CTN Test Tools

A Guide to Programs for Testing DICOM Functionality

Nilesh Gohel
Stephen M. Moore
Chander Sabharwal

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 contains documentation on various utilities useful for CTN operation and testing.

1 Introduction

This manual describes a set of tools (applications) that we have developed for the purpose of performing tests on DICOM systems and for exercising the systems. The applications that we have developed can be used to drive a DICOM communication session (establish an association), to examine DICOM (image) information objects, and to listen passively to two other systems that are communicating. This manual describes the background and purpose of each application and describes how each application is to be used.

2 DICOM Network Snooper

2.1 Purpose

A DICOM snooper was developed as a tool to passively listen to DICOM communications between two systems on a network. Such a tool may be used to investigate a failure in communications that has occurred at the DICOM Upper Layer level. This failure may be due to an invalid or inappropriate DICOM PDU and or improper handshaking. In a communication between two entities, the DICOM network snooper provides a third party view of the communications by dumping all DICOM elements and association parameters (request and accept or reject). DUL states of the initiator and acceptor are also tracked and reported based on the DICOM PDUs intercepted.

2.2 Scope

The DICOM network snooper is limited in that it can only monitor communications on shared-media (e.g. Ethernet or FDDI) networks where the DICOM communications are based on TCP/IP. The listening node must be located on the network in such a manner that traffic between the nodes being monitored traverses the listening node's network interface. The real-time network snooping facility upon which the applications are based is called the SNP facility and was specifically developed for the Sun Solaris 2.3 (Sun OS 5.3) operating system environment as a tool for monitoring TCP/IP communications. This dependence on the Solaris 2.3 OS is a limitation in terms of environments for the running of the applications. Parameters of the communication that need to be specified are the initiator host name / address, acceptor host name / address, and acceptor's port number. TCP/IP sockets or associations established matching those parameters are then monitored.

The Snoop programs produce output that is intended to be examined by someone with a fair amount of DICOM experience. For example, these programs can print commands and responses generated by two applications, but does not try to interpret the sequence of commands and/or responses. The Snoop programs also perform no tests to determine if command or data sets are complete or otherwise correct.

2.3 Usage

Two applications are used for snooping on communications at the TCP or DICOM Upper Layer level.

2.3.1 snp_to_files

The *snp_to_files* application is used for snooping at the TCP level in real-time. Two output files of the parsed TCP data streams are created. One file is created for data sent by the initiator of the conversation; the second file stores data sent by the acceptor of the conversation. Each TCP segment, once it has been sequenced and acknowledged, is written to the files along with a header that contains the segment's sequence number so that the application reading the files may track the sequence of PDU arrivals between the two files. An end of association dummy header is also written to notify the application reading the files of that event.

More specifically the header attached to each segment placed in the data files looks like that shown in Figure 1.

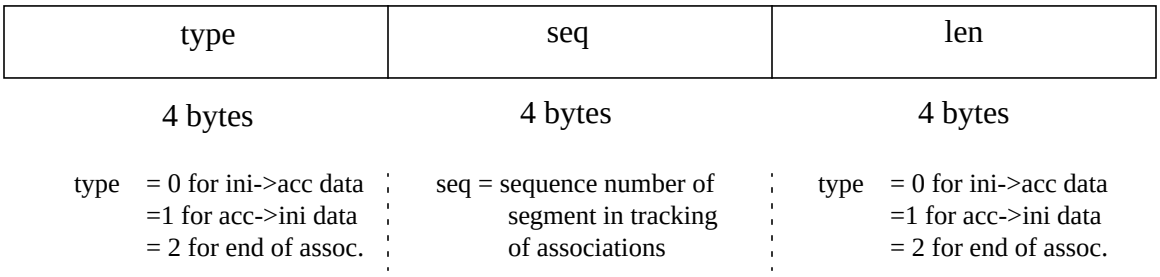

FIGURE 1. Header for TCP Segments Output by *snp_to_files* Application (Big Endian Format)

The application is invoked in the following manner:

snp_to_files device ppa initiator acceptor port buffersize(longs) associations ini->acc_file acc->ini_file

Where

device	Network interface device file, e.g. "/dev/le" for Ethernet
ppa	Physical Point of Attachment (PPA). To identifiy which network interface to use. Generally this will be 0 for the only network interface of this type, e.g. 0 for /dev/le0
initiator	Host name or IP address of initiator
acceptor	Host name of IP addres of acceptor
port	TCP port number of acceptor
buffersize	Size of buffer (in longs) used by underlying SNP facility to buffer data in asynchronous I/O operation. Recommended value is 32768
associations	Number of TCP or DUL associations to be monitored. Each DUL association takes one TCP association
ini->acc_file	Name of file in which to write TCP data stream in the initiator to acceptor direction
acc->ini_file	Name of file in which to write TCP data stream in the acceptor to initiator direction.

2.3.2 dcm_snoop

The *dcm_snoop* application is used for snooping at the DUL level. The application has two operating modes. In the first mode, the snooping is conducted from data files generated by the *snp_to_files* program. The second mode is the real-time operation directly from the network. These data are parsed for DICOM PDUs. Output of dcm_snoop is by default delivered to the terminal. The following information is provided for the associations tracked:

1. Notification of the arrival of a DICOM PDU, including its length (unless standard size) and type. If the type is P-DATA-TF, whether it carries a command or data PresentationData Value (PDV) fragment.
2. Association request parameters.
3. Association response (reject or accept) parameters
4. Dumps of command and data elements once all the PDV fragments for the element have been accumulated. Note that when the data element is of large size, the elements will be held in a file known as for_dcm_snoopITOA or for_dcm_snoopATOI depending on the direction of the data (from initiator to acceptor [ITOA] or from acceptor to initiator [ATOI]). This file is created in the directory from which the application is invoked.
5. On arrival of each DICOM PDU, the anticipated association state numbers for the initiator and acceptor are provided.
6. The end of an association.

The application is invoked in the following manner to use files created by the snp_to_files application:

dcm_snoop ini->acc_file acc->ini_file [initiator] [acceptor]

where the names of the initiator and acceptor are optional and are only used to fill in association parameters.

For invoking the application in a real-time mode, use:

dcm_snoop device ppa initiator acceptor port buffersize(longs) assoc

These parameters are explained above in section 2.3.1.

2.4 Overview of Software Architecture

Figure 2 shows the DICOM network snooper software architecture. For contrast, an example of a generic communications application using the DICOM protocol is provided. In this case, the application uses the DICOM Upper Layer protocol to deliver the commands and data.

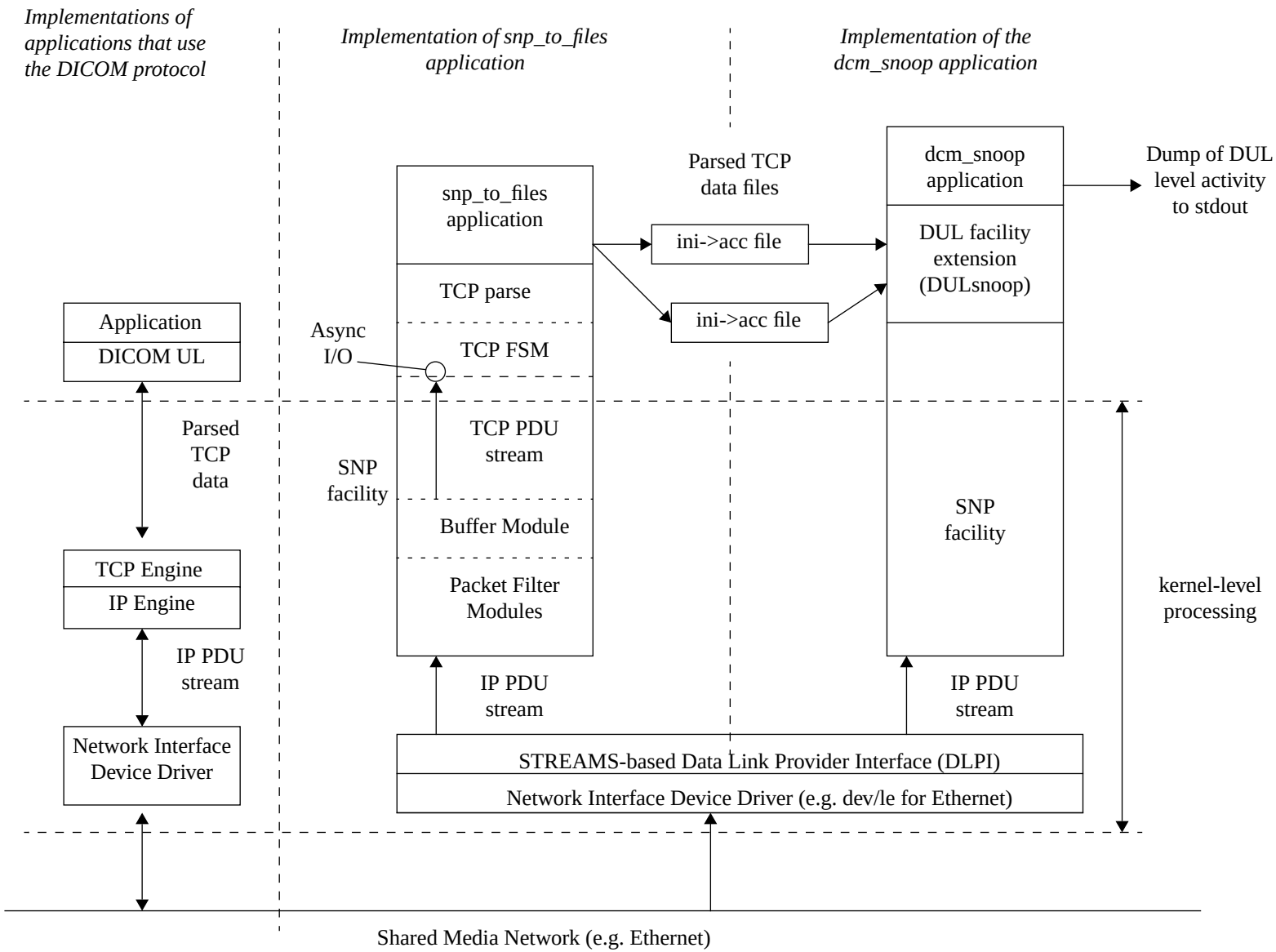

FIGURE 2. Block Diagram of DICOM Network Snooper Software Architecture

The real-time DICOM snooping applications are constructed on top of the SNP facility. This facility was created specifically for snooping on TCP sockets in a Solaris 2.3 environment. It uses the UNIX system communication services known as STREAMS. At the lowest datalink-level, STREAMS are used to communicate with the network interface using Version 2 of the standard Data Link Provider Interface (DLPI). The STREAM is attached and bound to a service access point (SAP). In our case, the SAP is 2048 which corresponds to IP traffic. By placing the interface in promiscuous mode, all IP PDUs on the network are intercepted and not just those for that node.

A chain of packet filters modules are then used on the STREAM at the kernel level. These parse the incoming IP PDUs for only TCP segments that are of interest (are TCP PDUs, have correct IP addresses, and a matching TCP port number). Segments from the initiator to the acceptor are intercepted as well as segments in the opposite direction. These segments are then buffered by another STREAMS module at the kernel level. When the buffer is full or a timer expires, the contents are read by the TCP FSM layer of the software in an interrupt driven asynchronous I/O scheme. The interrupts are serviced by the TCP FSM layer in which each of the TCP segments are reviewed for TCP flags. By monitoring the flags in the TCP segment stream, the SNP facility is able to monitor the states of the TCP socket (or association). The facility is only able to monitor those sockets that are set up while the snooping applications are running. Data or acknowledgments for these sockets are passed on to the next TCP parse layer only if the TCP socket (or association) is perceived to be in an ESTABLISHED state. Although only one association may be monitored at any given time, more than one consecutive association matching the criterion may be monitored using the SNP facility. The SNP facility keeps it's own set of states which allows for the reporting of error and end of association conditions to the next layer.

The TCP parse layer of the software is responsible for resequencing of the TCP segments and parsing the data contained within them. Parsed data is delivered onto the next software layer through callback functions registered with the SNP facility. State information is also passed in the same manner. Data may only passed onto the next software layer once it has been acknowledged as received by the receiving node (ACK is sent back over the network). In TCP, a windowing algorithm is used to transmit data and each byte of data in the transaction is numbered with a sequence number. These numbers are then used in the acknowledgments to indicate how much of the data being transmitted was successfully received. The same numbering scheme is also used for sequencing of segments and in the case of retransmissions, to indicate which data needs to be replaced. Data received is stored using the LST linked list facility awaiting acknowledgments from the receiver. The retransmission scheme allows the SNP facility to forgo checking for checksums; data delivered with an improper checksum will not beacknowledged by the receiver causing a retransmission of the data which replaces the older corrupt version.

The *snp_to_files* and *dcm_snoop* applications were created to use the SNP facility in their real-time operations. The operation of the *snp_to_files* application is described above in section 2.3.1. It is used to dump the TCP parsed data of the associations or sockets monitored to files. The *dcm_snoop* application, whose operations are described above in section 2.3.2, is used for the reporting of activity at the DUL protocol level by parsing the TCP data for DICOM PDUs. This DUL parsing operation is performed by an extension of the DUL facility called DULsnoop. The

TCP data is either supplied to it in files generated by the snp_to_files application or in real-time from the network using callback functions registered to the SNP facility.

The DULsnoop software layer, as a first step, parses the TCP data into DICOM PDUs. These DICOM PDUs are passed on to the next software layer through callback registered with this facility. The DICOM PDUs allow this DUL facility extension to track the DUL protocol states of the associations as well as the association request and response parameters. All this information is provided to the next software layer. As the data for the DICOM PDUs is being accumulated, it is stored in ring buffers.

Finally, the *dcm_snoop* application is used to report the activity at the DUL level as described in section 2.3.2. Command and data element fragments are pieced together and then dumped to the stdout using the DCM facility. Association parameters and DUL protocol state information based on arrival of the DICOM PDUs are also dumped to the stdout.

2.5 Parameterization

The *snp_to_files* and *dcm_snoop* applications and underlying facilities are CPU and I/O intensive to the point that they may not be able to keep up with high data rate transfers of large data elements (image data) in real-time. The order of the applications in terms of their ability to keep up with rapid data transfers is as follows (that with greatest ability first):

1. snp_to_files
   - Partition for output files is local to machine
   - Partition for output files is remote (NFS mounted)
2. dcm_snoop
   - With stdout > to a file on a local partition
   - With stdout > to a file on a remote partition (NFS mounted)
   - With stdout to terminal

The applications have only been tested with the Ethernet interface, /dev/le. Some parameterization tests were run with the snooping applications on a Sun SPARCstation 10-40 and network activity measured using an HP Network Protocol Analyzer (4972A). It was found that on data transfers with peaks of 50 % network utilization (~5.0 Mbps), none of the applications were able to keep up. This was the case for large image file transfers using the faster non-verbose mode of the receiving application. When the verbose mode of the receiving applications was used, all the real-time snooping applications were found to keep up. The peak network utilization in this case was found to be 18 % (~1.8 Mbps). Note that in these cases, the residual traffic on the network seemed to be independent of the results. The measurements for the residual traffic on the network when the above measurements were made was 3-5 % (~ 0.3 - 0.5 Mbps).

The above results indicate that, in the case of large data element transfers, it is advisable to place the sending or receiving applications into a “verbose” mode in order to ensure proper monitoring of the communications.

3 Association Tool

3.1 Purpose

association_tool is an application with a Motif GUI that is used to construct and transmit association requests. The goal of this application is to provide a tool for developers of DICOM acceptors (not restricted to SCUs or SCPs) to allow them to construct arbitrary association requests and to test their responses. The application provides one screen that allows the user to select from a fixed set of DICOM SOP classes. For each class that is selected, the user is allowed to select the SCU/SCP role that is being proposed and one or more transfer syntaxes.

3.2 Usage

association_tool takes no switches, requires no environment variables (with the exception of the X11 variable DISPLAY) and does not connect to any database. It uses standard X11R5 and Motif function calls and does not require any special X11 environment.

3.3 User Interface

Figure 3 is representative of the user interface that is provided by *association_tool*. The numbered boxes label parts of the user interface and are described in this paragraph. The general theory is that the user selects SOP classes from the list of available SOP classes (1). The user is allowed to select an SOP class multiple times and propose the same or different application context parameters for that SOP class. Each time an SOP class is selected, it appears in the scroll list in the upper, right-hand corner of the user interface (2). The user can select one of the values in that scrolled list to actually modify presentation context parameters. When the user selects an SOP class from (2), association_tool lists that class in the text line labeled (3) for verification. The user can select one SCU/SCP role from the selection box in (4) and one or more transfer syntaxes in the section labeled (5). The Update button (6) is used to actually change the context parameters for an SOP class (make the program retain the change). The Delete button (7) removes an SOP class from the list of selected classes (2). Section (8) is used to specify a set of parameters for initiating an association. The Request (9) and Release (10) buttons are used to explicitly request and release associations.

When a user requests an association, association_tool creates one presentation context item in the association request message for each SOP class selected. As noted above, if the user selects an SOP class twice, association_tool will create distinct presentation context items. The association is requested and user notified of the result of the request. The association is maintained until the user explicitly releases the association or until the user requests another association. If the user requests additional associations, association_tool releases any existing associations.

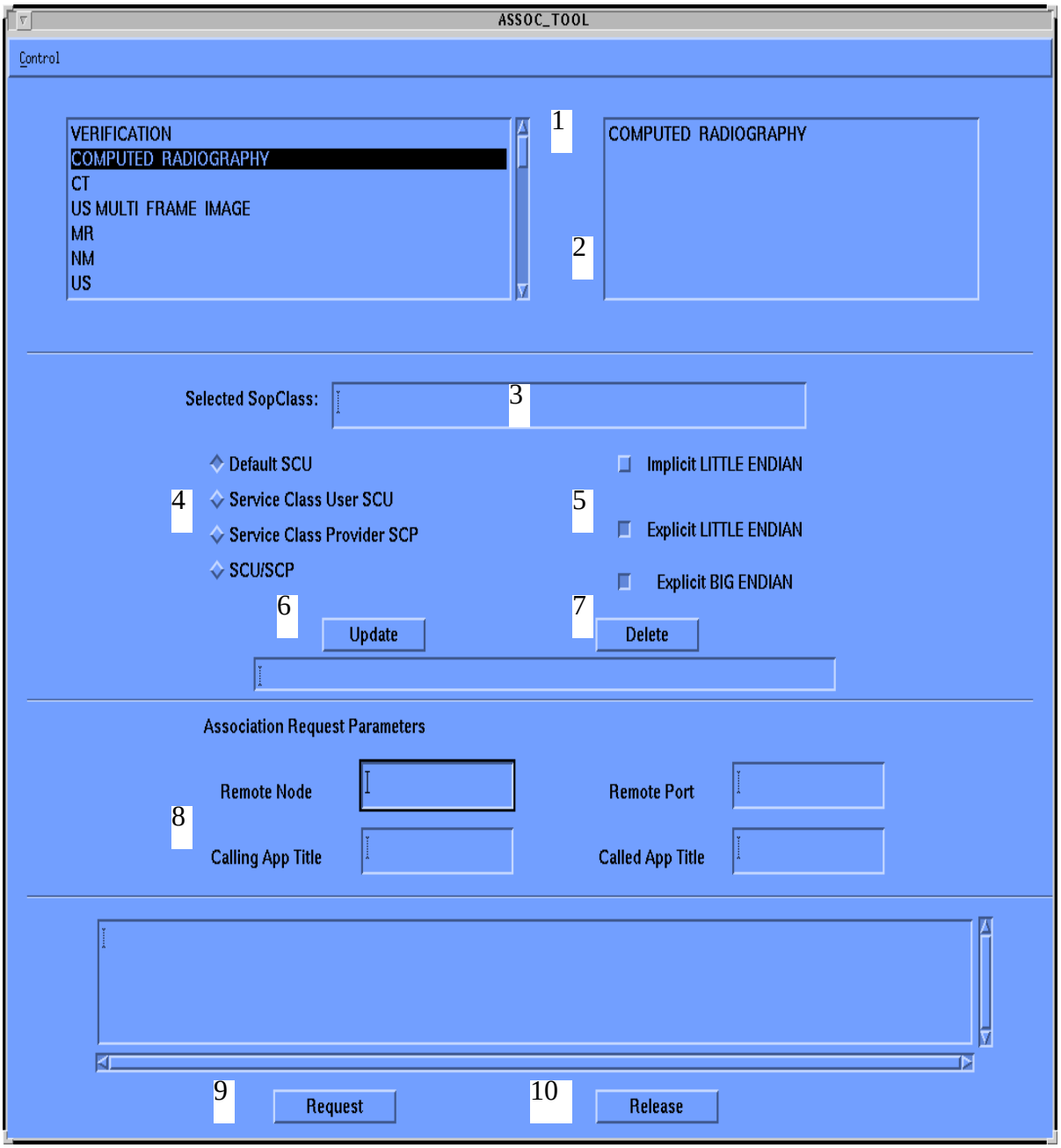

FIGURE 3. Representative User Interface for *association_tool*

4 Object Viewer

4.1 Purpose

Part 3 of the DICOM 3.0 standard defines a number of different information objects. These include normalized as well as composite objects. The definition of image objects (MR, CT, ...) uses a hierarchy that includes information entities, modules and the individual attributes. *object_viewer* is a Motif-based application that provides a simple browser for several image objects. These include CR, CT, MR, (the soon to be jettisoned) NM, US, multi-frame US and SC objects. The user interface description below will indicate how the objects are presented to the user.

4.2 Limitations

There are several natural extensions to this viewer. In this implementation, it does not allow the user to modify any of the attributes. It would also be useful if the program were driven by a simple table that allowed a user to add information objects with recompiling. The current implementation is based on a subroutine library that needs to be augmented for any additional image types. A general viewer that did not depend on the image type but could just present attributes would be another natural extension. That is not provided.

4.3 Usage

object_viewer takes no switches, requires no environment variables (with the exception of the X11 variable DISPLAY) and does not connect to any database. It uses standard X11R5 and Motif function calls and does not require any special X11 environment.

object_viewer is based on the DCM facility which stores files using the implicit little-endian transfer syntax. These files are snapshots of the data as they come across the network in a C-STORE request. They do not correspond to part 10 of the DICOM standard. This is the standard file format supported by all of the other tools in this DICOM package (and should be expanded to include Part 10 files).

4.4 User Interface

Figure 4 is representative of the user interface presented by object_viewer and presents example data for an MR image. The three boxes at the top represent scrolled lists for Information Entities, Modules and Attributes. In this example, the user has selected the Information Entity *Image*, the Module *Image Pixel* and the Attribute *Photometric Interpretation*. The boxes below indicate the actual attribute that was selected and the value of the attribute.

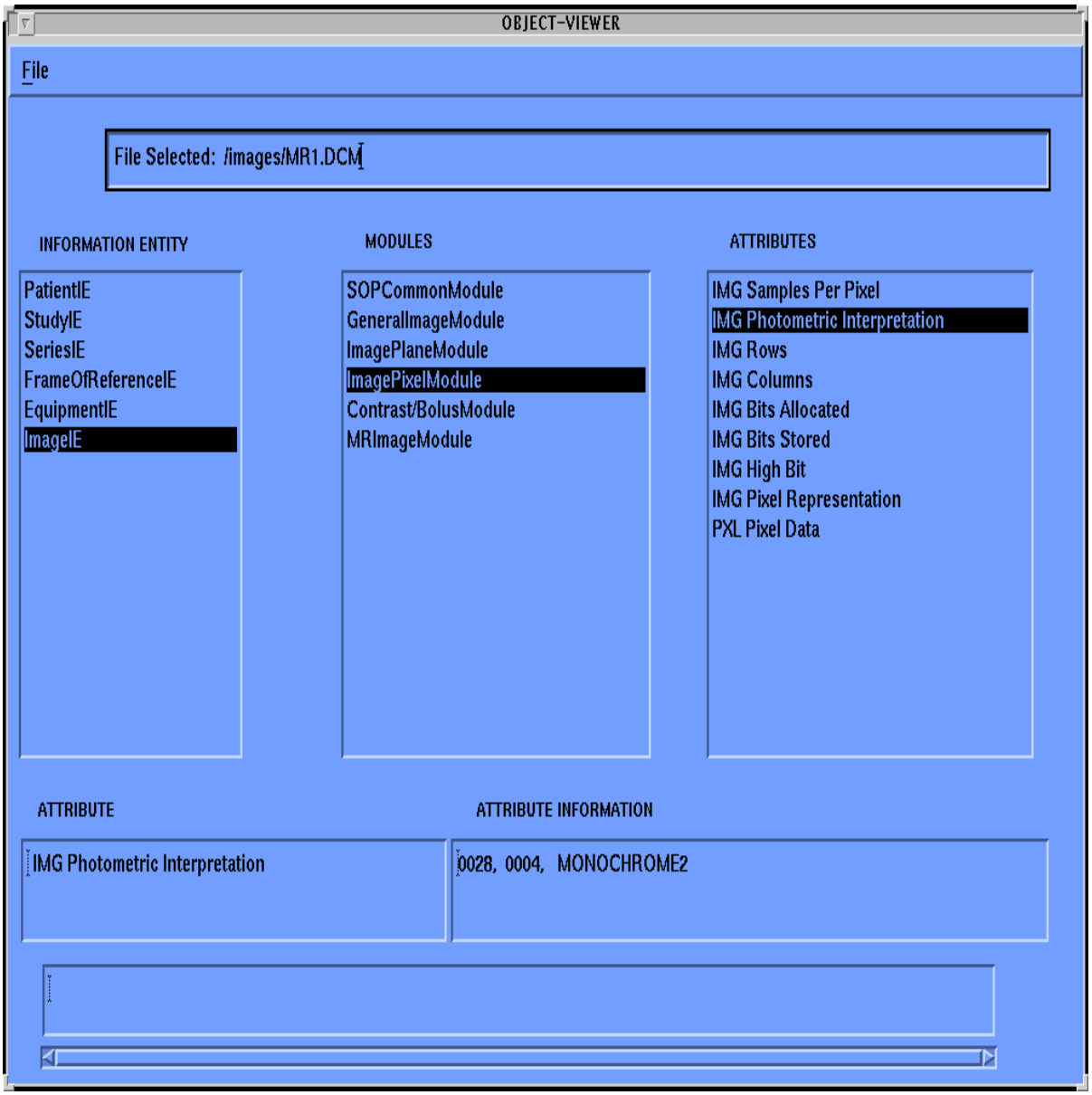

FIGURE 4. Representative User Interface for *object_viewer*