
Batch Edit Checks

Jeanine Hammar
Clinical DataFax Systems Inc.

What Are Batch Edit Checks?

- A framework in which all or some study edit checks can be executed outside the Validation Tool
- Use existing edit check functionality
- Capabilities of interactive edit checks are maintained

When Are They Appropriate?

- Execution of cross-plate ECs
- Applying new ECs or re-applying current ECs to existing data
- Applying ECs that do not require user interaction
- Executing user-specific ECs
- Creation of audit trail for EC execution

When Are They Inappropriate?

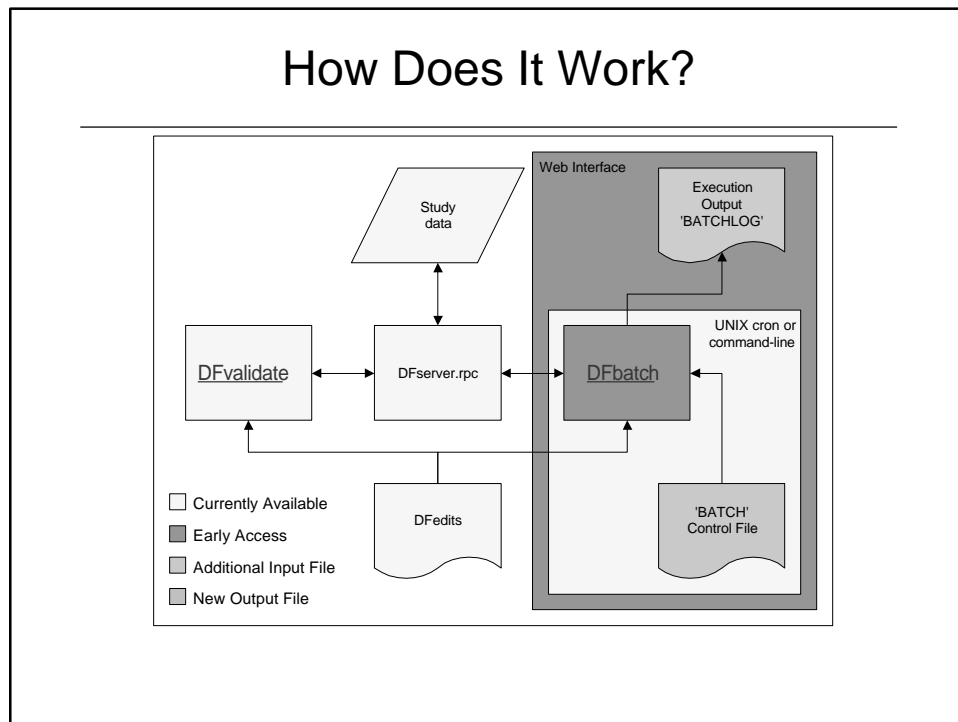
- Execution of lookup table ECs
- Execution of ECs that display any popup dialog windows (e.g. add qc, confirmation)
- Retrieval/validation of level 0 records

How Do They Function?

- Batch edit check framework:
 1. Input specification defining record selection criteria
 2. Program that processes selected records applying the edit checks
 3. Output specification that logs all events of the session
- Execution without user input

What About Existing Edit Checks?

- User can specify by name those edit checks that are to be executed in batch mode → default = execute all
- Use `dfbatch()` function within existing edit check code



Step 1: Implementing Batch Edit Checks

Defining Input Specifications

- Input specifications must be previously defined as batch edit checks do not rely on user input
- Is text- (XML-) based

Input File Specifications

- Name the batch
- Specify actions to occur
- Specify logging
- Select records using criteria similar to “By Data Fields” retrieval
- Specify processing (sort) order

Input File: Example

```
<?xml version="1.0"?>
<!-- This batch selects all primary records at level 2, sorts them in ascending order
      by id, visit, and plate, then executes the edit checks editname1, editname2,
      editname3, promoting records that are changed to level 3. -->
<BATCHLIST>
<ACTION>
  <APPLY detail="changes" level="3" which="all" />
</ACTION>
<BATCH name="batch1">
  <ACTION>
    <LOG detail="changes" which="all"
         file="/studies/254/batch/outputs/batch1.xml" mode="a" />
  </ACTION>
  <CRITERIA sort="+id:+visit:+plate:">
    <LEVEL>2</LEVEL>
    <STATUS>primary</STATUS>
    <EDIT>editname1 editame2 editname3</EDIT>
  </CRITERIA>
</BATCH>
</BATCHLIST>
```

Input File Formatting Rules

- Consists of nested elements having matching open/close tags
- May define more than one batch
- White space is ignored
- Comments are denoted in HTML/XML style

Step 2: Executing Batch Edit Checks

Processing Selected Records with DFbatch

- The DFbatch program:
 - Requires an input file
 - Applies edit checks to records selected from specifications in the input file
 - Batch execution is performed by DFbatch from a command-line or as a cron job

DFbatch Syntax

DFbatch -b "BATCHname" study# batchfile

- BATCHname = name of a BATCH as it appears in a BATCHLIST
- Multiple BATCHnames must be comma/space delimited and enclosed within double quotes
- batchfile = name of the file containing the BATCHLIST

DFbatch: An Example

- From a command-line

```
DFbatch -b "batch1" 254 /studies/254/batch/inputs/batch1.xml
```

- From a cron job

```
0 23 * * * DATAFAX_DIR=/opt/datafax; export DATAFAX_DIR;  
/opt/datafax/bin/DFbatch 254 /opt/studies/254/batch/inputs/batch1.xml
```

What Happens?

- DFbatch reads input file
- Executes each batch in the order that it appears in the BATCHLIST
- For each batch:
 - Retrieves records for each batch according to that batch's record selection CRITERIA
 - Executes specified edit checks
 - Applies first local then global ACTION directives (includes output)

Executing Parts of Edit Checks

- Include dfbatch() test in edit check
- Allows user to select which part of an edit check's code is to be executed in batch or interactive mode
- Returns TRUE/FALSE
 - TRUE = executing in batch mode
 - FALSE = executing in interactive mode

dfbatch(): An Example

```
edit BatchTest()
# Generate different messages for different
  execution modes
{
  if( dfbatch() == 0 )
    dferror( "Not executing in batch" );
  if( dfbatch() != 0 )
    dferror( "Executing in batch only" );
}
```

Step 3: Batch Output

Reviewing Output Logs

- A BATCHLOG exists for each BATCH whose execution created the result → identified by BATCH name and file
- A BATCHLOG contains R (record) elements for each record that was processed

Logging Edit Check Execution

- Edit check actions logged include:
 - Messages
 - Data field changes
 - QC notes added
 - Missing page QC notes added
 - Missing page QC notes deleted

Summary Logs

- When (start/stop time) and by whom the execution occurred
- How many:
 - Messages were generated
 - QC notes were added
 - Data records were modified
- Version information

Formatting/Filtering Log Information

- Log information is XML-based
- Easily formatted for a variety of audiences using XSL as a formatting/filtering tool
- Can be viewed with a web browser

Implications

Caveats

- Currently, a user's permissions to define and execute batch edit checks are the same as their Validation Tool permissions
- No roll-back feature – once changes are applied there is no going back
- Don't use batch edit checks to change data values (FDA)

Recommended Usage

```
<ACTION>  
  <APPLY which="none"/>  
  <LOG which="all"/>  
</ACTION>
```

- Log all edit check actions but do not apply changes to the database
- After confirmation, APPLY which="all" and re-execute

Data Security Issues

- Not possible to retrieve new, level 0 records
- Can not alter the validation level of any record to level 0 nor the status of any record to new
- Not possible to enter data into the database which can not be entered through the Validation Tool

Data Security Issues (cont'd)

- Illegal values cause record status to become dirty, if it is not already
- Changes to DataFax key fields are not permitted
- Secondary records can not be involved in any QC note operations
- Record locking and user permissions are enforced

Demonstration