

---

## Edit Checks

Wayne Taylor  
Clinical DataFax Systems Inc.

### What can be accomplished?

---

- Test data items:
  - consistent with other items, legal, missing
- Display messages to guide data entry
- Create and pop-up QC notes
- Auto-fill data fields
  - compute values, use look-up tables
- Run a UNIX program
  - e.g. mail someone a message

### Example: Testing Fields

---

```
edit checkdates() {  
  # date1 should precede date2  
  string ok ;  
  if( date1 < date2 ) ok="yes" ;  
  else ok="no" ;  
}
```

### Example: Displaying a Message

---

```
edit checkdates() {  
  # date1 should precede date2  
  # show any discrepancies  
  if( date1 >= date2 )  
    dferror(date1," should precede ",date2) ;  
}
```

## Example: Creating a QC Note

---

```
edit checkdates() {  
# date1 should precede date2  
# if a discrepancy occurs create a QC note  
string m, s ;  
if( date2 <= date1 ) {  
    m = "Date error: date1= " + (s=date1)  
      + " and date2= " + (s=date2) ;  
    dfaddqc(date2, 3, m,1, 2, "");  
}  
}
```

## When are Edit Checks executed ?

---

- **Interactively in the Validation tool**
  - on entry to a plate
  - on entry to a specific data field
  - on exit from a plate
  - on exit from a specific data field
- **In Batch Mode** (see early access program)
  - Run from the command line
  - Run automatically as a cron job

## How are they created ?

---

- **Edit Check language**
  - limited subset of C style syntax
  - DataFax functions, e.g. dferror, dfaddqc, dfask
  - see Edit Check chapter in programmers manual
- **Edit Check Worksheets**
  - planning: to clarify edit check requirements
  - testing: to create a test plan and document results
  - see [www.datafax.com](http://www.datafax.com) : current documentation set

## Where are they stored ?

---

- **Setup tool**
  - Edit Checks editor accessed from the Study menu
- **~/lib/DFedits**
  - plain text file
- **~/anywhere/anyfilename**
  - general edit checks can be written and included by reference in each study DFedits file
- **Batch Files**

## Recommended Steps

---

- **Plan It:**
  - Name, location, when does it execute
  - Data fields to be used
  - Exceptions: missing, blank, illegal, QCs
  - Restrictions: users, levels, visits, etc.
  - Tests & Actions
- **Enter It:**
  - Setup tool editor, check syntax
- **Test It:**
  - use a test plan
  - test boundary conditions and missing, blank and illegal values

## Example Test Plan

---

Fields and Values	Expected Results	Passed or Failed
Age BirthDate VisitDate		
40 1/Jan/60 1/Jan/00	nothing	passed
40 1/Jan/60 6/Jun/99	dialog pop-up with age=39	passed
40 1/Jan/60 blank,*	nothing	passed
tested by: D.W. Taylor    date: Jan 12,2000		

## Elements of Programming Style

---

Example:

- Set patient initials equal to the initials on plate 1 at visit 0.
- This saves a few keystrokes
- More importantly it allows the user to check that the initials match those on the first data entry CRF page.

## Style - comment your code

---

```
edit SetPinit() {  
# on entry to initials field bring forward  
# initials from plate 1, visit 0  
# field entry edit check on PatInitials style  
  
    @T = PINIT[,0,1] ;  
  
}
```

## Style - consider exceptions

---

```
edit SetPinit() {  
# on entry to initials field bring forward  
# initials from plate 1, visit 0  
# field entry edit check on PatInitials style  
  if( !dfblank(@T) ) exit;  
  if( dflmissing(PINIT[,0,1]) exit;  
  if( dfanyqc(PINIT[,0,1]) exit;  
  @T = PINIT[,0,1] ;  
}
```

## Style - consider restrictions

---

```
edit SetPinit() {  
# on entry to initials field bring forward  
# initials from plate 1, visit 0  
# field entry edit check on PatInitials style  
  if( DFSTATUS!=0 ) exit;  
  if( !dfblank(@T) ) exit;  
  if( dflmissing(PINIT[,0,1]) exit;  
  if( dfanyqc(PINIT[,0,1]) exit;  
  @T = PINIT[,0,1] ;  
}
```

## Style - consider FDA Guidance

---

```
edit SetPinit() {
# on entry to initials field bring forward
# initials from plate 1, visit 0
# field entry edit check on PatInitials style
  string m = "Set Patient Initials to " + PINIT[,0,1] ;
  if( DFSTATUS!=0 ) exit;
  if( !dfblank(@T) ) exit;
  if( dflmissing(PINIT[,0,1]) exit;
  if( dfanyqc(PINIT[,0,1]) exit;
  if( dfask(m,1,"no","yes") == 2 ) @T = PINIT[,0,1] ;
}
```

## Checking Initials in Batch Mode

---

```
edit BatchCheckInitials() {
# batch check consistency of patient initials
# write a useful error message
  string m = "Different Initials: visit 0 plate 1 = " ;
  if( !dfbatch() ) exit ;
  if( @T != PINIT[,0,1] ) {
    m = m + PINIT[,0,1]
      + " visit ", @[6], " plate ", @[5], " = ", @T ;
    dferror(m) ;
  }
}
```

## Checking Visit Dates

---

### Why?

- VisitDate errors can result in errors in overdue visit and next visit calculations.
- A simple check that visit dates are in the expected order will catch some data entry errors.
- To be useful the error message needs to say more than "VisitDate Error".

## Checking Visit Dates

---

```
edit CheckVisitDates() {
# check order of visit dates across visits 1 to 6
# field exit edit check attached to the VisitDate style
number v=1, flag=0;
string s, m = "Visit Dates are not in order:";
group vd = vdt[,1,5], vdt[,2,5], vdt[,3,5], vdt[,4,5], vdt[,5,5], vdt[,6,5];
if( dfmissing(@T) ) exit ;
while( v <= 6 ) {
  m = m + "\nVisit " + (s=v) + " " + (s=vdt[v]);
  if( v<@[6] && !dfmissing(vd[v]) && vd[v] >= @T ) { flag=1 ; m = m + " ?"; }
  if( v>@[6] && !dfmissing(vd[v]) && vd[v] <= @T ) { flag=1 ; m = m + " ?"; }
  v = v + 1 ;
}
if( flag==1 ) dferror("Visit Date Problem(s)\n",m) ;
}
```

## Checking Visit Dates

---

Example message displayed in a pop-up dialog box during data entry:

Visit Dates are not in order:

Visit 1 01/15/99

Visit 2 01/22/99

Visit 3 03/28/99

Visit 4 02/08/99 ?

Visit 5 \*

Visit 6 \*

OK

## Using Look-up Tables for Pick Lists

---

- A pick list is a file containing a sorted list of all possible response options.
  - investigator signatures, affiliations, ...
  - medications, treatment options, ...
  - occupations, states, ...
- On exit from a field with a pick list:
  - display the picklist if the value in the field is not in the picklist

## Defining a Look-up Table

---

- Create the file  
e.g. /studies/mystudy/lib/investigators.lut  
Barber, Jim  
Hay, John  
Sibley, Jack
- Register it in ~/lib/DFlut\_map  
investigators|investigators.lut

## InvestigatorPickList()

---

```
edit InvestigatorPickList() {  
# field exit edit check on Investigator style  
string s ;  
# look for an exact match, if found quit  
s = dflookup("investigators",@T,"",-1) ;  
if( !dfblank(s) ) return ;  
# display pick list. If user makes a selection  
# enter it into the investigator name field  
s = dflookup("investigators",@T,"",4) ;  
if( !dfblank(s) ) { @T=s ; return ; }  
}
```

## Investigator Pick List 2

---

### Extension:

- If user does not pick a value from the pick list and leaves the field blank add a missing value QC note.
- If user does not pick a value from the pick list and leaves some other value in the signature field add an illegal value QC note.

## InvestigatorPickList2()

---

```

edit InvestigatorPickList() {
# field exit edit check on Investigator style
  string s, m ;
# look for an exact match, if found quit
  s = dflookup("investigators",@T,"",-1) ;
  if( !dfblank(s) ) return ;
# display pick list, for user selection
  s = dflookup("investigators",@T,"",4) ;
  if( !dfblank(s) ) { @T=s ; return ; }
  if( dfblank(@T) ) dfaddqc(@T,1,"",1,2,"") ;
  else { m = @T + " is not a registered investigator." ;
        dfaddqc(@T,2,m,1,2,"") ; }
}

```

## Using a Look-up Table to Code Visit Numbers

---

### Objectives:

- To overcome an error made in designing the CRFs
  - visit number was bar coded but should have been a choice field
  - one of the possible choices was omitted and is being “written in” by investigators
- To allow for additional follow-up visits on CRFs added after the study started.

## Setup Tasks 1 of 2

---

- Setup Tool
  - Plate-Edit: set “Sequence is In the First Data Field”
  - If the seq number was previously in the bar code:
    - create a new data field for the seq number
    - reorder the fields to make it field 6
  - Fix legal values and format
  - Make sure the seq field is defined as required
  - Add field entry edit check: e.g. plt5seq

## Setup Tasks 2 of 2

---

- Create the lookup file  
e.g. /studies/mystudy/lib/plt5seq.lut  
baseline visit|0  
optional 2 week visit|1  
30 day visit|2  
3 month visit|3
- Register it in ~/lib/DFlut\_map  
plt5seq|plt5seq.lut

## Code Seq Number on Plate 5

---

```
edit plt5seq() {  
  # field entry edit check attached to field 6 on plate 5  
  string s ;  
  # exit if seq number has already been coded  
  if( dflegal(@T) ) return ;  
  # otherwise display lookup table for user choice  
  s = dflookup("plt5seq1",@T,"",0) ;  
  if( !dfblank(s) ) @T=s ;  
}
```

## plt5seq() version 2

---

```
edit plt5seq() {
  # field entry edit check attached to field 6 on plate 5
  string s ;
  # do not allow this edit check in batch mode
  if( dfbatch() ) exit ;
  # force user to make a choice from the lookup table
  while( !dflegal( @[6] ) ) {
    s = dflookup("plt5seq",@[6],"",0) ;
    if( !dfblank(s) ) @[6]=s ;
    else dferror("You must make a selection.") ;
  }
}
```

## Double Data Entry

---

- DataFax includes DFdde for double data entry (see programmers manual).
- DFdde uses 3rd party arbitration.
- Could double entry without 3rd party arbitration be implemented using edit checks?

## Double Data Entry without 3rd party arbitration

---

### Objectives:

- double entry is performed by specified staff when validating level 1 records up to level 2.
- on plate entry, if plate is level 1, blank all fields
- on exit from each data field check for discrepancies between the entry and exit values, and if a discrepancy occurs:
  - (1) pop-up a dialog showing the discrepancy
  - (2) ask the user to select the correct value
  - (3) enter the selected value - i.e. 2nd entry clerk wins
  - (4) log the discrepancy and it's resolution

## DDEset

---

```
edit DDEset() {
  # Double entry is performed by jim or sue when
  # validating plates at level 1 up to level 2.
  # On entry to a plate that is at level 1 this edit check
  # blanks all fields in preparation for DDE.
  # Add this edit check to the patient ID style.
  number f=7;
  if( dflevel()!=2 || DFVALID!=1 ) exit;
  if( dfuser()!="jim" && dfuser()!="sue") exit;
  while( dfvarinfo(@[f], DFVAR_GENERIC)
    != "DFSCREEN" ) { @[f]=""; f = f + 1; }
}
```

## DDEcheck

---

```

edit DDEcheck() {
# on field exit compare entry and exit values
# attach this edit check to every data field, or style
  number f = dfvarinfo(@T,DFVAR_FLDNUM) ;
  number v = dfvarinfo(DFVALID,DFVAR_ENTER_VALUE) ;
  if( dflevel()!=2 || v!=1 ) exit ;
  if( dfuser()!="jim" && dfuser()!="sue") exit ;
  if( FieldChange(f) == 1 ) FieldResolve(f) ;
}

```

## function FieldChange

---

```

number FieldChange(number f) {
# return 1 if original and double entries differ
# return 0 if original and double entries are the same
  string e1 = dfvarinfo(@[f], DFVAR_ENTER_VALUE) ;
  string type = dfvarinfo(@[f], DFVAR_TYPE) ;
  string m = dfmisscode(@[f]) ;
  number n ;
# use direct comparison for missing code,date or string
  if( !dfblank(m) && m!=e1 ) return(1) ;
  if( type=="string" || type=="date" ) {
    if( @[f] != e1 ) return(1) ; }
# use numerical comparison for all other cases
  else if( @[f]!=(n=e1) ) return(1) ;
  return(0) ;
}

```

## function FieldResolve

---

```
number FieldResolve(number f) {
  # ask user which version was correct and enter it
  string e1= dfvarinfo(@[f],DFVAR_ENTER_VALUE), e2= @[f] ;
  string m = "DDE Discrepancy for " + dfvarinfo(@[f],DFVAR_GENERIC)
    + "\nSelect Correct Value Below.\n" ;
  string lbl1, lbl2 ;
  if(dfblank(e1))      lbl1= "blank" ;
  else                lbl1= dfvarinfo(@[f],DFVAR_LABEL,e1) ;
  if(dfblank(lbl1) )  lbl1 = e1 ;
  if(dfblank(e2))      lbl2 = "blank" ;
  else                lbl2 = dfvarinfo(@[f],DFVAR_LABEL,e2) ;
  if( dfblank(lbl2) )  lbl2 = e2 ;
  if( dfask(m,2,lbl1,lbl2) == 1 ) @[f] = e1 ;
}
```

## Recording True Processing Times

---

- DF\_WFcrfs - uses inter-page sign-off times and thus can not determine the time required to process the first plate in each set.
- A plate entry edit check could set plate entry time in a global variable.
- A plate exit edit check could compare the exit time to the entry time to accurately calculate plate processing time.

## PlateEntryTime()

---

```
#global variables for plate entry and exit time  
string time1, time2;
```

```
PlateEntryTime() {  
# set time at which plate appeared on the screen  
# plate entry edit check on the patient ID style  
    time1 = dfrun("date +%H:%M:%S | tr -d '\n'");  
}
```

## PlateExitTime

---

```
PlateExitTime() {  
# journal time spent reviewing the record  
# plate exit edit check on the patient ID style  
    time2 = dfrun("date +%H:%M:%S | tr -d '\n'");  
    WFjnl( deltatime(time1,time2) );  
}
```

## function deltatime()

---

```

number deltatime(string time1, string time2) {
# return the number of seconds between 2 times
  number sec1,sec2, h1,m1,s1, h2,m2,s2 ;
  h1=dfgetfield(time1,1,":"); h2=dfgetfield(time2,1,":");
  m1=dfgetfield(time1,2,":"); m2=dfgetfield(time2,2,":");
  s1=dfgetfield(time1,3,":"); s2=dfgetfield(time2,3,":");
  sec1 = h1*60*60 + m1*60 + s1 ;
  sec2 = h2*60*60 + m2*60 + s2 ;
  if( sec1 <= sec 2 ) return( sec2-sec1 ) ;
  else return( 24*60*60 - sec1 + sec 2 ) ;
}

```

## function WFjnl()

---

```

number WFjnl(number seconds) {
# record time spent reviewing the record
  string when = dfrun("date +%Y%m | tr -d '\n'");
  string file = "/studies/mystudy/WFjnl/" + when + ".jnl" ;
  string jdate = dfrun("date +%Y/%m/%d | tr -d '\n'");
  string s, d="|";
  string record = dfwhoami() + d + jdate + d + time2 + d
    + (s=seconds) + d + (s=DFPLATE) + d
    + (s=DFSTATUS) + d + (s=DFVALID) ;
  dfopen(1,file,"a") ; dfwrite(1,record) ; dfclose(1) ;
}

```

## Work Flow Journal Version 2

---

### Objectives:

- Keep a separate work flow journal for each staff member.
- Record all record commits across all DataFax studies in each users journal.
- Start a new journal for each user for each month of the year.

## WFjnl() version 2

---

```

number WFjnl(number seconds) {
# record time spent reviewing a data record
string xxx = dfrun("date +%Y%m | tr -d '\n'") + ".jnl" ;
string file = "/studies/WFjnl/" + dfwhoami() + "/" + xxx ;
string jdate = dfrun("date +%Y/%m/%d | tr -d '\n'") ;
string s, d="|" ;
string record = (s=DFSTUDY) + d + jdate + d + time2 + d
                + (s=seconds) + d + (s=DFPLATE) + d
                + (s=DFSTATUS) + d + (s=DFVALID) ;
dfopen(1,file,"a") ; dfwrite(1,record) ; dfclose(1) ;
}

```

## Critique

---

- What's wrong with this edit check?  

```
PlateExitTime() {
# journal time spent reviewing the record
# plate exit edit check on the patient ID style
time2 = dfrun("date +%H:%M:%S | tr -d '\n'");
WFjnl( deltatime(time1,time2) );
}
```
- If a user selects clean when the record is dirty the record commit is blocked and the user will have to sign off again. Thus the edit check may run more than once for the same record.

## PlateExitTime - version 2

---

```
PlateExitTime() {
# journal time spent reviewing the record
# plate exit edit check on the patient ID style
# check for illegal fields and unresolved QCs when record is
# being signed off as clean at levels 1-6.
number n = dfvarinfo(DFSCREEN,DFVAR_FLDNUM) ;
number ok = 1;
if( DFSCREEN==1 && DFVALID<7 ) {
while( n>=6 && dflegal(@[n]) && !dfunresqc(@[n]) ) n=n-1 ;
if( n>=6 ) ok=0 ;
}
if( ok==1 ) {
time2 = dfrun("date +%H:%M:%S | tr -d '\n'");
WFjnl( deltatime(time1,time2) );
}
}
```

## Journalling Data Changes

---

- Could an Edit Check be written to keep your own journal of data changes?
- On exit from each plate:
  - compare all fields with their original values
  - for each value that has been modified journal:  
date,time,user, keys(image,id,visit,plate),  
field number/name, entry value and exit value.
- Why won't this work, even if we check for illegal fields and unresolved QCs?

## Limitations of Plate Exit Edit Checks

---

If the user sets the following preference:  
“ask for confirmation before saving record” the user can abort a record commit. This occurs after any plate exit edit checks have executed.

Remember:

Plate exit edit checks execute every time a user signs off a record by selecting clean, dirty or error on the status field.