Programmer’s Guide to the SQ Facility

Support Routines which Understand DICOM Sequences of Attributes

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 document describes a general facility for manipulating data elements with a representation of sequence.

1 Introduction

The DICOM V3 Standard defines a number of elements that have a value representation of SQ (or sequence). Any sequence element can have zero, one or more items. Each item in the sequence consists of one or more data elements. These elements can be sequences themselves, leading to a hierarchical structure.

This facility defines a number of structures for “well known” sequences. Each structure contains fixed-length fields for the elements that can be contained in the sequence item. This facility contains functions which allow the user to build and parse sequence elements using these fixed-length structures. The build function takes a list of fixed-length structures and creates a DCM_ELEMENT structure which can be added to a DCM_OBJECT. The parse function takes a DCM_ELEMENT which contains sequence items and builds a list of fixed-length objects.

2 Data Structures

This facility defines a number of fixed-length structures for sequence elements. This section provides an example of one of those structures. The other structures can be found in the include file for this facility.

```
typedef struct {
    void *reserved[2];
    long conditionalFields;
    char referencedSOPClassUID[DICOM_UI_LENGTH+1];
    char referencedSOPInstanceUID[DICOM_UI_LENGTH+1];
} SQ_REFPATIENTSOPINSTANCEUID;
```

This is an example of the fixed-length structure used for the Referenced Patient SOP Instance UID (0008 1120). This sequence contains two attributes, Referenced SOP Class UID (0008 1150) and Referenced SOP Instance UID (0008 1155). The reserved entry in the structure above is reserved for the LST facility. The conditionalFields flag is used for all structures in the SQ facility. It is used to hold bits which indicate optional elements that may be present in the sequence. In this example, both Referenced SOP Class UID and Referenced SOP Instance UID are mandatory, so the flag would not be used.

3 Include Files

Any source code that uses the SQ routines, structure definitions or constants should include the following files in the order given:

```
#include “dicom.h”
#include “condition.h”
#include “lst.h”
#include “dicom_objects.h”
#include “dicom_sq.h”
```

4 Return Values

The following returns are possible from the SQ facility:

SQ_NORMAL	Normal routine from SQ function.
SQ_MALLOCFAILURE	SQ function failed to allocated memory for a structure.
SQ_LISTCREATEFAILURE	SQ function failed to create a new list to hold sequence items.
SQ_MODIFICATIONFAILURE	SQ function failed to modify the values of a DICOM information object.
SQ_LISTFAILURE	SQ function encountered an LST error when accessing a list.

5 SQ Routines

This section provides detailed documentation for each SQ facility routine.

Name

SQ_BuildSequence - build a DCM_ELEMENT with value representation of SQ from a list of fixed-length structures.

Synopsis

CONDITION SQ_BuildSequence(LST_HEAD **list, SQ_TYPE type, DCM_ELEMENT **element)

list	Address of a pointer to a LST_HEAD object which contains a list of homogeneous SQ structures.
type	Defines the type of structures passed to this function in list.
element	Address of a pointer to a DCM_ELEMENT. This function creates a new DCM_ELEMENT and stores the address of the new element at the caller's address.

Description

DCM_BuildSequence builds a DCM_ELEMENT which has a value representation of DCM_SQ. The data field for such an element is a list. This function creates an empty list and stores that with the new DCM_ELEMENT The caller passes a list of structures (each structure having the same type). For each strucutre in the list, *DCM_SQBuildSequence* creates a new DCM_OBJECT and places that object in the list for the new DCM_ELEMENT. When the process is complete, the function returns to the caller.

Notes

The input list can be empty, meaning that the sequence will have zero elements.

Return Values

```
SQ_NORMAL
SQ_MALLOCFAILURE
SQ_LISTCREATEFAILURE
SQ_OBJECTCREATEFAILED
SQ_MODIFICATIONFAILURE
SQ_LISTFAILURE
```

SQ_ParseSequence

Name

SQ_ParseSequence - parse a DCM_ELEMENT containing sequence values into a list of fixed-length strucutres.

Synopsis

CONDITION SQ_ParseSequence(DCM_ELEMENT *element, SQ_TYPE type, LST_HEAD **list)

element	Pointer to a DCM_ELEMENT in the caller's space which is to be parsed as a sequence.
type	An enumerated value which defines the type of sequence item and structure to be created for each item in the sequence.
list	Address of pointer to a LST_HEAD structure. This function will create a new list and store the LST_HEAD pointer at the caller's address.

Description

SQ_ParseSequence parses a DCM_ELEMENT which contains a sequence and returns a list of fixed-length structures to the caller. The caller has examined the tag of the element and has determined the type of structure to be allocated for each item in the sequence. This function creates a new list and stores the pointer to that list at the address specified by list. SQ_ParseSequence creates one fixed-length structure for each item found in the sequence. The sequence item is parsed and the results placed in the fixed-length structure, and the structure is then added to the list of structures to be returned to the caller.

Notes

Return Values

```
SQ_NORMAL
SQ_LISTCREATEFAILURE
SQ_LISTFAILURE
```