Programmer's Guide to the FIS Facility

A Facility for Manipulating a Fake Information System

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 Fake Information System (FIS) provides some of the functionality that may be present in a Hospital Information System (HIS) or in a Radiology Information System (RIS). We have used the term Fake Information System because there are many functions or features that are not present in the FIS and because we make no claim that this could be used as the basis for an information system. The FIS was designed to provide basic functionality for support of demonstrations of a set of DICOM SOP Classes (patient, study, results, etc., management classes).

This facility implements one or more tables for each SOP classes defined as part of the DICOM Study Management SOP Classes. The tables are listed in Table 1-1 below. The FIS facility provides a general set of functions that can be applied to each table (Insert, Delete, Get, Update). The user of the facility designates a table indirectly by passing structures which have type fields defined. If the user passes a patient record, the FIS knows the operation should be made on the patient table.

This facility does not enforce any of the rules that might be implied by the fact that a DICOM SOP class is a “detached” SOP class. The philosophy of this system is to provide general database functions in this facility and push off the policy functions to another place in the system. For now, we implement those types of policy decisions in applications.

TABLE 1. List of Tables Defined in FIS

Patient	Corresponds to DICOM Detached Patient
Visit	UnimplementedP Class
Study	Corresponds to DICOM Detached Study Management SOP Class
Study Component	Corresponds to DICOM Study Component
Study Component Series	Contains one entry for each series in a study component
Study Component Image	Contains one entry for each image in a study component
Results	Corresponds to DICOM Detached Results Management SOP Class
Interpretation	Corresponds to DICOM Detached Interpretation
Unique Numbers	Used to keep record of unique numbers (not UIDs) assigned by system so that the next unique number can be assigned. Examples are patient ID or accession numbers for studies.

This FIS is built on top of the TBL facility. Therefore, it assumes a relational database as the underlying database mechanism. Because the TBL facility is used to provide database functions, the FIS is not tied to a particular database implementation.

2 Data Structures

fis.h is the primary include file for applications wishing to use the facility. There are several data structures defined which correspond to records in the various tables (patient, study, ...). The structures are listed below.

The FIS structures share a set of features. The first entryvoid *reserved[2]) is used by the list handling functions and are not to be touched by users of this facility. The Type entry will hold one of a set of enumerated types to identify the data in the structure. An example of one of the enumerated types is FIS_K_STUDY. The Flag entry has different uses depending on the function that is invoked. In general, it is used to indicate which data in the structure is valid. For example, when performing an insert, the user sets bits in the Flag to indicate which fields can be inserted into a record. Other fields will be entered as NULL in the database (whatever NULL means for the particular database implementation). Most of the field names are abbreviations of DICOM attribute names and should be fairly obvious. A specific table with attribute names and tags would be helpful, but we did not include that in this version of the software. Some of the fields are of type LST_HEAD*. This means that they contain lists of other structures that are defined in this facility. The first example is a patient record can contain a list of studies (StudyList).

```
typedef struct {
  void *reserved[2];
  FIS_DATA_TYPE Type;
  long Flag;
  char PatID[];
  char PatNam[];
  char PatUID[];
  char PatBirDat[];
  char PatSex[];
  LST_HEAD *StudyList;
  LST_HEAD *VisitList;
  LST_HEAD *PatientAliasList;
} FIS_PATIENTRECORD;

typedef struct {
  void *reserved[2];
  FIS_DATA_TYPE Type;
  long Flag;
  char VisUID[];
  char PatUID[];
  char RefPhyNam[];
  LST_HEAD *StudyList;
} FIS_VISITRECORD;

typedef struct {
  void *reserved[2];
  FIS_DATA_TYPE Type;
  long Flag;
  char PatUID[];
```

```
  char StuInsUID[];
  char VisUID[];
  char AccNum[];
  char StuID[];
  char SchStuStaDat[];
  char SchStuStaTim[];
  char SchStuLoc[];
  char ProDes[];
  char ReqPro[];
  char StuStaID[];
  char SchStuLocAE[];
  StuReaDat[];
  char StuReaTim[];
  LST_HEAD *StudyComponentList;
  LST_HEAD *ResultsList;
} FIS_STUDYRECORD;

typedef struct {
  void *reserved[2];
  FIS_DATA_TYPE Type;
  long Flag;
  char StuComUID[];
  char StuInsUID[];
  char Mod[];
  char StuDes[];
  char ProCodVal[];
  char ProCodSchDes[];
  char ProCodSchDes[];
  char ProCodMea[];
  char StuComStaID[];
  char StuID[];
  LST_HEAD *SeriesList;
} FIS_STUDYCOMPONENTRECORD;

typedef struct {
  void *reserved[2];
  FIS_DATA_TYPE Type;
  long Flag;
  char SerInsUID[];
  char StuComUID[];
  char SerDat[];
  char SerTim[];
  char RetAETit[];
  char StoMedFileSetID[];
  char StoMedFilSetUID[];
  LST_HEAD *ImageList;
} FIS_SCSERIESRECORD;



typedef struct {
  void *reserved[2];
  FIS_DATA_TYPE Type;
```

```
  long Flag;
  char SOPInsUID[];
  char SerInsUID[];
  char StuComUID[];
  char SOPClasUID[];
  char RetAETit[];
  char StoMedFilSetID[];
  char StoMedFilSetUID[];
} FIS_SCIMAGERECORD;

typedef struct {
  void *reserved[2];
  FIS_DATA_TYPE Type;
  long Flag;
  char ResUID[];
  char StuInsUID[];
  char ResID[];
  char Imp[];
  LST_HEAD *InterpretationList;
} FIS_RESULTSRECORD;

typedef struct {
  void *reserved[2];
  FIS_DATA_TYPE Type;
  long Flag;
  char IntUID[];
  char ResUID[];
  char IntID[];
  char IntTex[];
  char IntTypID[];
  char IntStaID[];
} FIS_INTERPRETATIONRECORD;
```

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 “dicom_objects.h”
#include “manage.h”
#include “fis.h”
```

4 Return Values

The following returns are possible from the FIS facility:

`FIS_NORMAL`	Normal return from FIS function.
`FIS_ERROR`	General unspecified error.

FIS_OPENFAILED	FIS_Open failed to open one of the FIS tables.
FIS_MALLOCFAILURE	Failure to malloc memory at runtime.
FIS_CLOSEFAILED	FIS_Close failed to close one of the FIS tables.
FIS_ILLEGALRECORDTYPE	Caller passed a record with an illegal or wrong type.
FIS_NEWRECORDFAILED	The function FIS_NewRecord failed to create a new record.
FIS_INSERTFAILED	Failed to insert a record into the FIS.
FIS_PATIENTINSERTFAILED	Failed to insert a patient record into the FIS.
FIS_GETFAILED	Failed to get a record from the FIS.
FIS_PATIENTGETFAILED	Failed to get a patient record from the FIS.
FIS_DELETEFAILED	Failed to delete a record from the FIS.
FIS_PATIENTDELETEFAILED	Failed to delete a patient record from the FIS.
FIS_NULLSTRING	Application passed a NULL string to FIS function (which was expecting a non-NULL string).
FIS_UPDATEFAILED	Failed to update a record in the FIS.
FIS_PATIENTUPDATEFAILED	Failed to update a patient record in the FIS.
FIS_STUDYGETFAILED	Failed to get study record from FIS.
FIS_STUDYDELETEFAILED	Failed to delete study record from FIS.
FIS_STUDYUPDATEFAILED	Failed to update study record in FIS.
FIS_ACCESSIONNUMBERFAILED	Failed to generate an accession number.
FIS_MISSINGMANDATORYELEMENT	User passed a structure (probably for insert) that was missing a mandatory element (for example, a patient UID in a patient record).
FIS_RESULTSUPDATEFAILED	Failed to update a results record.
FIS_INTERPRETATIONUPDATEFAILED	Failed to update an interpretation record in FIS.
FIS_IDFAILED	Failed to create a new ID.
FIS_UIDFAILED	Failed to create a new UID.
FIS_REQUESTEDATTRIBUTEMISSING	Failed to parse a requested attribute when converting from a DICOM object to an FIS structure.
FIS_ILLEGALDELETECRITERIA	User tried to delete one or more records from FIS using illegal criteria.
FIS_SINGLEGETFAILED	FIS failed to retrieve the single record requested by caller. Perhaps two records matched and the function expected only one.
FIS_SENDEVENTFAILED	Failure occurred when trying to send an event notification to a remote application
FIS_SENDSETFAILED	Failure occurred when trying to send a set request to a remote application.
FIS_PARSEFAILED	Failure occurred when trying to parse a DICOM object (to produce an FIS structure)

5 FIS Routines

This section provides detailed documentation for each FIS facility routine.

FIS_BuildObject

Name

FIS_BuildObject - build a DICOM information object from a FIS structure.

Synopsis

CONDITION FIS_BuildObject(DCM_OBJECT **obj, FIS_DATA_TYPE type,
void *d, FIS_EVENT event)

object	Address of the information object to be created.
type	One of the enumerated FIS_DATA_TYPE's which identifies the structure passed to FIS_BuildObject.
d	Pointer to an FIS structure in the caller's space that contains the data to be used to create the information object.
event	One of the FIS events (e.g., FIS_K_STUDY_SCHEDULED) which will be used by FIS_BuildObject insert the appropriate data elements in the information object.

Description

FIS_BuildObject creates a new DICOM information object by calling DCM_CreateObject with the caller's object argument. Once the information object has been created, FIS_BuildObject examines the structure and event type passed by the caller. FIS_BuildObject adds all of the required data elements for the particular event type as defined in Part 4 of the DICOM standard.

Notes

FIS_BuildObject assumes the data structure has sufficient information to build the appropriate DICOM object. This function is designed to be used by the FIS_SendEvent function and may not be useful to most users of this facility.

Return Values

```
FIS_NORMAL
FIS_ILLEGALRECORDTYPE
FIS_BUILDFAILED
```

FIS_ClearList

Name

FIS_ClearList - clear a list of FIS structures created by FIS_Get

Synopsis

void FIS_ClearList(LST_HEAD *l)

l Caller's LST_HEAD structure which identifies list to be cleared.

Description

FIS_ClearList removes all FIS structures from a caller's list that had been generated from a call to FIS_Get. The memory for each structure is freed, along with memory for any substrctures in the structure. A caller would use this function for final cleanup before exiting.

Notes

See the notes for FIS_Get. The user need not call this function before a call to FIS_Get.

Return Values

none

FIS_Close

Name

FIS_Close - close the connection to the FIS.

Synopsis

CONDITION FIS_Close(FIS_HANDLE **handle)

handle The FIS handle.

Description

FIS_Close closes a user's connection to the FIS by closing all open FIS tables.

Return Values

```
FIS_NORMAL
FIS_CLOSEFAILED
```

FIS_Debug

Name

FIS_Debug - turn debugging on or off.

Synopsis

void FIS_Debug(BOOLEAN flat)

flag Indicates if debugging should be turned on (TRUE) or off (FALSE0.

Description

This function turns debugging mode on or off. When debugging mode is on, extra messages are sent to std-out out during function calls.

Return Values

None

FIS_Delete

Name

FIS_Delete - delete a record from the FIS.

Synopsis

CONDITON FIS_Delete(FIS_HANDLE **handle, FIS_DATA_TYPE type, char *uid);

handle	The FIS handle.
type	One of the enumerated FIS data types. This identifies the type of record to be deleted.
uid	The unique identifier which identifies the record to be deleted.

Description

FIS_Delete deletes one record from the FIS. The caller identifies the type of record and the UID of the record through the type and uid arguments.

Notes

This implies that the user of the facility cannot perform wildcard deletes and can only delete one record at a time.

Return Values

```
FIS_NORMAL
FIS_DELETEFAILED
FIS_NULLSTRING
```

FIS_Get

Name

FIS_Get - Get one or more records from the FIS.

Synopsis

CONDITION FIS_Get(FIS_HANDLE **handle, FIS_DATA_TYPE type,
FIS_DATA_TYPE criteriaType, char *uid, long listFlag, LST_HEAD *getList)

handle	The FIS handle.
type	One of the enumerated FIS data types that defines the type of records to be retrieved from the FIS.
criteriaType	One of the enumerated FIS data types which identifies the type of UID that will be used to search the FIS. For example, one might search the study table by patient UID for a list of studies for a patient or by study UID for a particular study.
uid	A unique identifier that is used as the search criterion.
listFlag	Flag identifies a set of lists that can be requested by the caller for each record that is returned.
getList	Pointer to an existing list in the caller's space that will be loaded with FIS records.

Description

FIS_Get performs a limited set of get (query) function against the FIS. Its initial purpose was to perfom the retrievals that would be needed by FIS servers implementing DICOM N-Get functions. Other FIS applications may find FIS_Get a simple mechanism for retrieving records from the FIS.

FIS_Get searches the FIS using the Unique Identifer (uid) as the search criterion. The caller identifies the type of records to be retrieved (type) and the data type of the UID to be used for the search (criteriaType). For example, FIS_Get supports searches on the patient table only by patient UID's. The study table can be searched for a single study using a study UID or it can be searched for all of the studies for a patient using a patient UID. This section needs a table to define which values are legal for criteriaType, but that is not complete.

listFlag is a flag that is passed by the caller to identify which substructures should be filled in by FIS_Get. For example, when a patient record is retireved, the caller can set a bit in listFlag to request the list of studies to be included. As above, we need a table to identify the legal combinations.

Return Values

```
FIS_NORMAL
FIS_GETFAILED
```

FIS_GetOne

Name

FIS_GetOne - get one record from the FIS.

Synopsis

CONDITION FIS_GetOne(FIS_HANDLE **handle, FIS_DATA_TYPE type,
FIS_DATA_TYPE criteriaType, char *uid, long listFlag, void *record)

handle	The FIS handle.
type	One of the enumerated FIS data types that defines the type of record to be retrieved from the FIS.
criteriaType	One of the enumerated FIS data types which identifies the type of UID that will be used to search the FIS. For example, one might search the study table by patient UID or by study UID.
uid	A unique identifier that is used as the search criterion.
listFlag	Flag identifies a set of lists that can be requested by the caller for each record that is returned.
record	Pointer to an existing structure in the caller's memory space to hold the one record to be returned.

Description

FIS_GetOne searches the FIS using the Unique Identifer (uid) as the search criterion, expecting to find one record that matches. The caller identifies the type of record to be retrieved (type) and the data type of the UID to be used for the search (criteriaType). For example, FIS_GetOne supports searches on the patient table only by patient UID's. The study table can be searched using a study UID or it can be searched using a patient UID. Since the function is expecting to find one record, it does not make much sense to search for one study record by patient UID, but the functionality is provided.

listFlag is a flag that is passed by the caller to identify which substructures should be filled in by FIS_GetOne. For example, when a patient record is retireved, the caller can set a bit in listFlag to request the list of studies to be included.

Return Values

```
FIS_NORMAL
FIS_GETFAILED
FIS_SINGLEGETFAILED
```

FIS_Insert

Name

FIS_Insert - insert one record into the FIS.

Synopsis

CONDITION FIS_Insert(FIS_HANDLE **handle, FIS_DATA_TYPE type, void *record)

handle	The FIS handle
type	Identifies the type of record to be inserted.
record	Pointer to a record passed by the caller to be inserted into the FIS.

Description

FIS_Insert inserts one FIS record into the FIS. The caller is responsible for filling in all required values and providing any optional values. The caller should set a bit in the Flag value of the FIS structure for each value in the structure that contains legal data. The type argument contains one of the FIS enumerated data types (FIS_DATA_TYPE) and identifies the type of record to be inserted. FIS_Insert attempts to insert one record into the FIS and returns a success or failure status.

Notes

FIS records may contain lists of related records. With the current implementation, FIS_Insert does not attempt to insert any of the related records into other tables in the FIS. The type argument is redundant since the record structure contains a Type. This provides one extra level of checking.

Return Values

```
FIS_NORMAL
FIS_INSERTFAILED
```

FIS_NewRecord

Name

FIS_NewRecord - help create a new FIS record by creating necessary unique identifiers.

Synopsis

CONDITION FIS_NewRecord(FIS_HANDLE **handle, FIS_DATA_TYPE type, void *record)

handle	The FIS handle.
type	The data type of the record which is being created.
record	Pointer to a FIS structure in the caller's space that will be updated with unique identifiers.

Description

FIS_NewRecord is called by applications that wish to insert records into the FIS. Each record has one or more unique values; these values are generated by the FIS facility and not by applications. Applications should call FIS_NewRecord to generate the proper unique identifiers for each record before calling FIS_Insert.

Notes

Applications can call FIS_NewRecord before or after they fill in the other data values in the FIS record. Applications are required to fill in the Type field in the record before calling FIS_NewRecord. FIS_NewRecord generates new identifiers by extracting values from an FIS table and by calling the UID facility to generate new DICOM UIDs.

Return Values

```
FIS_NORMAL
FIS_NEWRECORDFAILED
```

FIS_Open

Name

FIS_Open - open a connection to the FIS by opening a series of FIS tables.

Synopsis

CONDITION FIS_Open(char *databaseName, FIS_HANDLE **handle)

databaseName	The name of the database that holds the FIS tables.
handle	Address of a FIS_HANDLE in the caller's space. FIS_Open will allocate memory during the open process and will store a value in the caller's handle.

Description

FIS_Open is the first function that must be called in order to use the FIS. It establishes a link to the FIS by opening a series of FIS tables (patient, study, ...). When the FIS is successfully opened, FIS_Open creates a private data structure and places it in the caller's handle. This handle will be used in calls to other FIS functions.

Notes

The databaseName identifies a particular database. Individual FIS tables have predefined names and must be found in the database. In the current implementation of the underlying relational database, more than one FIS can exist because the database names are different.

Return Values

```
FIS_NORMAL
FIS_OPENFAILED
FIS_MALLOCFAILURE
```

FIS_ParseObject

Name

FIS_ParseObject - parse a DICOM object and place the results in an FIS structure.

Synopsis

CONDITION FIS_ParseObject(DCM_OBJECT **obj, FIS_DATA_TYPE type, void *d)

obj	The caller's DICOM object to be parsed.
type	One of the enumerated types which defines the type of FIS structure should be used as output of the parser.
d	Pointer to an FIS structure in the caller's space which will be the destination for the data parsed from the DICOM object.

Description

FIS_ParseObject parses a DICOM object and places the result in an FIS structure that is allocated by the caller. The caller should already know what kind of object is being parsed and should therefore identify the appropriate FIS data type with the type argument.

Notes

This function provides no verification of "completeness" of an object that has been parsed. It does set the appropriate bits in the Flag field of the structure to indicate which elements were found in the object.

Return Values

```
FIS_NORMAL
FIS_PARSEFAILED
```

FIS_SendEvent

Name

FIS_SendEvent - send a DICOM event notification to a remote application.

Synopsis

CONDITION FIS_SendEvent(FIS_HANDLE **fis, DMAN_HANDLE **dman,
FIS_DATA_TYPE type, void *d, FIS_EVENT event, char *localApplication, char *dst)

fis	The FIS handle.
dman	The DICOM Management handle (see DMAN facility). This handle will be used by the FIS function to map application titles to host addresses.
type	One of the enumerated FIS data types that defines the type of the record used for the event notification.
d	Pointer to a caller's FIS data structure which contains the data for the event notification.
event	Identifies the FIS event that has occurred and that will be transmitted as a DICOM event.
localApplication	The application title of the caller's application. This is used during association negotiation with the remote node.
dst	The application title of the remote node (which is the destination for the event notification).

Description

FIS_SendEvent is used to send one event notification to one or more remote applications. The caller provides an FIS data record and an event type. FIS_SendEvent maps the data and event type into a DICOM information object which will convey the event notification. FIS_SendEvent establishes a DICOM Association with a remote node (identified by dst) and delivers the event notification. After the event notificaiton is sent, the Association is released.

Notes

FIS_SendEvent useds the DMAN facility to map the destination title (dst) to a presentation address that can be used to establish a DICOM Association. When the caller invokes DMAN_Open, the parameters for requestingTitle and respondingTitle can be "". Please refer to the Programmer's Guide to the DMAN Facility. The caller can send event reports to multiple destinations by including multiple application titles in dst separated by ':'. This precludes the use of ':' in an application title for a destination.

If the user tries to send notifications to multiple systems and one notification fails, FIS_SendEvent will not try to send the other notifications. There is no way to determine which destination failed.

Return Values

```
FIS_NORMAL
FIS_SENDEVENTFAILED
```

FIS_SendSet

Name

FIS_SendSet - send a DICOM set request to a remote application.

Synopsis

CONDITION FIS_SendEvent(FIS_HANDLE **fis, DMAN_HANDLE **dman, FIS_DATA_TYPE type, void *d, char *localApplication, char *dst)

fis	The FIS handle.
dman	The DICOM Management handle (see DMAN facility). This handle will be used by the FIS function to map application titles to host addresses.
type	One of the enumerated FIS data types that defines the type of the record used for the set request.
d	Pointer to a caller's FIS data structure which contains the data for the set request..
localApplication	The application title of the caller's application. This is used during association negotiation with the remote node.
dst	The application title of the remote node (which is the destination for the set request).

Description

FIS_SendEvent is used to send one DICOM set request to one or more remote applications. The caller provides an FIS data record with flag bits set indicating which values should be transmitted in the set request. FIS_SendSet maps the data into a DICOM information object which will convey the set request. FIS_SendSet establishes a DICOM Association with a remote node (identified by dst) and delivers the set request. After the request, the Association is released.

Notes

FIS_SendSet useds the DMAN facility to map the destination title (dst) to a presentation address that can be used to establish a DICOM Association. When the caller invokes DMAN_Open, the parameters for requestingTitle and respondingTitle can be "". Please refer to the Programmer's Guide to the DMAN Facility.

The caller can send set requests to multiple destinations by including multiple application titles in dst separated by ':'. This precludes the use of ':' in an application title for a destination. If the user tries to send requests to multiple systems and one request fails, FIS_SendSet will not try to send the other requests. There is no way to determine which destination failed.

Return Values

```
FIS_NORMAL
FIS_SENDSETFAILED
```

FIS_Update

Name

FIS_Update - update one record in the FIS.

Synopsis

CONDITION FIS_Update(FIS_HANDLE **handle, FIS_DATA_TYPE type, void *record)

handle	The FIS handle.
type	Identifies the type of record to be updated.
record	Pointer to a record passed by the caller.

Description

FIS_Update is used to update a single record in the FIS. This record is identified by the UID in the record (for example, the Study UID identifies the study record). The caller can update individual fields in the record by filling in values and by setting bits in the Flag field in the FIS structure. That is, the update function only updates those fields whose corresponding flag bits are set in Flag.

Return Values

```
FIS_NORMAL
FIS_UPDATEFAILED
```