

The server core (**dgate.exe** = **dgate** under Linux) compiles and runs on Linux systems and Solaris. I develop primarily under Windows, but currently I test the code and scripts under Linux Debian 12, and Docker. I also had the server compiled on a Raspberry Pi, Rocky Linux (requires manual install of prerequisites), Docker in Synology and many other systems.

The Linux release of the server core works default with SQLite driver built in into the server (no ODBC). The DbaseIII driver is also supported. Piotr Filipczuk has added a PostGreSQL driver. The native MySQL interface also can be used. The graphical user interface has not been ported to Linux, but the WEB interface is provided, either using Apache or a built-in mini web server (Ladle). In this version, most options have been well tested – it is a stable release. However, there are often subtle differences between Linux distributions, making installation (and writing a manual) difficult. There are several contributions on the forum, and there are text files with specific command orders to be found in the linux subfolder of the server.

To use the server, one needs a valid version of the configuration files in the same directory as the dgate executable.

## INSTALLATION

### Option 1 – preferred; using an install script

This does require that the logged-in user is in the /etc/sudoers file. The script should not be run as user root.

```
sudo apt install lua5.1
```

```
rm installer.lua
```

```
wget https://raw.githubusercontent.com/marcelvanherk/Conquest-DICOM-Server/master/install/installer.lua
```

```
lua5.1 installer.lua
```

It will first test and ask to install required prerequisites. If you allow that, then after this completes you must again run:

```
lua5.1 installer.lua (or lua installer.lua)
```

This completes the setup of the server based on interactive questions, including installation as a service and providing the web interface. The installer can run on Lua5.1, which is then shared with the server. If Lua5.1 is not available on your Linux version, you can use newer versions to run installer. However, Conquest requires Lua5.1 which then must be compiled in into the server, using the -l option of installer.lua.

The command line options are shown as follows:

```
lua installer -h
```

```

-v --verbose
-d --database sqlite|mariadb|dbaseiii|pgsql|null
-r --recompile (needed when changing database)
-c --configure (force reconfigure)
-l --luabuiltin (compile lua and socket into server)
-w --webbuiltin port (autostart ladle webserver on port)
-p --port port (server port)
-s --servername name (server name)
-r --regen y/n (force y/n dbase regen)
-y --yes (say yes to most questions)
-z --sudopw pw (password)

```

So to install conquest unattended use e.g.:

```

cd ~
lua installer.lua -d sqlite -p 5678 -s CONQUESTSRV1 -r n -y -z Password

```

If folder CONQUESTSRV1 exists, this will update the server while keeping the configuration, data and database. The option -r n avoids regeneration of the existing database, which is a good idea if updating.

If it does not exist server CONQUESTSRV1 will be created in folder CONQUESTSRV1. The correct command line is then:

```

cd ~
lua installer.lua -d sqlite -p 5678 -s CONQUESTSRV1 -r y -y -z Password

```

If you run installer.lua inside the server folder, you can use it to recompile and reconfigure, but not to update the installation.

## Option 2; installing manually

Prerequisites: 1) a running Linux system. 2) sudo, nano, systemctl, gettext-base installed. These are normally present but missing in bare-bones Docker containers and in that case must be added first. Note that I only test the scripts on Ubuntu, but the web based installer script linux.sh has a bit of info on Fedora.

These packages needed to be installed in a plain Linux system for a release using SQLite or DbaseIII:

```

sudo apt update
sudo apt install make
(or: sudo apt install build-essential)
sudo apt install g++
sudo apt install apache2
sudo apt install php libapache2-mod-php php-sqlite3
sudo apt install unzip
sudo apt install 7zip

```

```

get compilers
may be skipped if using precompiled

may be skipped if using precompiled
get webserver
get PhP integration
not standard in Ubuntu server
parts of the web interface use 7za

```

```

sudo ln -s /usr/bin/7zz /usr/bin/7za
sudo apt install lua5.1
sudo apt-get install lua5.1-dev
(or sudo apt install liblua5.1-0)
(or sudo apt install liblua5.1-0-dev)
(sudo ln -s /usr/lib/x86_64-linux-gnu/liblua5.1.so.0 /usr/lib/x86_64-linux-gnu/liblua5.1.so)
(sudo ln -s /usr/lib/aarch64-linux-gnu/liblua5.1.so.0 /usr/lib/aarch64-linux-gnu/liblua5.1.so)
sudo apt install lua-socket

```

Conquest expects 7za  
since 1.5.0 lua is external  
may be skipped if using precompiled  
is required when using precompiled

sometimes needed  
same for ARM Linux

```

sudo apt install luarocks
sudo luarocks install luafilesystem

```

to install additional Lua libraries

```

(or for fedora:
dnf install gcc-c++-sh-linux-gnu.x86_64 gcc-c++-x86_64-linux-gnu.x86_64 clang.x86_64
)

```

```

sudo a2enmod cgi
sudo a2enmod rewrite
sudo sed -i 's/AllowOverride None/AllowOverride All/g' /etc/apache2/apache2.conf
sudo sed -i 's/memory_limit = 128M/memory_limit = 512M/g' /etc/php/*/apache2/php.ini
sudo sed -i 's/upload_max_filesize = 2M/upload_max_filesize = 250M/g' /etc/php/*/apache2/php.ini
sudo sed -i 's/post_max_size = 8M/post_max_size = 250M/g' /etc/php/*/apache2/php.ini
sudo systemctl restart apache2
(or for older systems: sudo service apache2 restart)

```

enable CGI in server (only for install.sh)  
enable .htaccess

give PHP more oomph

The rest of the installation can be performed manually, or by a web based method, explained below.  
The following steps illustrate a minimal installation:

First get the server:

```

wget http://natura-ingenum.nl/dicomserver/dicomserver150g.zip
mkdir conquest
cd conquest
unzip ../dicomserver150g.zip
rm ../dicomserver150g.zip

```

get server zip

make folder to store conquest  
to there

**Or:**

```

sudo apt install git
git clone https://github.com/marcelvanherk/Conquest-DICOM-Server
cd Conquest-DICOM-Server

```

if git not installed yet  
get latest from GitHub

Then compile and install it:

```

chmod 777 maklinux
./maklinux

```

compile and install web access

choose option 3 or 5  
say 'y' to 'Regenerate the database'  
say 'y' to 'Install as service' Shows status hit 'q' to return

SQLite or SQLite precompiled  
**Deletes previous database contents**

Now the server should be running and <http://localhost/app/newweb/> should provide a working web interface. <http://localhost/app/ohif> should work as well.

Note that in [dicomserver150g](#) a precompiled dgate from 1.5.0g (compiled by me on Ubuntu14, using SQLite database) is included, to try that use option 5 in *maklinux*. Tested on Ubuntu 18.04, 19.10, Debian 12. If used the following packages may be *omitted*: **make, g++, lua5.1-dev**; but if you do omit them then the following package must be *added*: **liblua5.1-0**. This option reduces the size of the Linux system by a few hundred MB. To run conquest on the command line use e.g., `./dgate -v`

## **Web based installation <has been deprecated but not removed>**

To run the web based installer (after installing prerequisites):

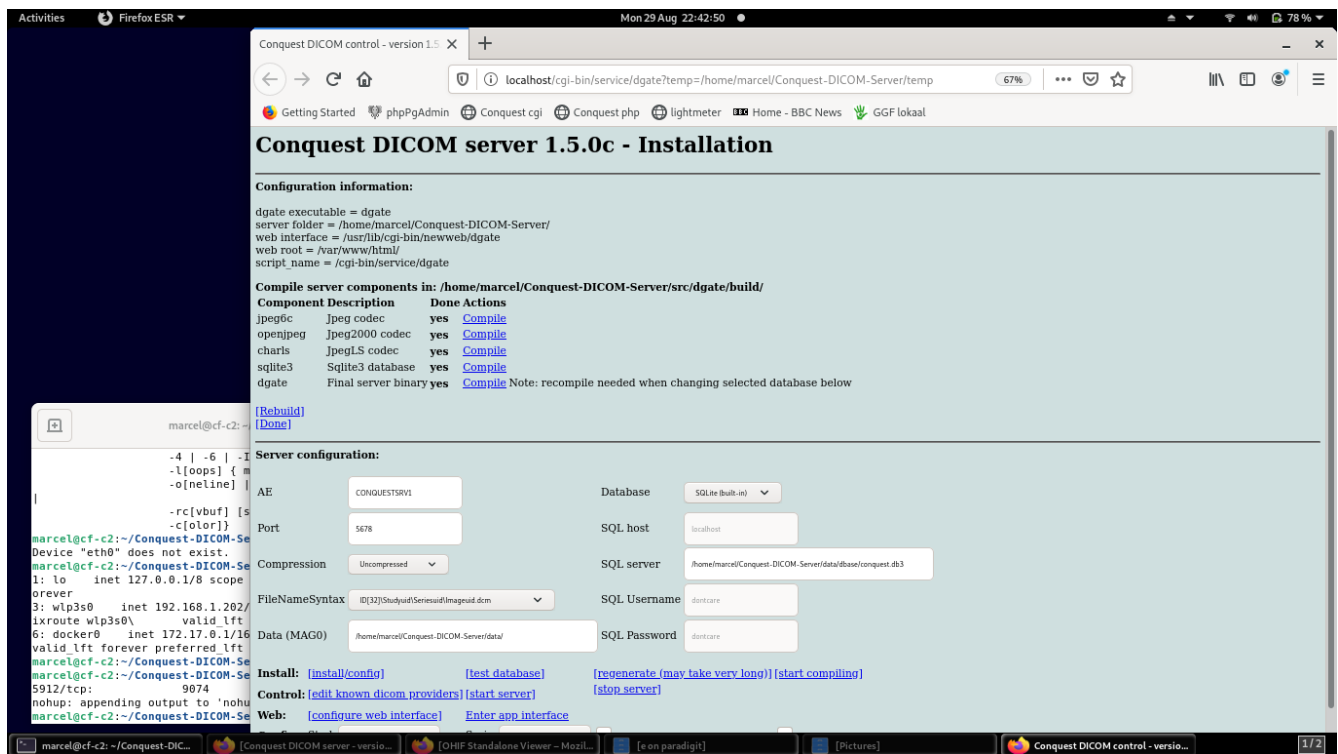
```
wget http://natura-ingenum.nl/dicomserver/dicomserver150g.zip  
mkdir conquest  
cd conquest  
unzip ../dicomserver150g.zip
```

get server zip  
make folder to store conquest

```
cd install  
chmod 777 linux.sh  
./linux.sh
```

make run-able

This compiles a minimal server binary (dgatesmall) or uses the pre-compiled one, that is run as service control manager and, if a web server and client exist, opens web page <http://127.0.0.1/cgi-bin/service/dgate>. The resulting web page allows and guides the user through compilation, configuration, re-generation of the database if needed, starting the server, setting up the web server and opening the web client. A screen-shot of the install page is shown below:



The required steps (most are shown in the welcome area) are:

- 1) Select required database type (start with SQLite if unsure)
- 2) Start compiling → compile jpeg6c, compile openjpeg, compile charls, compile lua, compile luasocket, compile sqlite, compile dgate; [done].

*If any of the compilation steps fails error messages can be found in file nohup.out. If the compilation information disappears click start compiling again.*

- 3) Set other parameters (keep defaults if unsure)
- 4) Configure server
- 5) Start server (may need be repeated a few times if does not start)
- 6) Regenerate database
- 7) Configure web interface (select viewers and access rights)
- 8) Enter server's web interface <http://localhost/app/newweb>

An additional web tools that is installed by the installer is <http://localhost/app/ohif>.

Feedback on this new installation method would be appreciated. After installation, the server runs as part of the control manager. To make it run permanently, stop the server control manager (dgate) with ^C, and use the new start-stop-daemon method described above or the old one below. Note that stopping the server using this web page on Linux disables restarting it for a minute or so (due to an IP port being blocked). Be patient when it fails not restart and try again after a while.

## Deamon configuration

All of the script install, web install and maklinux now create a daemon as follow, changing the file to point to the conquest installation:

```
sudo cp conquest.service /etc/systemd/system/conquest.service
```

```
sudo systemctl daemon-reload
```

After installation you can control the conquest service as follows:

```
sudo systemctl start conquest.service
sudo systemctl enable conquest.service
sudo systemctl status conquest.service
```

hit 'Q' to return

```
sudo systemctl stop conquest.service
sudo systemctl disable conquest.service
```

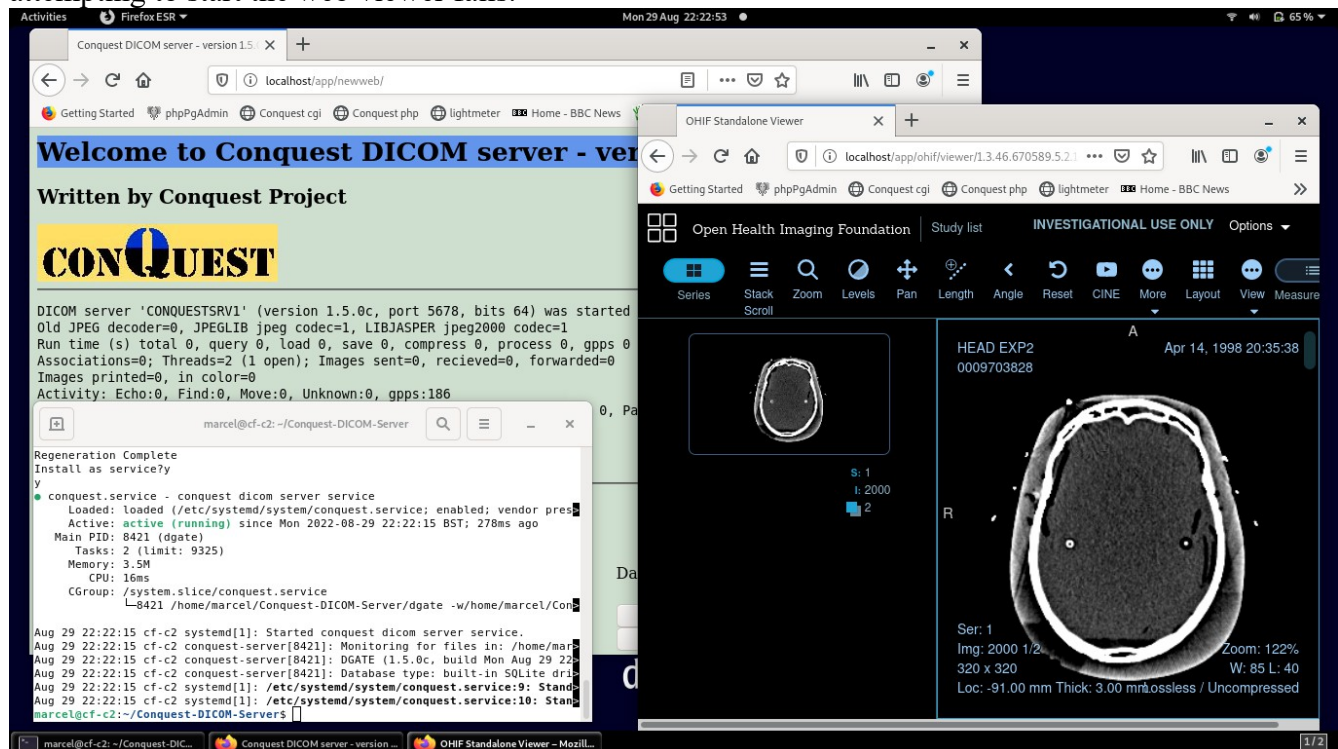
Now the server should be running, also after a system restart, and localhost/cgi-bin/newweb/dgate should provide a working web interface.

## Built-in WEB Viewer

A new single user web viewer can be run as follows:

```
chmod 777 linux/webviewer.sh
linux/webviewer.sh
```

This is the same web viewer as can be accessed from a full featured web server, but instead it runs on 127.0.0.1:8086, using Ladle (single user web server) as mini web server. After stopping the browser, the Ladle function is stopped. It takes a minute or so for the used port (8086) to be released. Until then attempting to start the web viewer fails.



Example of newweb and ohif web viewer running on Debian 11. Works directly after running maklinux.

## Installing with Postgres

To install with Postgres as database, these commands are needed to install and setup Postgres:

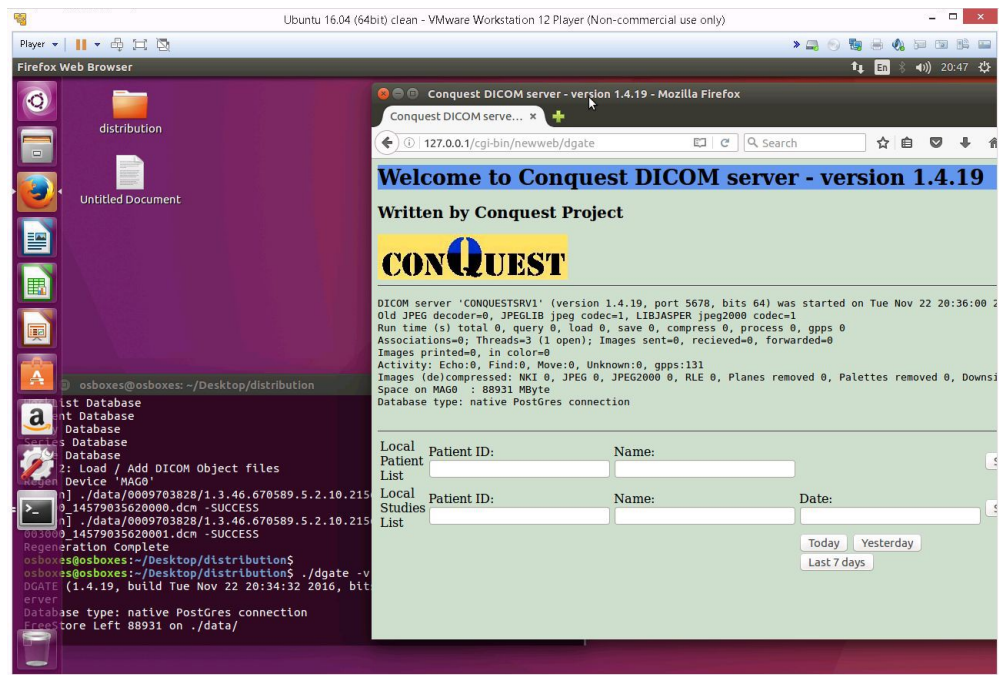
|                                              |                                |
|----------------------------------------------|--------------------------------|
| <code>sudo apt-get install libpq-dev</code>  | Postgres development tools     |
| <code>sudo apt-get install postgresql</code> | Postgres database              |
| <code>sudo su</code>                         | become superuser               |
| <br>                                         |                                |
| <code>su - postgres</code>                   | become postgres user           |
| <code>psql</code>                            | set the password to postgres   |
| <code>\password</code>                       |                                |
| <code>postgres</code>                        | (password)                     |
| <code>postgres</code>                        | (repeat password)              |
| <code>\q</code>                              |                                |
| <code>createdb conquest</code>               | create database conquest       |
| <code>exit</code>                            |                                |
| <code>exit</code>                            |                                |
| <br>                                         |                                |
| <code>./maklinux</code>                      | compile and install web access |
| <code>choose option 2</code>                 | Postgres                       |

The build process always gives a few error messages that can be ignored:

```
/usr/bin/install: cannot create regular file '/usr/local/man/man1/cjpeg.1': No such file or directory
Makefile:200: recipe for target 'install' failed
mkdir: cannot create directory 'data/dbase': File exists
```

During database creation (dgate -v -r) there can be error messages about non-existing databases, e.g. for postgres:

```
osboxes@osboxes:~/Desktop/distribution$ ./dgate -v -r
Regen Database
Step 1: Re-initialize SQL Tables
*** ERROR: relation "dicomworklist" does not exist
....
***Error: ERROR: table "uidmods" does not exist
WorkList Database
Patient Database
Study Database
Series Database
Image Database
Step 2: Load / Add DICOM Object files
Regen Device 'MAG0'
[Regen]
./data/0009703828/1.3.46.670589.5.2.10.2156913941.892665339.860724_0001_003000_14579035620001.dc
m -SUCCESS
Regeneration Complete
osboxes@osboxes:~/Desktop/distribution$ ./dgate -v
```



Conquest in action on Ubuntu16.04, with Postgres database and web interface

## Installing with Mariadb

To install with Mariadb as database, these commands are needed to install and setup:

```
sudo apt install mariadb-server
sudo apt install libmariadb-dev
```

Mariadb server  
Client code

```
sudo mysql
>create user conquest;
>grant all privileges on *.* to 'conquest'
>set password for conquest = password('conquest')
>create database conquest;
>flush privileges;
>\q
```

database superuser

create user conquest  
with password conquest  
create database

```
./maklinux
choose option 1
```

compile and install web access  
mariadb

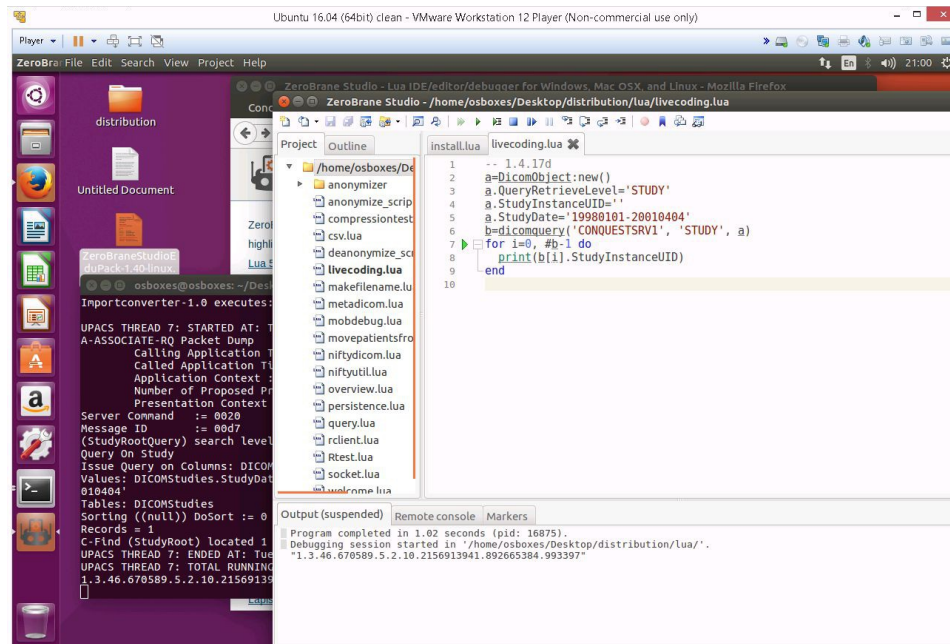
## ZeroBraneStudio IDE

To install and use ZeroBrane Studio with the conquest DICOM server under Linux, take these steps. First download ZeroBraneStudioEduPack-xxx-linux.sh. Then in a command prompt run:

```
chmod 777 ZeroBraneStudioEduPack-xxx-linux.sh
```

```
sudo ./ZeroBraneStudioEduPack-xxx-linux.sh
```

After installation is done run ZeroBrane Studio from the command prompt as “sudo zbstudio” and run the install script /dicomserver/ZeroBraneStudio/install.lua in ZeroBrane Studio as described in this file. After running the conquest install script as root, ZeroBraneStudio can be run as a normal user.



Integration of Conquest with Zerobrane Studio

## CONFIGURATION

Configuration files under Windows and Linux are the same except for the use of a forward slash instead of back slash in directory paths. The following essential entries are therefore different for Linux (these are the defaults):

```
SQLServer          = /home/user/conquest/data/dbase/conquest.db3
MAGDevice0         = /home/user/conquest/data/
```

See the Windows manual for more details about the configuration files (you need at least to edit **acrnama.map** to define DICOM systems that will be retrieving information from your server). All configurations options in **dicom.ini** (e.g., for DICOM routing) are listed in **windowsmanual.pdf**. You may also need to edit the web server configuration file **/var/www/html/api/dicom/config.php** to set the IP address of the machine. If wrong some 3<sup>rd</sup> party viewers functions will not function.

After copying the files, if needed, regenerate the database with “conquest/dgate -v -r” then run the server with “conquest/dgate -v b” or “conquest/dgate -^serverstatus.log”. NOTE: regeneration is only needed after an upgrade if **dicom.sql** is updated. If you want to avoid regeneration do NOT replace **dicom.sql**

The build process for the server was tested with several Linux versions. Both 32 and 64 bit OS's are supported. Warnings are produced but these do not impact server operation.

These are the settings in dicom.ini for MySQL:

```
SQLHost      = localhost
SQLServer    = conquest
Username     = root
Password     =
Mysql        = 1
DoubleBackSlashToDB = 1
```

For Postgres a copy from **dicom.ini.postgres** to **dicom.ini** would set the following values:

```
SQLHost      = localhost
SQLServer    = conquest
Username     = postgres
Password     = postgres
PostGres     = 1
DoubleBackSlashToDB = 1
UseEscapeStringConstants = 1
```

Installation uses a normalized database (as defined in **dicom.sql**) for most database operations, e.g., by copying **dicom.sql.postgres** to **dicom.sql** and a denormalized database for DbaseIII, e.g., by copying **dicom.sql.dbase** to **dicom.sql**.

## The following are donated scripts by Mark Pearson for start/stop and rotating logfiles and this information is for expert users only:

To install this script (it is in the distribution as nconquest-pacs.sh) do:

```
sudo cp nconquest-pacs.sh /etc/init.d/  
sudo chmod 755 /etc/init.d/nconquest-pacs.sh  
sudo apt-get install authbind  
sudo /etc/init.d/nconquest-pacs.sh start
```

---

```
#!/bin/bash  
#  
# conquest-pacs.sh      SysV init script for Conquest PACS.  
#  
#      Written by Miquel van Smoorenburg <miquels>.  
#      Modified for Debian GNU/Linux by Ian Murdock <imurdock>.  
#      Customized for Conquest by Mark Pearson <markp>  
#  
#      HOME and PACSUSER should be the only variables that may need to be modified.  
#  
PATH=/sbin:/bin:/usr/sbin:/usr/bin  
  
# Modify HOME to suit your environment.  
HOME=/usr/local/conquest  
# This is the user to run as. Modify it if you don't use username conquest.  
PACSUSER=conquest  
  
DAEMON=$HOME/dgate  
INI=$HOME/dicom.ini  
NAME=conquest_pacs.sh  
  
# All defaults here will be overridden by values from $HOME/dicom.ini  
STATUSLOG=$HOME/serverstatus.log  
PORT=104  
DESC="Conquest PACS Server"  
  
STOPPACS=$HOME"/dgate --quit:"  
STARTAS=$DAEMON  
  
test -f $DAEMON || echo "Cannot find $DAEMON" exit 0  
test -f $INI || echo "Cannot find $INI" exit 0  
  
set -e  
  
if grep "TCPPort" $INI > /dev/null ; then  
    PORT=`egrep -i '^*TCPPort *= ' $INI | sed 's/\r//' | awk '{ print $3}'`  
fi  
  
if [ $PORT -le 1024 ]; then  
    test -f /usr/bin/authbind || echo "authbind is needed for access to ports < 1024" exit 0  
    STARTAS="/usr/bin/authbind "  
fi  
  
if grep -is "^ *StatusLog" $INI > /dev/null ; then  
    STATUSLOG=`egrep -i '^*StatusLog' $INI | sed 's/\r//' | awk '{ print $3}'`  
fi  
  
PIDFILE=/var/run/$NAME.$PORT.pid  
if [ $STARTAS = $DAEMON ]; then  
    ARGS=" -^$STATUSLOG"  
else  
    ARGS="$DAEMON -^$STATUSLOG"  
fi  
  
case "$1" in  
    start)  
        if [ -f $HOME/disable_autostart ]; then  
            echo "Not starting $DESC: disabled via $HOME/disable_autostart"  
            exit 0  
        fi  
    fi
```

```

        echo -n "Starting $DESC: "
        start-stop-daemon --start --quiet --pidfile $PIDFILE \
            --chuid $PACSUSER --chdir $HOME --exec $DAEMON \
            --startas $STARTAS --background -- $ARGS
        echo "$NAME."
        ;;
stop)
    echo -n "Stopping $DESC: "
    cd $HOME
    $STOPPACS

    start-stop-daemon --oknodo --stop --quiet --pidfile $PIDFILE \
        --exec $DAEMON -- $ARGS
    echo "$NAME."
    echo
    ;;

restart|force-reload)
    echo -n "Restarting $DESC: "
    start-stop-daemon --stop --oknodo --quiet --pidfile $PIDFILE \
        --exec $DAEMON -- $ARGS
    sleep 1
    start-stop-daemon --start --quiet --pidfile $PIDFILE \
        --chuid conquest --chdir $HOME --exec $DAEMON -- $ARGS
    echo "$NAME."
    ;;
*)
    N=/etc/init.d/$NAME
    echo "Usage: $N {start|stop|restart|force-reload}" >&2
    exit 1
    ;;
esac

exit 0

```

---

For security reasons I have added a user "conquest" and the package authbind to allow access to priveleged ports. I added the following entries to dicom.ini:

```

HomeDir = /usr/local/conquest
StatusLog = /var/log/conquest/NMPACS.serverstatus.log
TroubleLog = /var/log/conquest/NMPACS.PacsTrouble.log

```

The file /etc/cron.weekly/conquest\_rotate does weekly log rotation for me.

---

```

#!/bin/bash

# conquest_rotate      Cron script to rotate conquest log files.
#   Keep files for 365 days
#   Read filenames from dicom.ini
#
#
#           Written by Mark Pearson 20070711 <markp>.
#
# Modify this line to suit your environment
HOMES=(/usr/local/conquest /usr/local/conquest-icon)
for i in ${HOMES[@]}; do

    INI=${i}/dicom.ini
    STATUSLOG=${i}/serverstatus.log
    TROUBLELOG=${i}/PacsTrouble.log

    set -e

# defaults will be overridden by values from ${i}/dicom.ini
    if grep -is "^ *StatusLog" $INI > /dev/null ; then
        STATUSLOG=`egrep -i '^ *StatusLog' $INI | sed 's/\r//' | awk '{ print $3}'`
    fi
    if grep -is "^ *TroubleLog" $INI > /dev/null ; then
        TROUBLELOG=`egrep -i '^ *TroubleLog' $INI | sed 's/\r//' | awk '{ print $3}'`
    fi

```

```
if [ -s $TROUBLELOG ]; then
    savelog -p -c 365 -n -q $TROUBLELOG
fi

if [ -s $STATUSLOG ]; then
    savelog -p -c 365 -n -q $STATUSLOG
fi

done
```

---

This copes with multiple pacs instances on the same host. The advantage of using savelog is that old logfiles are compressed. It should be quite simple to edit the files to have executable or log in /opt. Also, don't forget to set the appropriate file permissions for the user that runs conquest.

Finally, Here are the command lines to compile the server under OS X xcode using 10.4u sdk on a PowerPC (not recently tested):

```
g++ -isysroot /Developer/SDKs/MacOSX10.4u.sdk -arch ppc -Wno-multichar -I/usr/local/mysql/include -L/usr/local/mysql/lib -DDARWIN
-DUSEMYSQL -DHAVE_LIBJASPER -DHAVE_LIBJPEG -DB_DEBUG -o dgate total.cxx -lpthread -lgcc_s.10.4 -lstdc++.6 -lmysqlclient -lz
```

And to compile under SOLARIS 10:

```
/usr/sfw/bin/g++ -DUNIX -DNATIVE_ENDIAN=1 -DHAVE_LIBJASPER -DHAVE_LIBJPEG -DSOLARIS total.cxx -o dgate -lpthread -lsocket -lnsl
-lposix4
```