

---

## Advanced Tips and Tricks

Eric Bosch and Martin Renters  
Clinical DataFax Systems Inc.

---

## Workshop Format

- **Several encapsulated scenarios**
  - What is the problem/goal?
  - Problem analysis
  - Possible solutions
  - Work-through of one complete solution
  - Demonstration
- **You can use these at home!**

## Common Characteristics

---

- Bourne shell
- Example study is 254
- short scripts
- Minimal error checking

## ESN: Goal

---

- Determine all enabled study numbers
- Efficiently!

## ESN: Analysis

---

- Determine all study numbers

- Examine DFstudies.db

```
% awk -F\| '{print $1}' DFstudies.db
```

## ESN: Analysis

---

- Is a particular server enabled?

- Query server directly

```
% DFgetparam.rpc -s254 HOSTNAME
```

```
purgatory%echo $?
```

```
0
```

problem: verbose, need to re-direct output

```
% DFgetparam.rpc -s254 HOSTNAME > /dev/null 2>&1
```

side-effect: starts server!

- Examine DFstudies.db

```
% awk -F\| '$2 !~ /^-/ {print $1}' DFstudies.db
```

## ESN: Solution

---

- DFstudies.db has everything we need

```
% cat $DATAFAX_DIR/lbin/ESN
#!/bin/sh
awk -F\| '$2 !~ /^-/ {print $1}' \
$DATAFAX_DIR/lib/DFstudies.db
```

## ESN: Application

---

- Get status of all studies

```
% cat $DATAFAX_DIR/lbin/ESNapp
#!/bin/sh
for snum in ` $DATAFAX_DIR/lbin/ESN `
do
DFstatus -cs $snum
done
```

## Perms: Goal

---

- cron script that monitors permissions on all study directories

## Perms: Analysis

---

- We need to run DFstudyPerms on each study
  - should DFstudyPerms fix problems?
- We need to get a list of studies
- We need to figure out how cron works

## Perms: Analysis (con't)

---

- Cron
  - Runs programs periodically, at specified times.
  - Useful for backups and system admin functions
  - Uses a 'crontab' configuration file to specify times commands are to be run
  - Command output sent as email

## Perms: Analysis (con't)

---

- Crontab file format (6 fields):
  - minute (0-59)
  - hour (0-23)
  - day of month (1-31)
  - month (1-12)
  - weekday (0-6, 0-Sunday)
  - command to execute

## Perms: Analysis (con't)

---

- Example: We want to run 'prog' at 2AM on the 1<sup>st</sup> and 15<sup>th</sup> of each month

```
0 2 1,15 * * prog
```

- The '\*' indicates that any value matches
- Use your favorite editor to construct a file containing lines in the format above.
- The '#' symbol can be used for comments

## Perms: Analysis (con't)

---

- Use the command 'crontab filename' to submit your crontab file for processing
- Use 'crontab -l' to list the contents of your crontab file

## Perms: Solution

---

- Running DFstudyPerms on a single study is easy

```
DFstudyPerms 254
```

## Perms: Solution (con't)

---

- We can use ESN to get a list of studies

## Perms: Solution (con't)

---

- Put them together:

```
#!/bin/sh
DATAFAX_DIR=/opt/datafax;export DATAFAX_DIR
for s in ` $DATAFAX_DIR/lbin/ESN `
do
    echo "Checking permissions for Study $s"
    DFstudyPerms $s
done
```

- We need to define DATAFAX\_DIR because cron doesn't initialize it

## Perms: Solution (con't)

---

- Add an entry to your crontab file containing:

```
0 2 * * * /export/home/martin/check_study_perms
```

- submit it via crontab  
`crontab mycrontabfile`
- wait for the email to appear at 2AM

## Perms: Issues

---

- The ability to run cron jobs may be restricted by your sysadmin
- If you specify a day of the week in your crontab entry, remember that after midnight is a new day

## Serialization: Goal

---

- Serializing DF\_QCupdate, DF\_QCreports, and DF\_QCfax in a cron job

## Serialization: Analysis

---

- We want to make sure that DF\_QCupdate finishes successfully before running DF\_QCreports
- If any errors are encountered, we want to terminate
- We can do this by checking the return status of each of the commands

## Serialization: Solution

---

```
#!/bin/sh
DATAFAX_DIR=/opt/datafax; export DATAFAX_DIR

$DATAFAX_DIR/reports/DF_QCupdate 254
if [ $? -ne 0 ]; then
    echo "QC generation failed!"
    exit 1
fi
$DATAFAX_DIR/reports/DF_QCreports 254
if [ $? -ne 0 ]; then
    echo "QC reports failed!"
    exit 1
fi
$DATAFAX_DIR/reports/DF_QCfax 254
```

## PRO: Goal

---

- Publish report output from selected reports to intranet
- Give access to DataFax reports output without a DataFax interface/license

## PRO: Analysis

---

- Input: list of reports
  - will include a description for good measure
- Run each report and format to HTML
  - simple HTML
- Use ESN to apply to all studies
- Output: requires a repository

## PRO: Caveats

---

- **Permissions**
  - DataFax issue: program should be run as user 'datafax' to ensure that all requested studies are permitted
  - Client issue: include only reports that you would like everyone to see output of
- **Timing**
  - output is a 'recent' snapshot
  - can use cron to improve 'recency'

## PRO: Solution

---

- **Input: list of reports (could be any command)**

```
% cat $DATAFAX_DIR/llib/PRO.cf
Database Status|bin/DFstatus -cs$STUDY
User Permissions|utils/DFuserPerms -l :$STUDY
Patient CRF info|reports/DF_PTcrfs $STUDY
Patient Scheduling|reports/DF_PTschedule $STUDY
QC Fax Log|reports/DF_QCfaxlog $STUDY
QC Reports Status|reports/DF_QCstatus
```

## PRO: Solution

---

- Two scripts
  - [1] create index page and run [2] for each study
  - [2] run each report and format to HTML

## PRO: Solution – Script[1]

---

```
% cat PRO
#!/bin/sh
DATAFAX_DIR=/opt/datafax; export DATAFAX_DIR
REPOSITORY=/opt/www/studies; export REPOSITORY
(
echo "<HTML><TITLE>PRO</TITLE><BODY>"
for snum in ` $DATAFAX_DIR/lbin/ESN `; do
mkdir $REPOSITORY/$snum 2> /dev/null
$DATAFAX_DIR/lbin/PROstudy $snum $REPOSITORY
echo "<DIV><A HREF=\" \"$snum\"/PRO.html\">"
echo `DFgetparam.rpc -s $snum STUDY_NAME`
echo "</A></DIV>"
done
echo "</BODY></HTML>"
) > $REPOSITORY/PROindex.html
```

## PRO: Solution – Script [2]

---

```
% cat PROstudy
#!/bin/sh
STUDY=$1; export STUDY
REPOSITORY=$2; export REPOSITORY
INPUT=$DATAFAX_DIR/l1lib/PRO.cf
(
echo "<HTML><TITLE>PRO for" $STUDY "</TITLE><BODY>"
nawk -F\| '{
printf "<DIV><A HREF=\"#%d\">%s</A></DIV>\n", NR, $1
}' $INPUT
nawk -F\| '{
printf "<H1><A NAME=\"%d\">%s</A></H1>\n", NR, $1
printf "<DIV><PRE>\n"
system( "$DATAFAX_DIR/"$2 )
printf "</PRE></DIV>\n"
}' $INPUT
echo "</BODY></HTML>"
) > $REPOSITORY/$STUDY/PRO.html 2>&1
```

## PRO: Solution

---

- cron statement to execute PRO 3 times daily

```
0 8,12,16 * * * /opt/datafax/lbin/PRO
```

## PDD: Goal

---

- Publish DataFax data to intranet
- Give access to DataFax data without a DataFax interface/license

## PDD: Analysis

---

- Similar to Tip of the Week 1999/08/30
- Simple tabular listings of all plates
  - get all raw (primary) data
- Format to HTML
  - tabular HTML
  - sortable rows
- Use ESN to apply to all studies
- Output: requires a repository

## PDD: Analysis

---

- DFexport.rpc gets the data out
  - variable names too!
- script to do first pass at HTML
- Microsoft TDC does the remaining HTML for us
  - this only works with Internet Explorer 4+

## PDD: Solution[1]

---

- Get the data and variable names for each plate

```
% cat PDDstudy
#!/bin/sh
STUDY=$1; export STUDY
DIR=$2/$STUDY; export DIR
for p in `DFlistplates.rpc -s $STUDY`; do
  DFexport.rpc -s primary -h $STUDY $p - |\
  sed 's/|$/' > $DIR/$p.exp
done
```

## PDD: Solution[2]

---

- HTML index page

```
(
echo "<HTML><TITLE>PDD for" $STUDY \
"</TITLE><BODY>"
for p in `Dflistplates.rpc -s $STUDY`; do
echo "<DIV><A HREF=\"PDD$p.html\">Plate $p</A></DIV>"
done
echo "</BODY></HTML>"
) > $DIR/PDD.html 2>&1
```

## PDD: Solution[3]

---

- HTML tables

```
for p in `Dflistplates.rpc -s $STUDY`; do
(
head -1 $DIR/$p.exp | \
nawk -F\| 'BEGIN{plt=sprintf( "%03d", '$p' )} {
print "<HTML><HEAD><SCRIPT LANGUAGE=\"VBSCRIPT\">"
printf "function Sort(name)\nplt%s.Sort=name\n", plt
printf "plt%s.Reset()\nend function\n", plt
print "</SCRIPT></HEAD><BODY>"
printf "<OBJECT ID=\"plt%s\" \n", plt
print "CLASSID=\"clsid:333C7BC4-460F-11D0-BC04-0080C7055A83\">"
printf "<PARAM NAME=\"DataURL\" VALUE=\"%s.exp\">\n", plt
print "<PARAM NAME=\"UseHeader\" VALUE=\"true\">"
print "<PARAM NAME=\"FieldDelim\" VALUE=\"|\">"
```

## PDD: Solution[4]

---

```
printf "</OBJECT>\n<TABLE BORDER=1 DATASRC=\"#plt%s\">\n", plt
print "<THEAD><TR>"
for ( i = 1; i <= NF; i++ )
  printf "<TH><DIV onClick=\"Sort(%c%s%c)\">%s</DIV></TH>\n", \
    39, $i, 39, $i
print "</TR></THEAD><TBODY><TR>"
for ( i = 1; i <= NF; i++ )
  printf "<TD><SPAN DATAFLD=\"%s\"></SPAN></TD>\n", $i
print "</TR></TBODY></TABLE></BODY></HTML>"
}'
) > $DIR/PDD$p.html
done
```

## PDD: Solution

---

- create index page + run for each study

```
% cat PDD
#!/bin/sh
DATAFAX_DIR=/opt/datafax; export DATAFAX_DIR
REPOSITORY=/opt/www/studies; export REPOSITORY
(
echo "<HTML><TITLE>PDD</TITLE><BODY>"
for snum in ` $DATAFAX_DIR/lbin/ESN `; do
  mkdir $REPOSITORY/$snum 2> /dev/null
  $DATAFAX_DIR/lbin/PDDstudy $snum $REPOSITORY
  echo "<DIV><A HREF=\"\"$snum\"/PDD.html\">"
  echo `DFgetparam.rpc -s $snum STUDY_NAME`
  echo "</A></DIV>"
done
echo "</BODY></HTML>"
) > $REPOSITORY/PDDindex.html
```

## PDF: Goal

---

- simple tool for assembling DataFax CRFs into one or more PDFs

## PDF: Analysis

---

- ImageMagick is one of many tools that can already do this
  - but it can't bookmark pages!
- Need DataFax-aware bookmarking
- Simple means of specifying CRFs to include
  - DataFax Retrieval File
  - DFexport.rpc file

## PDF: Solution

---

- DFpdf
  - available though Early Access Program
  - usage

```
-s studynum
-d
[-i infile]
[-o outfile]
```
  - example

```
% DFpdf -ds 254 \
-i /opt/studies/val254/work/sample.drf \
-o /opt/www/studies/254/sample.pdf
```

## PDF: Solution

---

- What else should this do?
  - Wednesday AM workshop topic

## Pager: Goal

---

- How can we get DataFax to notify us via a pager (or PCS phone) that an AE event has occurred?

## Pager: Analysis

---

- HylaFax supports sending pages to pagers via the IXO/TAP protocol which most paging companies support
- `sendpage` is the HylaFax command to do this
- We can call `sendpage` using an edit check

## Pager: Solution

---

- Configure HylaFax paging support

```
# cd /var/spool/fax/etc
# vi pagermap
```

- Add in entries for each user who we might want to page. We need to know the TAP access number and the pager PIN (from your pager company)

The format is:

```
Username    TAPACCESS/PIN
```

e.g.

```
martin     18883223436/9055223282
```

## Pager: Solution (con't)

---

- Next we need to write an edit check to send us the page

```
edit sendAepage()
{
    dfsystem("/opt/hylafax/bin/sendpage
            -q -p martin 'AE event occurred for
            patient ", PID, "'");
}
```

- The `-q` option tells HylaFax to queue and not wait for page completion

## Pager: Issues

---

- TAP servers that don't hang up after a page is submitted require a small patch to HylaFax

## ShellEC: Goal

---

- Returning results from a shell command to a DataFax edit check
- Useful for calculating check digits, complex calculations, retrieving system information, etc.

## ShellEC: Analysis

---

- The edit checks language contains a function called `dfrun()` which executes a shell command and returns the first 1024 bytes of the result

## ShellEC: Solution

---

- In your edit check do:

```
edit unixcmd()  
{  
    string s;  
    s = dfrun("verify-check-digit ", PID);  
    dfmessage("The value returned was ", s);  
}
```

## ShellEC: Issues

---

- You cannot return more than 1024 characters
- Assign the result to a local string variable
- If you return results containing '|' characters, you'll need to split the fields apart with `dfgetfield()`
- Don't return new-line characters

## EDE: Goal

---

- Export DataFax data for Excel import

## EDE: Analysis

---

- Do not want to be prompted for import parameters
- Excel's 'native' text format
  - field delimiter is comma
  - text delimiter is double-quote
  - filename has .csv extension

## EDE: Analysis

---

- Field contains comma
  - insert field with text delimiters  
`1|left,right|2 -> 1,"left,right",2`
- Field contains double-quote
  - replace double-quote with double double-quote  
`1|"severe" trauma|2 -> 1,"""severe"" trauma",2`

## EDE: Solution

---

```
% cat EDE
DFexport.rpc -cs primary $STUDY $plt - |\
nawk -F\| '
function doF(f){
    flag = 0
    if ( index( f, "\"" ) )
        { flag = 1; gsub( /"/, "\"\", f ); }
    if ( index( f, "," ) ) flag = 1;
    if ( flag == 1 ) printf "\"%s\"", f
    else printf "%s", f
}
for ( i = 1; i < NF; i++ )
    { doF( $i ); printf "," }
doF( $NF )
printf "\r\n"
}' > $plt.csv
```

## EDE: Issues

---

- date format
- doesn't handle missing 'very well'

```
* -> char('*')
```

## FaxDispatch: Goal

---

- Forwarding faxes via email with FaxDispatch

## FaxDispatch: Analysis

---

- FaxDispatch is a HylaFax file that lives in /var/spool/fax/etc/FaxDispatch
- It allows faxes to automatically be emailed based on the sender's phone number
- Email is sent as a PostScript file
- Process occurs before DataFax
  - no study information is available
  - includes all 'non-DataFax' faxes

## FaxDispatch: Solution

---

- Suppose we want to copy all faxes received from 905 522 7284 to 'eric@datafax.com'
- Create a FaxDispatch file (by default there is none) with the following lines:

```
case "$SENDER" in
*905*522*7284*) SENDTO="eric@datafax.com" ;;
*) SENDTO=" " ;;
esac
```
- The '\*'s match anything and allow for non-numeric in the sender id

## MultiFax: Goal

---

- Fax redundancy with two or more HylaFax servers
- Remote fax servers

## MultiFax: Analysis

---

- We can use the FaxDispatch script to send faxes via email to the DataFax server
- This allows us to also have a remote collection site and use the internet to route faxes, saving long-distance charges

## MultiFax: Analysis (con't)

---

- DataFax expects emailed images to be in TIFF format
- HylaFax receives images in TIFF format so all we need to do is MIME encode the TIFF files and send them to `datafax@yourserver.com`

## MultiFax: Solution

---

- A great little program for MIME encoding files is 'mpack'
- In your FaxDispatch file you can do something like:

```
mpack -s "FAX Image" -c "image/tiff" $FILE  
datafax@yoursite.com >/dev/null 2>&1
```

## MultiFax: Issues

---

- Is redundancy really possible?
  - If you decide to use this setup for redundancy, remember that your 1-800 number is going to start at a single phone line and if that server doesn't answer, it won't roll over to the other server

## App-defaults: Goal

---

- Customizing your DataFax installation via the app-defaults mechanism

## App-defaults: Analysis

---

- What types of things could we want to change?
  - size of DFvalidate panes for different screen sizes
  - defaults for Internal/External QCs
  - which buttons are enabled by default in DFstatus, DFqc
  - where status windows (like the record list) appear

## App-defaults: Analysis (con't)

---

- You can see what options can be set by looking at the files in `$DATAFAX_DIR/app-defaults`

## App-defaults: Solution

---

- System wide app-defaults are stored in `$DATAFAX_DIR/app-defaults`
- User defaults are stored in `$HOME/.Xdefaults`

## App-defaults: Solution (con't)

---

- To make the status tool come up with the database status button enabled, add the following to your `.Xdefaults` file:

```
DFstatus*databaseStatusToggle.set: True
```

- Make sure there are no trailing spaces

## App-defaults: Issues

---

- `$DATAFAX_DIR/app-defaults` is owned by CDSI and will be overwritten when patches or updates are installed
- Use user-level overrides when possible
- `.Xdefaults` are read at login time
  - logout/login to apply changes or
  - `/usr/openwin/bin/xrdb -override .Xdefaults`

## Quick Tips

---

- Backups: put \$DATAFAX\_DIR/archive and \$STUDY\_DIR/pages on different backup media
  - if one media fails, you still have the other
- DFexport.rpc can split time stamps into date/time pairs

```
% DFexport.rpc -G "DFmodify:2x8w" 254 1 -  
00/01/14|17:25:36
```