private structure and store data in handle to allow other DMAN functions to access the DMAN tables.

Description

DMAN_Open is called by a function to open all of the DMAN control tables. The caller provides the name of a database that contains the control tables. If the application accepts DICOM associations, the caller should also supply the requestingTitle and the respondingTitle. These values are used by other DMAN functions to check access requests.

Upon successful opening of the DMAN control tables, *DMAN_Open* allocates a structure to be used to access the table by other DMAN functions. This sttructure is invisible to the caller, but is referenced through handle.

Notes

The user should call *DMAN_Close* after all access to the DMAN control tables is complete (just before exiting).

Return Values

DMAN_NORMAL

DMAN_PermImageFile

Name

DMAN_PermImage - create a file name for a permanent file (to be stored by an image server)

Synopsis

CONDITION DMAN_PermImageFile(DMAH_HANDLE **handle, char *SOPClass, const char *study, const char* series, char *rtnFileName, size_t fileNameLength)

handle	The DMAN handle.
SOPClass	The SOPClass of the image for which the file name is to be created.
study	A study identifier derived by the caller which can be used to group images from one study into a common directory.
series	A series identifier derived by the callwer which can be used to group images fromone series into a common directory.
rtnFileName	Pointer to storage allocated by the user to hold the file name derived by this function.
fileNameLength	Length of the storage allocated by the caller.

Description

DMAN_PermImageFile is used by the DMAN facility to provide a consistent mechanism for placement and naming of image files. DMAN_PermImage combines inputs from the caller with data stored in the control tables to determine where images should be stored. Typically, this information is taken from the Storage-Control table and is a function of the requesting application and the responding application.

If the call is successful, DMAN_PermImageFile generates a file name that can be used by other functions to open a file for storing data.

Notes

Return Values

DMAN_NORMAL

DMAN_Select

Name

DMAN_Select - select one or more records from a DMAN control table.

Synopsis

CONDITION DMAN_Select(DMAN_HANDLE **handle, DMAN_GENERICRECORD *workRecord,
DMAN_GENERICRECORD *criteriaRecord, LST_HEAD *head,
CONDITION (*callback)(), long *count, void *ctx)

handle	The DMAN handle.
workRecord	Pointer to storage allocated by the caller used during the select process. Each record read by DMAN_Select will be copied into this structure. If DMAN_Select returns one record, the caller can retrieve it from this structure.
criteriaRecord	Pointer to a DMAN record in the caller's space that identifies the search criteria. The caller should fill in data values to be used as criteria and set bits in the Flag field to indicate which data values should be used in the search. Note that the Type fields of workRecord and criteriaRecord are used to determine which control table to search and should be the same.
head	Pointer to a list structure in the caller's space. If provided by the caller, DMAN_Select will place each record found in the control table in this list. If the caller provides NULL, no entries are added to a list (although the caller can retrieve the data through other mechanisms).
callback	User callback function that is called for each record that is retrieved from the control database. Can be NULL. (Currently is not invoked, so don't count on it with this release).
count	Address of a variable in the caller's space that will be updated with the number of records retrieved during the select function.
ctx	A context pointer that can be used by the caller to pass data to the callback function as records are retrieved from the control database. Currently has no effect.

Description

DMAN_Select is used to search one DMAN control table and will return one or more records. The caller supplies a criteriaRecord with data values and bits set in the Flag field to identify which values should be used as search criteria. The caller's workRecord is used as temporary storage for each record that is retrieved from the control table.

If the caller supplies a list head (head), *DMAN_Select* allocates a new structure for each record found in the appropriate control table and places the structure in the caller's list. DMAN_Select clears the caller's list before each select is executed, so the caller does not have to clear the list. If the caller wishes to process each record as it is retrieved, the caller should provide a callback function. Note that the callback function is curretnly not implemented.

Notes

Return Values

DMAN_NORMAL

DMAN_Set

Name

DMAN_Set - update one or more records in a DMAN control table.

Synopsis

CONDITION DMAN_Set(DMAN_handle **handle, DMAN_GENERICRECORD *workRecord,
DMAN_GENERICRECORD *criteriaRecord)

handle	The DMAN handle. workRecordContains one or more data values to be updated in one more more records in a control table.
criteriaRecord	Contains the data values which are used as criteria for determining which records are to be updated in a control table.

Description

DMAN_Set updates one or more records in a DMAN control table. The caller supplies a workRecord which contains the data to be updated. The caller should set appropriate bits in the Flag field of the workRecord to indicate which fields are to be updated. The *criteriaRecord* contains data values (and corresponding bits set in the Flag field) to be used as the criteria during the update function. The caller can control the number of records updated (one, few, many) depending on the criteria used for the update.

Notes

Return Values

DMAN_NORMAL

DMAN_StorageControl

Name

DMAN_StorageControl - a convenience function which provides a simple mechanism for retrieving a record from the StorageControl table.

Synopsis

CONDITION DMAN_StorageControl(DMAN_HANDLE **handle, char *requestingTitle,
char *respondingTitle, DMAN_STORAGECONTROL *control)

handle	The DMAN handle.
requestingTitle	The title of the requesting application.
respondingTitle	The title of the responding application.
control	Pointer to a structure in the caller's space. If the call is successful, it will fill the structure with data from one record in the StorageControl table.

Description

DMAN_StorageControl provides a simple interface for an application to find a record in the StorageControl control table using requestingTitle and respondingTItle as the search criteria.

Notes

The same functionality can be accomplished via DMAN_Select, but this mechanism is used often and is easier to use.

Return Values

DMAN_NORMAL

DMAN_TempImageFile

Name

DMAN_TempImageFile - devise a file name that can be used for temporary storage of a DICOM information object (an image).

Synopsis

CONDITION DMAN_TempImageFile(DMAN_HANDLE **handle, char *SOPClass, char *rtnFileName, size_t fileNameLength)

handle	The DMAN handle. SOPClass The SOP Class of the information object to be stored in the temporary file.
rtnFileName	Storage allocated by the caller to hold the file name devised by this function.
fileNameLength	The length of the storage allocated by the caller.

Description

DMAN_TempImageFile is used by the DMAN facility to provide a consistent mechanism for placement and naming of image files. DMAN_TempImage combines inputs from the caller with data stored in the control tables to determine where images should be stored. Typically, this information is taken from the Storage-Control table and is a function of the requesting application and the responding application.

If the call is successful, *DMAN_TempImageFile* generates a file name that can be used by other functions to open a file for storing data.

Notes

Return Values

DMAN_NORMAL

DMAN_VerifyApplication

Name

DMAN_VerifyApplication - verify that an application that is requesting an association is recognized by the system (is present in the control tables).

Synopsis

CONDITION DMAN_VerifyApplication(DMAN_HANDLE **handle, char *title, char *node)

handle	The DMAN handle.
title	The title of the application for which verification is requested.
node	The name of the node from which the Association request was received.

Description

DMAN_VerifyApplication is used by an application to verify that a requesting application is recognized by the system and that the application is coming from the appropriate node. *DMAN_VerifyApplication* searches the ApplicationEntity control table by title. If a record is found, the function compares the node name passed by the caller with the node name found in the control table. If the node names are equivalent, the application is verified and the function returns DMAN_NORMAL.

If the function fails to verify the application, it returns DMAN_APPLICATIONVERIFICATIONFAILED.

Notes

The comparison of node names is done by looking up the node names in the host table and comparing the nodes names based on the lookup. This means that the caller may pass a fully qualified node name (including domain name) while the database may contain only a host name. The other case should work equally well.

Return Values

```
DMAN_NORMAL
DMAN_APPLICATIONVERIFICATIONFAILED
```