CTN Software Installation Guide

Guide to Installing Software on UNIX and Windows Computers

Stephen M. Moore

David E. Beecher

Nilesh R. Gohel

Mallinckrodt Institute of Radiology

Electronic Radiology Laboratory

510 South Kingshighway Boulevard

St. Louis, Missouri 63110

314/362-6965 (Voice)

314/362-6971 (FAX)

Version 2.12.0

November 16, 2000

CTN Installation Guide - Version 2.12.0 – November 16, 2000

1 Introduction 2
2 Installation Procedure 3
  2.1 Building the Release from Source Code 3
  2.2 Instructions for Unix Systems 4
  2.3 Instructions for Windows Systems 5
3 Runtime Notes 5
4 Test Procedure 6
  4.1 Examine Images 6
  4.2 Test Network and Snooper Software 7
  4.3 Test Database Routines 9
  4.4 Test Queuing Routines 12
  4.5 Test Demonstration Programs 13
Appendix A: Macros Controlling Build Environment for Unix 14
Appendix B: Platform Specific Notes 17
Appendix C: Database Issues 18
Appendix D: X11/Motif Installation Notes 22

1 Introduction

This manual describes procedures for installing the CTN demonstration software developed by the Electronic Radiology Laboratory (ERL) at the Mallinckrodt Institute of Radiology (MIR). The software is regularly compiled using the Sun Solaris (2.6, 7) operating system and Red Hat Linux (6.0) system. The software has been compiled and used with several other unix versions as well as the Window NT 4.0 operating system.

This manual discusses installation procedure and identifies any machine-dependent issues. This manual also provides a description of the software libraries, applications and other source files present with the system and procedures used to test the installation.

2 Installation Procedure

The software is distributed as source code. From time to time, we also make binary releases for computers and operating systems found in our laboratory. Binary releases are not necessarily made for each source code release.

2.1 Building the Release from Source Code

The software is distributed in both tar and zip formats. You will find these files on an ftp site (ftp.erl.wustl.edu, ftp.rsna.org). The distribution files are usually of the form: ctn-<version>. Untar or unzip the file distribution as appropriate. You should find these directories:

apps	Contains source code for all applications delivered with this system. You will find a number of subdirectories which contain individual applications.
bin	Contains target directories for binary versions of our applications. You will find separate directories under bin for each machine that we support, but you will find no binaries. You may choose to install your binaries in these directories or in a different directory (like /usr/local/bin).
contributed	Contains scripts, instructions, software contributed by users.
cfg_scripts	Contains configuration scripts and data used to setup the databases needed for CTN applications.
environments	Contains short scripts which set up the environments needed to compile and link the software. You will modify these files to fit your system.
facilities	Contains subdirectories with the sources for each of the major subroutine libraries.
include	Contains the common include files defined for each facility.
lib	Contains the target directories for different machine architectures. The software release does not include precompiled libraries.
libsrc	A directory which has links to all facilities. This serves as the build area for the libraries.
runtime	A directory which contains runtime definitions for fonts and queues. Not used in this version of the software.
winctn	Directory with MSVC++ workspace and project definitions.

2.2 Instructions for Unix Systems

2.2.1 Establish Compilation Environment

The first step is establishing your environment. This allows you to :

- Identify target directories
- Define compiler options
- Define machine specific options

The environments directory contains several directories. Each directory is targeted at a specifc operating system (sunos, solaris, osf). In the OS-specific directories are pairs of files that define the compilation environment Examples are:

solaris.2.x.msql.gcc.noopt.env	make.solaris.2.x.msql.gcc.noopt
solaris.2.x.msql.noopt.env	make.solaris.2.x.msql.noopt

The <os.options>.env file contains several environment variables which control the installation. We use the csh, so these variables use the csh syntax. To use these variables, you need to be in the top-level directory of the software and enter this command:

```
source environments/<os>/<environment file>
```

make.os.options is a file which is included by all Makefiles for the purpose of compiling the subroutine libraries (and applications). It defines a global set of rules to be used during the make process. The make environment has two sets of macros. One set of macros is used to define the existence of features in your system. If a macro is defined to the compiler, certain parts of the CTN code will be compiled and operational. A second set of macros defines other switches and constants that control the general system environment. Appendix A describes the macros that are used to build the software.

The best approach is to examine the existing files and to modify them to fit your environment.

2.2.2 Install the software

Previous versions of the software required several steps to install the software. This has now been simplified. From the top level directory, enter this command:

```
make install
```

The Makefiles will build and install the libraries and then build and install the binaries. This procedure does not build the GUI applications which rely on Motif. If you want those applications, install them from the *apps* directory:

```
make gui-install
```

The binaries are installed in $DICOM_BIN.

2.3 Instructions for Windows Systems

The CTN software has been compiled with the MSVC++ 6.0 compiler. One project file is used to control the build process. The build instructions are as follows:

1. The header files are stored with individual libraries and need to be copied to the common include area. Use a DOS window and run these steps:
   - `cd winctn\scripts`
   - `headerexport.bat`

   This is an artifact of how the software was originally developed. If you make modifications to include files, you will need to run this step before rebuilding.
2. Start the MSVC++ development environment. Open the workspace *winctn\winctn.dsw*.
3. Set the active project to *ctn_lib*. Choose *Release* or *Debug* setting as desired and build the library.
4. Set the active project to *ctn_apps*. Choose the *Release* or *Debug* setting and build the applications.
5. Use a DOS window to install the executables. The scripts *copy_release.bat* and *copy_debug.bat* in the *winctn\scripts* directory copy executable files to a target directory. Use *notepad* to modify the batch file and then execute the file to install the applications.

3 Runtime Notes

4 Test Procedure

Once the applications have been installed in the destination directory, there are several tests which should be run to verify that the software is working on your system. The section presents the test procedure in order. You may be able to run these tests in a different order but it is safest to initially run them in the order presented.

4.1 Examine Images

In this section, you will examine the test images and any images that you already have. The test programs distributed with this software assume that image files consist of a stream of bytes that correspond to the DICOM little-endian (implicit) transfer syntax as defined in Part 5 of the Standard. We also have a switch that allows us to examine images that were stored in big-endian format. We are working on support for DICOM Part 10 files and would appreciate any reports of the test programs failing on Part 10 files.

1. Obtain test images from our ftp site or from the distribution CD. At the ftp site (ftp.erl.wustl.edu), these will be in the directory /pub/dicom/images/version3/ctntest. Place the test images in the images directory of the distribution.
2. Run the program dcm_dump_file on one of the images in the images directory:

   ```
   dcm_dump_file image
   ```

   This program will produce verbose information on the standard output and will print a description of all of the attributes in the image.
3. Run the program dcm_dump_file on one or more of the images that your organization has or produces:

   ```
   dcm_dump_file image   or
   dcm_dump_file -b image
   ```

   The -b switch is used for images that are stored in big-endian byte order. Hopefully, the program will print a similar description of your image(s). If the program fails, try adding the -v switch for even more verbose information. If this still fails, you need to contact the CTN provider to find out why their software does not understand your image format.
4. Run the program *dcm_verify* on one of the supplied test images:

   ```
   dcm_verify image
   ```

   This program examines an image and tries to determine if it includes all of the Modules and Attributes as defined by Part 3 of the Standard. It looks at each Information Entity and dumps information to the standard output. The last part of the output is a summary of type 1 and type 2 attributes that are missing. Hopefully, we have not included any images that are incomplete.

5. Run the program *dcm_verify* on one of your images. Since you have already accomplished step 2 above, you should know if you need the -b switch for this program. You will get a detailed list of the Information Entities, Modules and Attributes present in the image. The program will also print a list of type 1 and type 2 attributes that are missing from your image. If your V3 image fails to pass this test, it will likely cause problems for the rest of our software. You should examine the output of this program and Part 3 and correct your images to include all of the required attributes (or contact the CTN provider who may have incorrectly implemented the rules for this test program).
6. This test is only available on Unix systems.

   Run the program *dcm_x_disp* on one of the test images as well as one of your own images. *dcm_x_disp* will display the image on an X11 display; it should look "reasonable", although there may be problems with window center and width. *dcm_x_disp* understands the -b option. Don't forget to set the DISPLAY environment variable. *dcm_x_disp* requires certain parameters to be present before an image can be displayed. See the manual page for dcm_x_disp if there are problems.

It is possible that your organization may not agree with the summary information produced by the *dcm_verify* program. If you feel that we have made an error, please contact us so we can correct the problem.

4.2 Test Network and Snooper Software

This section describes a procedure for testing network connections between CTN applications and vendor applications. These tests demonstrate that Associations can be established which exercise the storage and verification classes. The last test exercises the DICOM communications monitoring (snooper) software that was specifically developed for the Solaris 2.x environment. The snooper software is only available under Solaris.

1. Run the program simple_storage which acts as an SCP of the storage and verification classes:

   ```
   simple_storage portnumber
   ```

   portnumber is the TCP/IP port address you choose for this server. It is typically the well known port number reserved for DICOM applications, 104. On Unix systems, you will need to be root to use this port number. You may prefer to choose a different number. For example, we run our tests with port 2100.

   Once the server has started, try to establish a connection with dicom_echo:

   ```
   dicom_echo -c DICOM_STORAGE hostname portnumber
   ```

   The -c switch tells *dicom_echo* to use DICOM_STORAGE as the called Application Entity Title. *hostname* is the name of the machine which is running *simple_storage*. *portnumber* is the number you selected for *simple_storage* at the top of this step.

If everything works as expected, *dicom_echo* will print several lines of summary information that are extracted from the C-ECHO Response message. The last line should be "Verification Successful". If this does not work as expected, run both *simple_storage* and dicom_echo with the -v switch.

2. *simple_storage* has an application title of DICOM_STORAGE. This program is picky and will reject association requests that do not use the proper called application entity title. This feature can be overridden by using the -i switch on *simple_storage* (-i stands for ignore some incorrect parameters in the Association request). If you use the -i switch, *simple_storage* has no mechanism for verifying that the caller's title is recognized. Therefore, you should be able to run whatever application you have which implements the Verification class and have it send a C-ECHO request message to *simple_storage*.

3. Run this test with one of the example images provided on our ftp site. Pick an arbitrary directory to work in. We will call that directory A. Create subdirectories in A with names that correspond to modalities as defined by the Standard: MR, CT, US,... In the directory A, run simple_storage as defined above. Use the program *send_image* to establish an Association with simple_storage and send one image:

   ```
   send_image <host name> portnumber imagefile
   ```

   send_image should print response messages when finished. If the image transmission was successful, there will be an image file in one of the subdirectories you created. (If you run *simple_storage* without the expected subdirectories, simple_storage will create them for you.)

4. Repeat test 3 with one of your own images.

5. Use *send_image* to send an image to your SCP of the Storage Class. There are switches to send_image which allow you to set the calling and called titles appropriately. Real applications will be more stringent about checking those parameters.

6. [Note: This test only applies to Solaris 2.x systems on which the SNP facility, DULsnoop extension, and DICOM snooper applications have been installed. This software has only been tested with Ethernet networks using the "/dev/le" interface. The test assumes that the same type of network and interface will be used.]

   Use the *dcm_snoop* application to monitor communications as described in step 1 of this section. *The dcm_snoop* application has to be run in super-user mode on a third machine that shares the same network as the two running the simple_storage and dicom_echo programs. The command on the third machine that needs to be given prior to the execution of the *simple_storage* and *dicom_echo* programs is:

   ```
   dcm_snoop /dev/le ppa host1 host2 portnumber 8192 1
   ```

   ppa is the number of the interface (generally 0 unless there is more than one Ethernet interface on the machine) where 0 is for /dev/le0, 1 is for /dev/le1, and so on.

host1 is the name/IP address of the machine running dicom_echo.

host2 is the name/IP address of the machine running simple_storage.

portnumber is the port number used by simple_storage.

8192 is the buffersize used.

1 is the number of associations to be monitored.

If everything works as expected, the output of the dcm_snoop program will show the exchange of the various PDUs of the DICOM association. The association request and accept parameters will be dumped. The DICOM commands will also be dumped.

4.3 Test Database Routines

This section describes a procedure for testing the table facility (TBL) which is used as a basis for the database operations in the CTN. The applications are found in the ../apps/tbltest directory. These applications should be available after the make: *ttunique, ttinsert, ttdelete, ttselect, ttlayout*, and *ttupdate.* These tests will demonstrate that you have the ability to manipulate records contained in the database. It will be helpful to inspect the source to these routines before running each application to get a feeling for the how the applications are designed and what they actually do.

The first step is to create the database tables. The steps for creating the tables under Unix are included in this paragraph. Configure the database with the appropriate temporary tables needed to run these test procedures. This configuration script is CreateTables and may be found in the ../cfg_scripts/sybase or cfg_scripts/msql directory. If you are using a database system other than Sybase or miniSQL these configuration scripts will need to be modified appropriately. The arguments to this script for this test should be TBLTest TBLTest. Upon successful completion, this script will create two new tables in the specified database, TBL_Persons, and UniqueNumbers.

Use the graphical user interface tools provided by Microsoft to create a database: TBLTest. After this step, you need to create database tables. In the Windows environment, we use the SQL Enterprise Manager. Run this application and select tools->SQL Query Tool. From this tool, you can select Load SQL Script and run the script in

cfg_scripts/mssql_server/createtbltesttables.sql.

1. The success of the table creation can be checked by running the program ttlayout. ttlayout requests two pieces of in formation, the database name and the table name. In this example, the database name is TBLTest and the table name is TBL_Persons. This routine should yield the following in formation:

```
Column #: 1  Length: 50  Type: String  Name: FNAME
Column #: 2  Length: 50  Type: String  Name: LNAME
Column #: 3  Length:  4  Type: Signed4 Name: AG
Column #: 4  Length:  4  Type: Signed4 Name: ZIP
Column #: 5  Length:  4  Type: Float4  Name: WEIGHT (Float8 for miniSQL)
```

This is the layout of the TBL_Persons table, and this output verifies that the function TBL_Layout is working as well as the database initialization scripts.

2. Run the test application *ttinsert* . If it is successful, it will report the simple message: "All Inserts succeeded". If not, there will be other error messages that need to be addressed.
3. Run the test application *ttselect*. This application will produce the following output if successful:

```
In callback: Count is: 1
Field Name: FNAME [7]: JOE
Field Name: LNAME [7]: JONES
Field Name: AGE [4]: 30
Field Name: ZIP [4]: 63100
Field Name: WEIGHT [5]: 150.100

In callback: Count is: 2
Field Name: FNAME [7]: SMELDA
Field Name: LNAME [7]: SMITH
Field Name: AGE [4]: 40
Field Name: ZIP [4]: 63200
Field Name: WEIGHT [5]: 160.200

In callback: Count is: 3
Field Name: FNAME [7]: JOHN
Field Name: LNAME [7]: JONES
Field Name: AGE [4]: 50
Field Name: ZIP [4]: 63300
Field Name: WEIGHT [5]: 170.500

In callback: Count is: 4
```

```
Field Name: FNAME [7]: SMITHY
Field Name: LNAME [7]: SMITH
Field Name: AGE [4]: 60
Field Name: ZIP [4]: 63400
Field Name: WEIGHT [5]: 180.250
ALL DONE--Count: 4
```

4. Run the test application *ttupdate* to test the update function. If successful, this application will report the message "Update operation succeeded". Inspect the last name of the record that has an age field equal to 50, and the last name should now be "Woodrow".
5. The delete function is tested with the application *ttdelete.* If successful, it will report "Delete operation succeeded". Inspect the records in the database to ensure that the record containing a zip code of "63100" is gone.
6. The last function to be tested is a very simplistic unique number generator. The application that tests this function is *ttunique*. Running this application should produce the following output:

```
UN1 iteration: 1  count: 1
UN1 iteration: 2  count: 2
UN1 iteration: 3  count: 3
UN1 iteration: 4  count: 4
UN1 iteration: 5  count: 5
UN1 iteration: 6  count: 6
UN1 iteration: 7  count: 7
UN1 iteration: 8  count: 8
UN1 iteration: 9  count: 9
UN2 iteration: 0  count: 0
UN2 iteration: 1  count: 1
UN2 iteration: 2  count: 2
UN2 iteration: 3  count: 3
UN2 iteration: 4  count: 4
UN2 iteration: 5  count: 5
UN2 iteration: 6  count: 6
```

```
UN2 iteration: 7  count: 7
UN2 iteration: 8  count: 8
UN2 iteration: 9  count: 9
UN3 iteration: 0  count: 0
UN3 iteration: 1  count: 1
UN3 iteration: 2  count: 2
UN3 iteration: 3  count: 3
UN3 iteration: 4  count: 4
UN3 iteration: 5  count: 5
UN3 iteration: 6  count: 6
UN3 iteration: 7  count: 7
UN3 iteration: 8  count: 8
UN3 iteration: 9  count: 9
```

4.4 Test Queuing Routines

These tests are only available for Unix systems.

This section describes a procedure for testing the queueing mechanism which will be used by the print server display program. In order to test out the queuing mechanism, a pre-existing utility will be used that was originally designed for *ctndisp*. Although this example may not make much sense at first, successful completion will indicate the the GQ facility is operating properly.

1. Use setenv to set the QUEUE_DIRECTORY environment variable to your current directory (e.g. setenv QUEUE_DIRECTORY ./). Use the following command to create a new queue of 10 elements where each element is 516 bytes in length (the queue id is 0):

   ```
   gqinitq 0 10 516
   ```

   If no error messages appeared, the command succeeded. You can use the *ipcs* command to check and make sure that there is one semaphore and one shared memory resource that belongs to your current login.

2. Put some elements on this queue with the *enq_ctndisp* command. Use the following commands:

   ```
   enq_ctndisp 0 image1 dpn1 1 1
   enq_ctndisp 0 image2 dpn2 2 2
   enq_ctndisp 0 image3 dpn3 3 3
   ```

If no error messages appeared, the enqueues were successful.

3. Examine the queue with the command `pq_ctndisp 0`. The ouput should look something like the following:

```
<<<  HEAD >>>
Queue Element: 1
Image File: image1
DPN id: dpn1
Connection: 1--  Image num: 1
        .
        .
        .
<<< TAIL >>>
```

4. Try removing the queue with the command gqkillq command:

```
gqkillq 0 516
```

   The command operates silently unless an error occurred. Use the command ipcs to determine that the semaphore and shared memory resources once allocated to your login id are now gone.

4.5 Test Demonstration Programs

Once you have completed the tests above, you have demonstrated the pieces that are necessary to run the demonstration programs. There is no detailed procedure to test those programs. The *User's Guide for CTN Demonstration Applications* provides sufficient information for configuring and running those programs.

Appendix A: Macros Controlling Build Environment for Unix

TABLE A-1: Macros Defined for Enabling CTN Features

Macro Name	Definition
DEBUG	Turns on some additional print/debug information in facilities. This code will be tripped when you place a facility in "debug" mode.
BIG_ENDIAN_ARCHITECTURE	Needs to be defined on big-endian on big-endian machines.
LITTLE_ENDIAN_ARCHITECTURE	Needs to be defined on big-endian on little-endian machines.
SHARED_MEMORY	Required for shared memory operation in queueing functions.
SEMAPHORE	Required for semaphore operations in queueing functions.
USLEEP	If the usleep function is provided by your operating system.
SYBASE	Defined if your system has sybase installed.
MSQL	Defined if your system has miniSQL
USEREGCOMP	Defined if you want to use the regcomp and regexec functions. These are found on HP and other systems.
SNOOP	Turns on SNP facility, DULsnoop extension, and DICOM snooper applications (Note: These will only work in a Solaris 2.x environment on Sun equipment.
CTN_NO_RUNT_PDVS	In some instances, the CTN can generate 0-length PDVs when writing data over the network. Turning on this option will perform some additional runtime tests to eliminate this. This was defined to satisfy one vendor who believes that 0-length PDVs are illegal. We don't agree, but added the software as an option.
CTN_USE_THREADS	Define this compile time macro to make the CTN code thread safe. Note: This is currently under test with Solaris and Windows and is not guaranteed to produce thread safe code.

TABLE A-2: Macros Defined for Controlling the Compilation Environment

Macro Name	Definition
LIBPATH_X11	Switch passed to linker for pathname for X11 libraries. Probably need not be defined unless your X11 libraries are not in a standard location.
LIBPATH_MOTIF	Switch passed to linker for pathname for Motif libraries. Probably need not be defined unless your Motif libraries are not in a standard location.
LIBPATH_UCB	Switch passed to linker for pathname for UCB libraries that are required for some operations (socket) under Solaris.
LIBPATH_DATABASE	Switch passed to linker for pathname for database libraries. This can be the pathname for sybase libraries or for a different database product.
LIBS_X11	Switches passed to linker to tell it which libaries to search for applications that use X11 (not the path to the libraries).
LIBS_MOTIF	Switches passed to linker to tell it which libraries to search for applications that use Motif (not the path to the libraries).
LIBS_XAW	Switches passed to linker to tell it which libraries to search for applications that use the Athena widget set (not the path to the libraries).
LIBS_OS	Switches passed to linker to tell it to search any libraries that are dependent

	on the OS. We found this important for some libraries under Solaris 2.x.
LIBS_DATABASE	Switches passed to linker to tell it to search libraries to resolve database references. This is the switch you would use to tell the linker to search the libraries supplied by sybase to resolve the TBL references to sybase functions (or miniSQL).
LIBS_CTN	The list of libraries provided in the CTN software in order needed to resolve refererences. This is a convenience macro that makes it simple for someone to link an application (without having to remember each CTN library). Beginning with version 2.11, we are compiling the CTN object files into a single library. This list is now the single CTN library and any system specific libraries.
LIBS_CTN_NODB	Similar to LIBS_CTN, but includes no database references. This is useful for building standalone utilities independent of a database. You will not need the shared library for the database to use the utility.
OS	An environment variable that we pass to makefiles to define the operating system. Will get used for conditional compilation. Values that are supported are: AIXV3 HPUX IRIX OSF SOLARIS SUNOS ULTRIX.
CFLAGS_X11	Any flags passed to compiler when compiling X11 applications. In some environments, this is a -I switch to give the location of include files.
CFLAGS_MOTIF	Any flags passed to compiler when compiling Motif applications. In some environments, this is a -I switch to give the location of include files.
LONGSIZE	The size of a variable (in bits) of type long on your system. Needs to be defined for our code to compile.
INTSIZE	The size of a variable (in bits) of type int on your system. Needs to be defined for our code to compile.
SHORTSIZE	The size of a variable (in bits) of type short on your system. Needs to be defined for our code to compile.
C_OPTS	The concatenation of a number of options to be passed to the compiler. Makefiles are expected to set CFLAGS equal to $(C_OPTS) plus whatever local options are needed.

The environment files contain variables that define path names for files and target directories. It is safest to make these absolute path names.

TABLE A-3: Macros Defined for Controlling the Compilation Environment

Environment Variable	Definition
DICOM_ROOT	The root directory for the installation. Most other directories are defined from this point.
DICOM_BIN	The location of the compiled binaries for the system. This will be the target directory when you rebuild the applications. Our destination directory is DICOM_HOME/bin/OS. You may choose to install your binaries somewhere else (e.g. /usr/local/bin).
DICOM_LIB	The location of the compiled library files. This will be the targetdirectory when you rebuild the libraries. Our

	destination directory is DICOM_HOME/lib/OS. You may choose to install your libraries somewhere else (e.g. /usr/local/lib). This is also used by our Makefiles when linking applications.
DICOM_INCLUDE	The location of the common include files for the subroutine libraries. Our Makefiles use this variable but also require that the include directory is DICOM_HOME/include. (The "make" supplied by one of our computer vendors has a problem with the VPATH variable.) Please use DICOM_HOME/include for this variable.
DICOM_MAKE	The path name to a file which is included by all Makefiles in the system. This allows us to set a number of global make options.
CC	The C compiler to use to compile libraries and applications. This should be an ANSI compliant C compiler. If your system's cc is ANSI compliant, you do not need to define this variable (its default is CC). 5.2.2 Windows: Compilation

Appendix B: Platform Specific Notes

B.1: Silicon Graphics workstation, IRIX

This information is based on experience with SGI machines running 5.x releases of Irix. We are likely well behind the current release of the OS.

We had a report from someone using Sybase on an SGI about installation problems. The messages said:ninit: t_open, No such device or address

ninit: All master network listeners have failed.

Sybase says this is a known problem with SGI and that you need to install an SGI patch to correct the problem. The SGI patch is EOE1.sw.svr4net

We don't have more explicit information about what version of IRIX this applies to or how to obtain the patches from Silicon Graphics.

There are also some libraries that Silicon Graphics uses for network operations that appear in libnsl.a. If you find unresolved references to t_errno or t_open, you will want to include the switch -lnsl in the macro LIBS_OS for the SGI environment. We also had a report about a problem with the SGI implementation of gethostbyname. To get around this, the user included the switches -lc -lnsl on the link line (in that order).

Since these reports, we have compiled a version of the software under Irix 5.2 and using mSQL. We have not run extensive tests, but we were able to get a clean compile.

B.2: Linux

We have compiled the software on Red Hat Linux (6.0) and run some simple tests. Linux does not ship with Motif but there is a Motif clone called LessTif (http://www.lesstif.org). We have compiled the CTN software and performed minimal tests with this package.

We had two different users submit scripts for building the CTN software under Linux. These are in the contributed directory. You are welcome to use those scripts; we do not use them ourselves nor do we support them. We build the software using the install procedures described in this manual.

B.3: Win32

We are compiling the software using version 6.0 of the Microsoft Visual C++ compiler and version 7.0 of the Microsoft SQL server. We have not seen any issues with either the compiler or database.

Appendix C: Database Issues

Some of the CTN demonstration applications require the use of relational database. In the original implementation, this was accomplished through the use of Sybase. In this version of the CTN software, we have included an implementation that can use Sybase, miniSQL, PostgreSQL on Unix systems and Microsoft SQL Server 7.0 on Windows NT systems. Users can choose the appropriate version through the use of the proper environment files.

The remainder of the section contains notes about the installation of Sybase, miniSQL and PostgreSQL. This does not serve as a substitute for the documentation for those products. We do not include precompiled libraries or executables for either database product.

C.1: MiniSQL Installation Notes

The author of miniSQL has changed his licensing terms. We feel that we can no longer distribute the miniSQL code with our software. Your organization will need to obtain the software and abide by the licensing rules. The ftp site is bond.edu.au. miniSQL is now at version 2 (or higher). We have email from a user who said he only needed to change the scripters for creatating tables. We have included the changes in cfg_scripts/msql2 but have not tested it yet. We are running version 1.0.16.

There are a number of steps that must be performed in order to successfully install MiniSQL (msql) on your particular machine. These steps are enumerated below:

1. You should definitely print out the file .../doc/mSQL-1.0.ps. This is a copy of the user/admin manual and is very useful.
2. In general, follow the instructions in the README file in the MiniSQL install directory for compiling, installing, and testing msql.
3. You should create an msql user, and that user should be a member of the group that owns the CTN files (as released, group ID 100). Log in as msql to perform the following tasks. Database administration can only be performed as the msql user (please remember this fact).
4. Obtain the msql software from the ftp site listed above. Please pay the licensing fee.
5. Follow directions in README file for the make, setup, and install of msql. We suggest you use the default installation directory, /usr/local/Minerva. It also tells you how to start the server, etc. Be advised that when you start the server (msqld&) you will probably get a message that says:

   ```
   Couldn't open ACL file: No such file or directory
   ```

   This is ok. It just means that all the msql databases created will be globally available to all users. If you want to restrict usage, there are sample acl files around which will tell you how to do that, but for the purposes of testing, it probably just easier to leave access open.

6. Proceed to the directory ...../msql-version/targets/<your target>/tests in the mSQL distribution.
7. The "killer" script should now run.
8. The "rtest" script should also run.
9. If all this goes well, this should confirm that msql is alive and well.
10. You will have to make small changes to the script files killer and rtest to correctly specify locations for executables, etc. Please remember that as a rule, we have not changed any of this release, and we have had very few problems with the system to this point. Be advised that building and testing the system may be slightly different than you are used to with the DICOM distribution developed at MIR.

C.2: Sybase Installation Notes

There are a number of steps that must be performed in order to successfully install Sybase on your particular machine. The majority of these steps are outlined in the Sybase Installation Guide and will not be repeated in this section. If you are not using Sybase, the scripts that are supplied to you will be unusable except to examine them for functionality before porting them to your particular database environment.

After successfully installing Sybase, there are a couple of items that need some special attention. First, you must add a new user to the Sybase system called sybase with a password of sybase. All libraries supplied access the databases(s) with this user name and password. You must also be sure that you have successfully created devices on your system that Sybase can use for database storage. These device descriptions will of course vary depending on your system configuration. The script CreateDevice.template, located in ../cfg_scripts/Sybase/MakeDevices, will be of considerable use. You may need to be logged in under the sybase user account to use these scripts. The applications can be run from other accounts because they have code to use the sybase account.

Note: It may be OK to use the account name sybase, but it is a bad idea to use sybase as the password. for that account. We strongly suggest that you use a different password. for the account.

Sybase itself has a password for each user that is different than the system password. By default, our software uses the name sybase and the password sybase to access the SQL server. If you want to use a different password for sybase, you need to change the constant SYBASE_PASSWORD which is defined in

facilities/tbl/tbl_sybase.h

A problem that you will most likely experience using Sybase is one of running out of user connections at some point. This is relatively easy to repair using the following commands. Get into the interactive SQL interpretor (isql -Usa), and execute the command sp_configure.

The two parameters to look for are "user connections" and "memory". Both of these will most likely need to be increased and this is accomplished as follows:

```
    sp_configure "memory", 8192
go
sp_configure "user connections",50 (or 100)
go
reconfigure
go
shutdown (you need to restart the server)
go
```

This procedure will set up the new parameters and restart the server so they can take effect. Your memory is most likely defaulted to 4096 and therefore doubling it is a reasonble thing to do. The user connections parameter is probably set at 20 or so and is actually the resource you are running out of. Be careful, you can't (shouldn't) increase the user connections without increasing the memory, or your server may not restart....so be sure you get the memory increased.

C.3: PostgreSQL Installation Notes

The PostgreSQL software is available at www.postgresql.org. If you are using Red Hat Linux 6.0 or greater, this software can be installed from the Red Hat distribution.

Pay special attention to the environment variables PGDATA and PGLIB which should be set for the postgres account. If these are not set properly, the database initialization (initdb) will fail. The default values in the Red Hat Linux 6.0 environment are:

Variable	Value
PGDATA	/var/lib/pgsql
PGLIB	/usr/lib/pgsql

Similarly, when you use the CTN applications from a different account (not the postgres account), you will need to set the PGUSER environment variable to postgres.

C.4: MS SQL Server Installation Notes

The MS SQL Server product provides a setup tool that guides you through the installation process. When the installation is complete, you will be able to interact with the server using interactive SQL and their GUI-based management tools. The CTN software communicates with the server using ODBC, and your PC must be configured to use that channel. Check the ODBC configuration to make certain it is correct.

Open the Control Panel folder and then open ODBC. Select the "System DSN" tab. There will be a list of system data sources. There should be one called LocalServer which uses the SQL Server driver. This entry is created by the SQL Server installation procedure. Our server is configured with these values (set by the SQL Server setup program).

Variable	Value
Name	LocalServer
Description	<blank>
Which server	local
Verify authenticity by …	SQL Server authentification
Login ID	ctn
Password	ctn

The CTN runs on the same machine as the SQL Server, so we use the local connection. We do not use the trusted connection option, but you might decided to do so depending on how you want to configure your system.

The database software uses the environment variable SQL_ACCESS to determine the 3 parameters needed to establish the connection to the database. The format of this variable is:

<server name>:<login>:<password>

We use this at our site: LocalServer:ctn:ctn

If you do not define this environment variable, the LocalServer:ctn:ctn values are used by default. You might choose to use a different login name or password or a different scheme. If you choose to use the Trusted Server feature of the system, you can leave the login and password values blank in the SQL_ACCESS variable: "LocalServer:::".

The SQL Server has security features that allow the administrator to restrict access to tables in the database. You may find that you need to open up access to get the CTN software to operate properly. Run the CTN programs as described below or in the User's Guide. If they complain about access privileges, you will need to use the SQL Enterprise Manager to give you access rights to the databases (insert, delete). There are several methods for allowing access. One simple method is to activate the Manage pulldown (in SQL Enterprise Manager) and select logins. For the login that you are using, alias that login as dbo (stands for database owner) for the databases you are using. That should give you the privileges you need. We also suggest you read the SQL Server documents to understand their security features (they will certainly explain them better than we can).

Appendix D: X11/Motif Installation Notes

All of the Motif-based applications in this release were designed with the commercially available interface builder, UIM/X. These applications automatically look for the file XKeysymDB in the directory /usr/lib/X11. If it isn't found, you will receive (many) warning messages of the following type when an application is started:

```
Warning: translation table syntax error: Unknown keysym name: osfActivate
Warning: ...found while parsing `<Key>osfActivate:ArmAndActivate()'
```

While these messages are not fatal to the application, they are a bother, and can be alleviated by placing the proper keysym definition file in /usr/lib/X11/XKeysymDB. This keysym file is typically a part of standard X11/Motif distribution, but may simply be placed in a different location than the one specified previously. The color definition file /usr/lib/X11/rgb.txt should also be present.

D.1: Motif/Linux

We have installed the LessTif package as a Motif clone. The software and documentation are at www.lesstif.org. We have used the default installation directory and the 1.2 version of Motif. Please read the LessTif documentation for installation instructions. One detail for RH Linux is that you will probably want to update the file /etc/ld.so.conf with the path to the LessTif shared libraries. Using the default installation location, this value is /usr/local/LessTif/Motif1.2/lib.

ss