

# **CTN Utility Programs**

## **A Guide to Programs for Testing and Demonstrating DICOM Functionality**

David E. Beecher  
Lon Duan  
Sheldon A. Hoffman  
Stephen M. Moore  
Pei Weng

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

Version 2.10.3  
February 8, 1999

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

Copyright (c) 1998 RSNA, Washington University

# 1 Introduction

This manual describes several utility programs that have proved to be useful during the development of the CTN software. The style adopted for most tests is to write simple, short programs that exercise or demonstrate specific parts of the system. Thus, there is not one general application whose purpose is to exercise the entire system.

Most of the utilities are written to accept a number of command line arguments (and switches). That is, these applications do not prompt you for parameters, nor do they operate in an interactive mode. All of the applications will specify what arguments are needed if they are invoked with no arguments (or an insufficient number of arguments). This short listing of arguments is intended for someone who is already familiar with the application and is not expected to replace the written documentation in this manual.

These documents are not intended to explain the implementation details of the utilities. Many of the details can be inferred from the function of the utility, and the documentation present in the source code for the individual utilities will aid in understanding the design and implementation details.

## 2 DICOM Object Test Programs

This section describes utilities that are useful for examining DICOM Information Objects (images) that are stored in files. This software assumes that Information Objects are stored in files as a stream of bytes that correspond to the DICOM Little-Endian Transfer Syntax. (The file format described in Part 10 of the Standard is not yet implemented.) The programs described in this section can be used to examine the contents of these files. *dcm\_dump\_file* reads a file and prints an ASCII description of each element that is present. *dcm\_dump\_element* extracts a single element and writes the data in binary to another file (very useful for extracting pixel data). *dcm\_verify* examines a file and determines which Information Entities, Modules and Attributes are present per the tables defined in Part 3 of the Standard.

## 3 General Queue Test Programs

The queuing facility was extended for DICOM '93 to accept a user-defined queue element which is defined at run time. As such, only a few utilities exist since this data structure must be correctly defined for whatever queue is being used. However, two utilities exist for manipulating queues. *gqinitq* and *gqkillq* will (respectively) create a new queue and destroy an existing queue. In the `../apps/gq` directory the source to `gqtest.c` shows gives some examples of working with other queue functions.

---

## 4 A Display Utility for DICOM Image Files

*dcm\_x\_disp* is an X-based display utility for DICOM image files. It is fairly simple but useful since it uses exactly the same algorithm for displaying image data as *ctndisp*. Therefore, display problems witnessed with *ctndisp* can be verified with *dcm\_x\_disp* to help find the source of any problems. *dcm\_x\_disp* also dumps out useful information about the image after displaying it.

The program *dcm\_w\_disp* is a display program in the windows environment. It provides slightly less functionality than *dcm\_x\_disp*.

## 5 Print Management Programs

*print\_client* is a program that requests applications with print servers and sends preformatted images to be printed. This client program does not have a GUI and is intended as a mechanism to test the protocol for printing images. *print\_client* is not designed to be a general purpose program for print images. Please refer to the documentation for *print\_manager* found in *User's Guide for CTN Demonstration Applications* for a general purpose print manager with a GUI.

## 6 Example Protocol Data Units

There are several applications that are written to generate association request and response messages and to dump the binary data in hexadecimal (ASCII) to be used as a learning tool. These applications are collectively described on the page labeled *pdus*.

## 7 Availability

The CTN software is supported on Unix and Windows (95, NT) platforms. Not all applications are available on both platforms. Each program description to follow will indicate which platform supports the application.

### Name

dcm\_ctnto10 - Convert a file in “CTN” format to a format that conforms to Part 10 of the DICOM Standard.

### Availability

Unix, Windows

### Synopsis

```
dcm_ctnto10 [-BL] [-v] filein fileout
```

### Description

*dcm\_ctnto10* converts a file from the traditional CTN format to a file in DICOM Part 10 format.. The program assumes the input transfer syntax is the DICOM default Little-Endian transfer syntax. The default output transfer syntax is also DICOM default Little-Endian transfer syntax. The user can specify DICOM explicit big-endian or DICOM little-endian transfer syntax.

The options are:

- B Output file should be written in DICOM explicit big-endian transfer syntax
- L Output file should be written in DICOM explicit little-endian transfer syntax
- v Place DCM facility in verbose mode.

### Notes

We have some problems with our encoding of pixel data with the big-endian transfer syntax, especially 8-bit data. Be wary of the output of this particular program in that regards. It is safest to stick with the little-endian transfer syntaxes for now.

## Name

dcm\_diff - Compare the attributes in two files

## Availability

Unix, Windows

## Synopsis

```
dcm_diff -b -g -l -o -t -v -z file1 file2
```

## Description

*dcm\_diff* reads the user designated files and compares the attribute values. Differences are printed to the standard output. Differences include the cases where an attribute is present in one file and missing in the other or the attribute values are different. This version of the program does not compare pixel data.

The options are:

- b Input files are stored in big-endian byte order
- g Remove group length elements
- l Use (retired) length-to-end attribute for object length
- o Place output in verbose mode
- t Part 10 file
- v Place DCM facility in verbose mode
- z Perform format conversion (verification) on data in files

## Notes

Comparing DICOM files is not a straightforward process. If you send a DICOM image to a storage SCP and then retrieve the image, you may find small differences in the header. For example, the storage SCP may coerce certain attributes (like the patient ID or the study UID). The storage SCP may also remove attributes that are not in its dictionary or may change the format of an attribute (“1.0” may become “1.000”).

This program prints differences to the standard output and lets the user decide if the files are different. We did not feel it was appropriate to set a policy on what constituted a difference that would be considered significant in all applications.

## dcm\_dump\_element

### Name

dcm\_dump\_element - dump one element in binary from a DICOM information object

### Availability

Unix, Windows

### Synopsis

dcm\_dump\_element [-b] [-t] [-v] group element filein fileout

### Description

*dcm\_dump\_element* reads a DICOM information object from a file and extract one data element. The binary data for that element is written to an output file. In the argument list, group and element should be specified in hex. filein and fileout specify the input and output files, respectively.

The options are:

- b Read data from input file assuming big-endian format.
- t Read data from input file assuming DICOM Part 10 format.
- v Verbose mode. Turn on verbose mode for DCM facility.

### Notes

This is the method that we use to extract pixel data from images.

### See Also

*Programmer's Guide to the DCM Facility*

## dcm\_dump\_file

### Name

dcm\_dump\_file - dump the contents of a DICOM V3 file to standard output in a human-readable form.

### Availability

Unix, Windows

### Synopsis

```
dcm_dump_file [-b] [-e] [-g] [-l] [-m mult] [-t] [-v] [-z] file [...]
```

### Description

*dcm\_dump\_file* uses the DCM\_DumpElements function to dump the contents of a DICOM V3 file to standard output in a human-readable form. The information printed for each data element includes:

- Tag (Group and Element Number)
- Data value length
- Short english description
- Some or all of the data

The english description is found in the data dictionary in the DCM package. Data elements not in the dictionary will not have a description.

Some of the data from each element will be printed if the element is found in the DCM data dictionary.

The following options are available:

- b Read data from input files assuming big-endian format.
- e Exit on file open error. Do not process other files. Normal mode is to continue processing other files.
- g Remove group length elements as file is read.
- l Use (retired) length-to-end attribute to find the end of the file.
- m mult For binary attributes with vm > 1, print mult attribute values.
- t Part 10 file
- v Verbose mode. Turn on verbose mode for the DCM facility.
- z Perform format conversion (verification) on data in files.

### See Also

*Programmer's Guide to the DCM Facility*

## dcm\_make\_object

### Name

dcm\_make\_object - make a DICOM information object from an ASCII description

### Availability

Unix, Windows

### Synopsis

dcm\_make\_object [-b] [-p pixels] [-i inputfile] [-v] fileout

### Description

*dcm\_make\_object* is a program that creates a DICOM information object and stores it in a file. *dcm\_make\_object* reads a description of the data elements in the object from stdin and adds them to the information object. The ASCII description of the DICOM object must conform to the following syntax specified in the BNF notation:

```
obj_specification -> element_list
element_list ->      element_list element | element
element ->           group_number element_number value
value ->             numeric_value | alphanumeric_value | quoted_string |
                    tag_value | multiple_values | sequence
tag_value ->         '<group_number \,' element_number '>'
multiple_value ->   '{ value_list }' | '{ tag_list }'
value_list ->        value_list \,' value | value
tag_list ->          tag_list tag_value | tag_value
sequence ->          sequence seq_item | seq_item
seq_item ->          '(' element_list ')'
```

The program stops accepting data when it detects EOF (^D on Unix systems). The *group\_number* and *element\_number* must be specified as hexadecimal numbers. The user can include pixel data from a separate (raw) pixel data file by using the *-p* switch.

### Options

- b        Use Big-Endian Ordering to store the object.
- i inputfile Use input file for the ASCII object description instead of standard input.
- p pixels    Read raw pixel data from the file pixels.
- v        Verbose mode. Place the DCM facility in verbose mode.

The mandatory argument *fileout* stores the DICOM object created by *dcm\_make\_object* from the ASCII description.

## Notes

This program is often used to create objects from descriptions in files. Comment lines in the input should start with two forward slash characters (//) in succession. Anything following the two slashes upto the end of line is considered as a comment. The # character is used to specify a NULL value for a string type element.

## Examples

```
// An example of single valued element
0018 1010 50          // patient's age in kilograms

// An example of multi-valued element
0054 0010 {1,1,1,1,1,2,2,2,2,2} // NM isotope vector

// An example of multi-valued element with quoted strings
0008 0008 {ORIGINAL, PRIMARY, "RECON GATED TOMO", EMISSION} //image type

// An example of a tag type element with value multiplicity 1
0028 0009 <0054, 0010> // Frame increment pointer (single valued)

// An example of a tag type element with value multiplicity 2
0028 0009 { // Frame increment pointer with multiplicity 2
    <0054, 0010>, // index to isotope vector
    <0054, 0020> // index to detector vector
}

// An example of a sequence type element with one item
0008 1110 ( // Referenced Study Sequence
    0008 1150 1.2.840.10008.5.1.4.1.1.20 // SOP class UID
    0008 1155 1.2.840.6839.1993.763.752369842.1 // Instance UID
)

// An example of a sequence type element with multiple items
0054 0062 ( // gated information sequence
    0018 1063 10 // frame time
    0018 1084 4 // intervals rejected
)
(
    0018 1063 12 // frame time
    0018 1084 3 // intervals rejected
)
```

## See Also

*Programmer's Guide to the DCM Facility*

## dcm\_map\_to\_8

### Name

dcm\_map\_to\_8 - map original pixel data (10, 12 bit) to 8 bit

### Availability

Unix, Windows

### Synopsis

dcm\_map\_to\_8 [-b] [v] [-W width] [-C center] input output

### Description

*dcm\_map\_to\_8* reads one DICOM image (monochrome) and maps the pixel data to 8 bits. It does this by applying window width, window center, rescale slope and rescale intercept values found in the information object. The user can override the window width and center values by supplying different values as command line arguments. The following options are available:

- b Read input file assuming the data is stored in big-endian byte order
- v Verbose mode. Place DCM facility in verbose mode.
- W width Override window width with this value.
- C center Override window center with this value.

### Notes

We use this application to map data from modalities for use with our print client software.

## dcm\_modify\_object

### Name

dcm\_modify\_object - modify data elements in an existing DICOM information object

### Availability

Unix, Windows

### Synopsis

dcm\_modify\_object [-b] [-i inputfile] [-p pixels] [-v] filein fileout

### Description

*dcm\_modify\_object* reads an existing DICOM information object from a file and modifies individual data elements based on input read from stdin. The ASCII description of individual elements must conform to the following syntax specified in the BNF notation:

```
modification_specification -> element_list
element_list ->      element_list element | element
element ->          group_number element_number value
value ->            numeric_value | alphanumeric_value | quoted_string |
                   tag_value | multiple_values | sequence
tag_value ->        '<'group_number ',' element_number '>'
multiple_value ->  '{' value_list '}' | '{' tag_list '}'
value_list ->      value_list ',' value | value
tag_list ->        tag_list tag_value | tag_value
sequence ->        sequence seq_item | seq_item
seq_item ->        '(' element_list ')'
```

The program stops accepting data when it detects EOF (^D on Unix systems). The *group\_number* and *element\_number* must be specified as hexadecimal numbers. The user can include pixel data from a separate (raw) pixel data file by using the -p switch.

### Options

-b	Use Big-Endian Ordering to store the object.
-i inputfile	Use input file for the ASCII modification instead of standard input.
-p pixels	Read raw pixel data from the file pixels.
-v	Verbose mode. Place the DCM facility in verbose mode.

The mandatory arguments are:

filein	file that contains original DICOM object.
fileout	file that stores the modified DICOM object.

## Notes

This program is often used to modify objects from descriptions in files. Comment lines in the input should start with two forward slash characters (//) in succession. Anything following the two slashes upto the end of line is considered as a comment. The # character is used to specify a NULL value for a string type element.

## Examples

```
// An example of single valued element
0018 1010 50          // patient's age in kilograms

// An example of multi-valued element
0054 0010 {1,1,1,1,1,2,2,2,2,2}      // NM isotope vector

// An example of multi-valued element with quoted strings
0008 0008 {ORIGINAL, PRIMARY, "RECON GATED TOMO", EMISSION} //image type

// An example of a tag type element with value multiplicity 1
0028 0009 <0054, 0010> // Frame increment pointer (single valued)

// An example of a tag type element with value multiplicity 2
0028 0009 { // Frame increment pointer with multiplicity 2
  <0054, 0010>, // index to isotope vector
  <0054, 0020> // index to detector vector
}

// An example of a sequence type element with one item
0008 1110 ( // Referenced Study Sequence
  0008 1150 1.2.840.10008.5.1.4.1.1.20 // SOP class UID
  0008 1155 1.2.840.6839.1993.763.752369842.1 //Instance UID
)

// An example of a sequence type element with multiple items
0054 0062 ( // gated information sequence
  0018 1063 10 // frame time
  0018 1084 4 // intervals rejected
)
(
  0018 1063 12 // frame time
  0018 1084 3 // intervals rejected
)
```

## **dcm\_print\_dictionary**

### **Name**

dcm\_print\_dictionary - Print the DICOM data dictionary

### **Availability**

Unix, Windows

### **Synopsis**

dcm\_print\_dictionary

### **Description**

*dcm\_print\_dictionary* prints the entries in the DICOM data dictionary that are maintained by the DCM facility. Each group is printed (with group number and a brief title) followed by the entries in each group. Output for each attribute includes the tag, value representation and attribute name.

### **Notes**

The DCM facility does not maintain value multiplicity, so we make no effort to print that.

### Name

dcm\_resize - resize a DICOM image

### Availability

Unix, Windows

### Synopsis

dcm\_resize [-b] [-c cols] [-r rows] [-v] input output

### Description

*dcm\_resize* resizes an existing DICOM image by applying pixel averaging. The user specifies the new size of the image by specifying a different number of rows and/or columns through command line switches.

The following options are available:

- b        Read data from input files assuming big-endian format.
- c cols   Produce new image with cols columns.
- r rows   Produce new image with rows rows.
- v        Verbose mode. Turn on verbose mode for the DCM facility.

### Notes

This will only work for monochrome images.

## **dcm\_rm\_element**

### **Name**

dcm\_rm\_element - remove one data element from a DICOM information object

### **Availability**

Unix, Windows

### **Synopsis**

dcm\_rm\_element [-b] [-v] group element filein fileout

### **Description**

*dcm\_rm\_element* reads the DICOM information object stored in filein, removes the element specified by (group, element) and stores the result in fileout.

The following options are available:

- b Read data from input file assuming big-endian format.
- v Verbose mode. Turn on verbose mode for the DCM facility.

## dcm\_rm\_group

### Name

dcm\_rm\_group - remove a group from a DICOM information object

### Availability

Unix, Windows

### Synopsis

dcm\_rm\_group [-b] [-v] filein fileout group [group ...]

### Description

*dcm\_rm\_group* reads the DICOM information object stored in filein, removes one or more groups as specified by group, [group ...] and writes the result in fileout.

The following options are available:

- b Read data from input file assuming big-endian format.
- v Verbose mode. Turn on verbose mode for the DCM facility.

## dcm\_template

### Name

dcm\_template - dump the description of the required Information Entities, Modules and Attributes to the standard output.

### Availability

Unix

### Synopsis

```
dcm_template [-v] SOPClass [SOPClass]
```

### Description

*dcm\_template* uses functions in the IE facility to dump the required Information Entities, Modules and Attributes for an Information Object in the SOP Class. Only the english description of the Information Entity, Module and Attribute will be printed.

The following option is available:

-v      Verbose mode. Turn on verbose mode for the DCM facility.

### See Also

*Programmer's Guide to the DCM Facility*

*Programmer's Guide to the IE Facility*

*DICOM V3, Part 3*

**Name**

dcm\_verify - dump the contents of a DICOM V3 file, the required Information Entities, Modules and missing Attributes (if any) to the standard output.

**Availability**

Unix, Windows

**Synopsis**

dcm\_verify [-b] [-t] [-v] filename

**Description**

*dcm\_verify* uses functions in the IE facility to examine an Information Object and determine which Information Entities, Modules and Attributes are present and missing (per the rules stated in Part 3 of the DICOM V3 Standard). The information printed for each Information Entity and Module includes:

- Structure type
- Short english description
- Requirement type (mandatory, optional)Status (complete, incomplete or missing)

The information printed for each Attribute includes:

- Tag (group and element number)
- Attribute requirement type
- Short english description
- Value of the data element

The following options are available:

- v Verbose mode. Turn on verbose mode for the DCM facility.
- b Read data from input file assuming big-endian format.
- t Read file with DICOM Part 10 format

**See Also**

*Programmer's Guide to the DCM Facility*  
*Programmer's Guide to the IE Facility*  
*DICOM V3, Part 3*

**Name**

dcm\_x\_disp - display a DICOM image file on an X display.

**Availability**

Unix

**Synopsis**

dcm\_x\_disp [-b] [-t] [-w width][-h height][-W window][-C center] <dicom image file>

**Description**

An X-based display utility used to view a DICOM image file. This utility uses the same image handling algorithms as *ctndisp*, making it useful for testing and debugging. The only required parameter is <dicom image file>. The optional parameters are:

- b use the ORDERBIGENDIAN option when retrieving image data.
- t Image file is in DICOM Part 10 format.
- w make the width of the display “width” pixels instead of a full screen default.
- h make the height of the display “height” pixels instead of a full screen default.
- W force the window for this image to be “window”.
- C force the center or level for this image to be “level”.

**Notes**

*dcm\_x\_disp* prints various statistics about the image to standard output after displaying the image data. In addition, *dcm\_x\_disp* requires certain attributes to be present before an image can be successfully displayed. These required attributes are:

0028,0002	Samples per Pixel	0028,0101	Bits Stored
0028,0010	Rows	0028,0102	High Bit
0028,0011	Columns	0028,0103	Pixel Representation
0028,0100	Bits Allocated	7FE0,0010	Pixel Data

The following optional parameters will be processed if present:

- Photo Interpretation (0028,0004) (MONOCHROME1 or MONOCHROME2 only)
- Window Center (0028,1050)
- Window Width (0028,1051)
- Rescale Slope (0028,1052)
- Rescale Intercept (0028,1053)

## Name

dicom\_echo - create an Association with a server and send one or more C-ECHO requests

## Availability

Unix, Windows

## Synopsis

dicom\_echo [options] node port

## Description

*dicom\_echo* requests an Association with a server running on node *node* and TCP/IP port address *port*. *dicom\_echo* uses a default set of called and calling AE titles, but allows the user to override these with command line arguments. The default mode for the program is to establish an Association and send one C-ECHO request. The caller can request a number of C-ECHO requests be sent by using the *-r* switch.

## Options:

- a title     Use title as the Calling Title in the Association Request
- c called    Use called as the Called Title in the Association Request
- d           Drop association after echo requests
- m mode     Mode for SCU/SCP negotiation (SCU, SCP, SCUSCP)
- n num      Number of network connections requested
- p           Dump service parameters after association request
- r repeat    Send repeat C-ECHO requests (per network/association request)
- v           Place facilities in verbose mode for debug purposes.
- x           Do not release associations when finished with echo

## Notes

A number of other (strange) switches have made their way into this application. These are normally used as a debugging tool to force *dicom\_echo* to do something out of the ordinary so we can observe how the receiving application performs.

**Name**

gqinitq - initialize a new queue.

**Availability**

Unix

**Synopsis**

gqinitq <qid> <queue size> <element size>

**Description**

*gqinitq* creates a new queue with id <qid>, with <queue size> number of elements, where each element is <element size> bytes.

**Notes**

The environment variable `QUEUE_DIRECTORY` must be set before using this utility.

**See Also**

All other gq utilities and the GQ Facility.

**Name**

gqkillq- remove an existing queue from the system.

**Availability**

Unix

**Synopsis**

gqkillq <qid> <element size>

**Description**

*gqkillq* removes the queue from the system whose identifier is <qid> with an element size of <element size>. The element size check is a safety precaution.

**See Also**

All other gq utilities and the GQ Facility.

## kill\_ctndisp

### Name

kill\_ctndisp - remove ctndisp from the system

### Availability

Unix

### Synopsis

kill\_ctndisp <queue\_number>

### Description

*kill\_ctndisp* kills the ctndisp using <queue\_number> from the system. In addition, it removes the queue identified by <queue\_number> from the system. Any resources (semaphores and shared memory) used by ctndisp are returned to the system.

### Notes

Processes still using the queue specified by <queue\_number> may behave unexpectedly after kill\_ctndisp is used.

### See Also

*ctndisp, enq\_ctndisp, pq\_ctndisp*

## object\_viewer

### Name

object\_viewer - a Motif application for navigating DICOM information (image) objects

### Availability

Unix

### Synopsis

object\_viewer

### Description

*object\_viewer* is a simple Motif-based application that allows a user to examine a DICOM image information object. The user can open a file and see the attributes present in the file. Data is broken down according to the groupings that are suggested in Part 3 of the DICOM standard. This means the program presents Information Entities, Modules and Attributes to the user.

### Notes

This program uses the IE facility to break down image information objects. Not all objects are implemented.

### See Also

*Programmer's Guide to the IE Facility*

**Name**

ex1\_initiator  
 ex2\_initiator  
 ex3\_initiatorex3\_acceptor  
 ex4\_initiatorex4\_acceptor

**Availability**

Unix

**Synopsis**

ex1\_initiator [-c calledTitle] [-v] node port  
 ex2\_initiator [-c callingTitle] [-d fac] [-m maxPDU] [-t calledTitle] [-v] node port  
 ex3\_initiator [-c callingTitle] [-d fac] [-m maxPDU] [-t calledTitle] [-v] node port  
 ex4\_initiator [-c callingTitle] [-d fac] [-m maxPDU] [-t calledTitle] [-v] node port  
 ex3\_acceptor [-f] [-p] [-v] port  
 ex4\_acceptor [-a] [-d fac] [-f] [-m maxPDU] [-p] -v

**Description**

These applications are used to generate association request and accept messages and to dump the protocol data units (PDUs) used in those messages in hexadecimal (ASCII) format. These PDU examples correspond to the the examples which are detailed in the document *DICOM Network Examples: Example Protocol Data Units*. The applications are used in pairs. *ex1\_initiator* and *ex2\_initiator* initiate associations and can be used with *simple\_storage*. *ex3\_initiator* should be used with *ex3\_acceptor*; *ex4\_initiator* should be used with *ex4\_acceptor*.

**Options:**

-a callingTitle Abort association during conversation (debugging tool).  
 -c Set the calling AE title to something other than default value.  
 -d fac Place facility fac in verbose mode.  
 -f Forgiving mode. Allow associations with some incorrect parameters in request.  
 -m maxPDU Set the size of the maximum length of the PDV list in a P-DATA PDU.  
 -p Dump association parameters in a human-readable format.  
 -t calledTitle Set the called AE title to something other than the default value.  
 -v Place DCM, DUL and SRV facilities in verbose mode.

The applications generate Association requests which demonstrate the following:

Example 1 Multiple presentation contexts with

Example 2 A single presentation context with multiplier

Example 3 One SOP Class repeated several presentation contexts with a different transfer syntax

Example 4 The Detached Patient Management Meta SOP Class showing SCU/SCP role negotiation.

## **Notes**

You will need to use the `-f` switch on the server applications and the `-v` switch on clients and servers to force the applications to dump the PDUs to the `stderr`.

## print\_client

### Name

print\_client - test print servers by establishing an Association and sending print commands.

### Availability

Unix

### Synopsis

print\_client [-c calledAETitle] [-f films] [-i imageFormat] [-p] [-s] [-t callingAETitle] [-v]  
node port file...

### Description

*print\_client* is designed as a test program for print servers. It establishes an Association with a print server and implements a basic film print session. The user specifies optional arguments to control the session.

### Options:

- c calledAETitle Use calledAPTtitle in the Association request. This should be the AE Title of the print server.
- f films Specify number of films to be printed. Default is 1.
- i imageFormat Parameter to be specified for Image Display Format attribute in Basic Film Box Presentation Module (refer to Part 3, Table C.13.3-1, July, 1993). For example: STANDARD\2\2.
- p Dump Association parameters after Association established (for debugging).
- s Silent mode; do not print results of all print commands.
- t callingAETitle Use callingAETitle in the Association Request as the AE Title of the client.
- v Use verbose mode for DUL and SRV facilities.
- nod The host name that is running a print server.
- port The TCP/IP port number of the print server.
- file The name of one or more files that contain preformatted images to be printed. Each file is opened and used as the preformatted image in the Basic Image Box (set via the N-SET command).

## send\_image

### Name

send\_image - send one or more images to a receiver application by using the DICOM Storage Class.

### Availability

Unix, Windows

### Synopsis

```
send_image [-a application] [-c called] [-m maxPDU] [-p] [-q] [-r] [-s SOPName] [-t] [-x FAC] [-v] node port image [image ...]
```

### Description

*send\_image* is an application that acts as a DICOM Storage SCU for a number of different SOP classes. It establishes one or more DICOM associations with an application that acts as a Storage SCP and sends one or more images using the DICOM Storage SOP Class (C-STORE command). The user supplies a list of one or more images to send to the receiving application. *send\_image* proposes one storage class per association. If it finds an image in the list of a different SOP class than the prior image, *send\_image* closes the current association and initiates a new one.

### Options:

-a application	Set the calling (local) AE title to application. Default is DICOM_TEST.
-c called	Set the called (remote) AE title to called. Default is DICOM_STORAGE.
-m maxPDU	Set my maximum PDU to maxPDU. Default is 16384.
-p	Alter image by sending minimal pixel data.
-q	Place application in quiet mode. Suppresses some messages.
-r	Check the status in the C-Store response message. If status is not success, then stop sending images. Normally, send_image just keeps trying to send images.
-s SOPName	Force send_image to create an association with one SOP class before reading the image in the list. We use the modality abbreviations for SOPName (CR, CT, MR).
-t	Time the image transfer. Print elapsed time and transfer rate.
-x FAC	Place one facility in verbose mode. Values for FAC are DUL, DCM, SRV.
-v	Place DUL and SRV facilities in verbose mode.
node	Name of the host to connect to.
port	TCP/IP port number of remote application.
image	A list of one or more image files to be sent.

The default file format is CTN format. If `send_image` fails to open the file as a CTN file, it will try as a DICOM part 10 file.

### **Notes**

The `-p` switch is used for tests we run when we really only care about the header information and not the pixels. We replace the pixels in the original image with 4 bytes of pixel data. These “smaller” images are useful for testing database operations.

### **See Also**

`simple_storage`

## simple\_storage

### Name

simple\_storage - a storage application that implements some of the storage classes and some of the query/retrieve classes (as an SCP)

### Availability

Unix, Windows

### Synopsis

```
simple_storage [-a] [-d fac] [-i] [-m max] [-n naming] [-p] [-s]
               [-t trips] [-v] [-w] [-z sec] port
```

### Description

*simple\_storage* implements several DICOM storage SOP classes and the DICOM verification SOP class. It communicates with clients that wish to use some of the query/retrieve SOP classes, but it always gives the same response and moves the same images in response to a move request, so it is not very interesting. It is a useful tool for checking the basic steps of Association establishment and image storage.

### Options:

- a Abort the Association after receiving one image (debugging tool)
  - d fac Place fac (DCM, DUL, SRV) in verbose mode.
  - i Ignore some incorrect parameters (like called application title) in Association request.
  - m max Set the maximum received PDU in the Association reply to max (default 16384)
  - n naming Using convention in file *naming* to create directory/file names.
  - p Dump the Association request parameters to stdout.
  - s Silent mode. Don't dump attributes of received images.
  - t trips Execute the main loop trips times (debugging tool).
  - v Place DUL and SRV facilities in verbose mode.
  - x dir Set the base directory for creating images. Default is current directory.
  - w Wait flag. Wait for 1 second in callback routines (debugging).
  - z sec Wait for sec seconds before releasing association
- port TCP/IP port number used to accept incoming network connections

### Notes

simple\_storage assumes its application title is DICOM\_STORAGE. It will reject Association requests without that parameter. You can change that behavior by using the -i switch.

*simple\_storage* is a single threaded application. It supports multiple associations, but sequentially.

*simple\_storage* creates files using one of two naming conventions. The default convention is <modality>/<SOP Instance UID>. For example:

CT/1.2.840.12345.123.456.789

The program allows the user to specify a naming convention using the -n switch. The naming convention file is a series of lines which describe how directories and files are created. Each line in the file describes one level of directory or file creation. You can have multiple directory specifications but only a single file specification. These directives are supported:

A            create a directory with calling AE title  
C            create a directory with called AE title  
D gggg eeee    DEFAULT  
             create a directory with the string taken from (gggg, eeee).  
             If the attribute is empty or missing, using the value DEFAULT  
             for the directory name.  
F gggg eeee DEFAULT  
             create a file with the string taken from (gggg, eeee).  
             If the attribute is empty or missing, use the value DEFAULT  
             for the file name.

The syntax also supports lines that begin with '#' as comments. The characters '^' and '\$' are mapped to '\_'.

Here is a common specification:

A  
D 0008 0050 ACCESSION  
D 0020 000E SERIESINSTANCE  
F 0008 0018 SOPINSTANCE

The attributes (0020, 000E) and (0008, 0018) are DICOM type 1 attributes. Accession number (0008, 0050) is a type 2 attribute and may be empty. In this case, we would have a directory named "ACCESSION" with multiple series in it for images that arrive with no value in that attribute.

I would like to use image number (0020, 0013) for the file specification, but some vendors don't provide good values in that attribute.

## MIR CTN Scripts

This section describes scripts that were created for configuring the demonstration or running the demonstration. These scripts assume (or create) a directory structure that has a single ROOT directory and subdirectories for the various parts of the demonstration. The default ROOT directory used by MIR is /mir\_ctn. You can use this root (create a real directory or a soft link) or you can choose your own root directory. The scripts that create directories or database files ask for the root directory. The scripts that start programs have the root directory defined at the top and do not prompt you for them.

You can use these scripts to setup the CTN demonstration in your lab and can use our default root. There is no requirement that you do so.

### Print Scripts

create_icons	This is the top level script for creating the files needed for the print_mgr application. It invokes a separate script ( icon_script) for each study that is included in the print demonstration. It uses the set of common images found in ROOT/img/db/common. It creates print files and icons in ROOT/print/images and ROOT/print/db. This script assumes that specific subdirectories exist in ROOT/img/db/common and ROOT/print/images for the common images. You should have copied the set of common images to ROOT/img/db/common. print_layout will create the subdirectories needed for the print demonstration.
icon_script	This script is used to populate a print database and an icon database with the information that is used by print_mgr. It is called by create_icons. We do not assume that you will run it.
print_layout	This script is used to create a layout for the print demonstration. It assumes a default ROOT directory of /mir_ctn and prompts the user to allow the user to change that assumption. It creates these subdirectories of ROOT: print, print/config, print/db, print/images. One subdirectory is created in print/images for each study in the set of common images (cr1, cr2, cr3, ct1, ct2, ...). This script does not copy a configuration file into print/config nor does it create the print database and icon files needed by print_mgr.
start_print_client	This script defines the ROOT directory and environment variables and starts the print_client program. The only argument to this script is the vendor name