Global Edit Checks
Development, Testing, and Documentation
Clare Mengebier
Overview

• Background on Edit Checks & Global Edit Check Library
• Our Process to Update the Edit Check Library
  o Review & feedback collection process
• Edit Check Testing & Documentation Process
• Lessons Learned & Next Steps
• Questions
Introduction to Edit Checks

• DataFax/DfDiscover provides a general-purpose programming environment where edit check language can be used to:
  o Perform **logic checks** on data fields
  o Generate **Quality Control (QC) notes**
  o Calculate and **insert values** into fields

• For example, edit checks can:
  o Generate a query message pop-up if the data entered in a field is outside of the legal range
  o Auto-fill the CRF version date based on the version number entered into the previous field
  o Create a new record at the same visit based on the response on a saved CRF
Introduction to Edit Checks

• Benefits of Edit Checks
  o Improve **data quality** within an EDC system
  o Provide **real-time feedback** for site staff as they enter data
  o Facilitate **early resolution** of data discrepancies
  o **Automate** the querying and data review process
  o Allow monitors and data managers to focus on more complex review & data cleaning tasks
What are Global Edit Checks?

- **Global Edit Checks**: a library of edit checks available for use without modification across studies on the same server.
- Tested separately from study-specific edit checks (stored in DFedits file).
- Examples:
  - `HideIf("Role")`: hides a variable if a user is logged in with the role specified in the edit check parameter.
  - `ReqIf(x,y)`: fires a query message if the current variable is blank and the comparison variable `x` contains the value `y`.
date format "dd/mm/yyyy"

# Use LTEST LUT on LB example 2 (plate 10) to enforce test names from the codelist

edit LUT_LTEST() {
    string s;

    # search for exact match without showing LUT
    s = dflookup("LTEST", $T, ",", "-1");

    # if exact match found, keep the value entered
    if ( s == $T ) $T = s;
Benefits of Using Global Edit Checks

• Pre-tested/validated code
• Require less technical knowledge about coding and programming
• Widely applicable across studies
• Saves hours of time in study setup and testing
  ○ In a typical study, approx. 75% of all edit checks can be made global
DF/Net Project to Update Global Edit Checks

Why?

- Limited documentation or testing on current global edit checks
- Some edit checks did not consistently work as expected
- Query language not appropriate or too study-specific
- Edit checks not adaptable to EDC
- Annotation inconsistent or confusing
- DM team re-writing the same types of custom edit checks for every new study
- Staff applying edit checks in DFsetup don’t have programming or coding backgrounds
Plan

Step 1: Review & Feedback Collection
Step 2: Coding Updates
Step 3: Testing & Documentation
Step 4: Development of User Resources
Team

- Data Coordinator Manager (Clare) in charge of overall project management, including documentation and testing
- Support Technician (Tadas) in charge of coding outline & changes
- Summer interns helped implement updates
- Data Coordinators helped with formal testing of edit checks
Step 1: Review and Feedback Collection

- Goal: come up with new edit check structure & final list of global edit checks
- Asked for feedback from Data Managers and Data Coordinators
  - What edit checks would you like to see added?
  - What edit checks could be improved?
  - How could global edit checks be more adaptable to your studies?
- Reviewed the custom edit checks in sample group of studies and listed out commonly used checks
- Informally tested all old global edit checks for a "first-pass" validation that code was working as expected
Step 1: Review and Feedback Collection

Improvements to Edit Checks:

• Creation of multiple variations of each edit check
  o Cross-plate variation
  o Batch-only variation
  o Customizable variation

• New global edit checks (e.g. ReqIfTwo)

• New functions that allow users in DFsetup to apply the same settings to all global edit checks at once
  o Example: whether or not edit checks fire when users tab backward

• “Packages” of similar edit checks (e.g. MedDRA edit checks) so users can add only the edit check files they need
Step 1: Review and Feedback Collection

Other Improvements:

1. Naming convention for all global edit checks (g_EditCheckName)
2. Update query language more generalized (e.g. “Please review.” instead of “Please initial, date and re-fax”)
3. Detailed annotation added for better user understanding and easier customization (for non-programmers)
4. Edit Checks testing organized and documented
Step 1: Review and Feedback Collection

Example of Multiple Variations Built into Global Edit Checks

- NotReqIf
- NotReqIf Cross Plate
- NotReqIf Batch-Only
- NotReqIf Cross Plate + Batch-Only
- NotReqIf w/ Customizable Code

```c
// Basic Version
for (int i = 0; i < 5; i++)
    if (field i has a value of 5 and current field is completed, fire a query)

NotReqIf (1-1,0) signifies if the previous field has a value of 0 and current field is completed, fire a query
NotReqIf(1,0) signifies if field i has a value of 0 and current field is completed, fire a query

// Cross Plate Version
for (int i = 0; i < 5; i++)
    if (field i has a value of 5 and current field is completed, fire a query)

NotReqIf(CP)(10,0,0,0,0) signifies if at visit 10,0, plate 0, field 0 has a value of 0 and current field is completed, fire a query
NotReqIf(CP)(1,0,0,0,0) signifies if at visit 1,0, plate 0, field 0 has a value of 0 and current field is completed, fire a query

// Batch Only
for (int i = 0; i < 5; i++)
    if (field i has a value of 5 and current field is completed, fire a query)

NotReqIf(Batch)(1,0,0,0,0) signifies if batch 1, plate 0, field 0 has a value of 0 and current field is completed, fire a query
NotReqIf(Batch)(1,0,0,0,0) signifies if batch 1, plate 0, field 0 has a value of 0 and current field is completed, fire a query

// Cross Plate + Batch Only
for (int i = 0; i < 5; i++)
    if (field i has a value of 5 and current field is completed, fire a query)

NotReqIf(CP+Batch)(1,0,0,0,0) signifies if at visit 1,0, plate 0, field 0 has a value of 0 and current field is completed, fire a query
NotReqIf(CP+Batch)(1,0,0,0,0) signifies if at visit 1,0, plate 0, field 0 has a value of 0 and current field is completed, fire a query

// Customizable Code
for (int i = 0; i < 5; i++)
    if (field i has a value of 5 and current field is completed, fire a query)

NotReqIf(Customizable)(1,0) signifies if the previous field has a value of 0 and current field is completed, fire a query
NotReqIf(Customizable)(1,0) signifies if the previous field has a value of 0 and current field is completed, fire a query
```

Copy this source code into profiles for custom edit checks

- NotReqIf(Customizable)(1,0) signifies if the previous field has a value of 0 and current field is blank, fire a query
- NotReqIf(Customizable)(1,0) signifies if the previous field has a value of 0 and current field is blank, fire a query
- NotReqIf(Customizable)(1,0) signifies if the previous field has a value of 0 and current field is blank, fire a query

```c
if (field i has a value of 0 and current field is completed, fire a query)
```

- If a negative number is used in first parameter, look at a...
Computations

NotReqIf
NotReqIf_cp
NotReqIf_b
NotReqIf_cp_b

Unique Outcome
Unique Outcome
Unique Outcome
Unique Outcome
# The Function

number NotReqIfTrigger(string visit, string plate, number field, string value) {
    number triggervalue = 0;
    NotReqIfType();

    if (@[,visit,plate,field] == value) {  # If specified field equals specified value
        if (!dfblank(@T)) {  # and current field is not blank,
            triggervalue = 1;  # fire query.
        }
    }
    return triggervalue;  # Otherwise, do not fire query.
Step 2: Coding Updates

- Created a test study on which to write and test new global edit checks
  - Created sample set of CRFs that included all possible field types
  - Worked on server where no active studies that utilized global edit checks were running
- Systematic & documented process for making updates
- 2 rounds of peer review prior to formal testing
# Step 2: Coding Updates

## Global Edit Check Project

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AlwaysSkip</td>
<td>Clare</td>
<td>Clare</td>
<td>Clare</td>
<td>Abigail</td>
<td>N/A</td>
<td>Michael</td>
<td>N/A</td>
<td>Clare</td>
<td>Clare</td>
<td>Clare</td>
</tr>
<tr>
<td>AutoFill</td>
<td>Tadas</td>
<td>Clare</td>
<td>Tadas</td>
<td>Abigail</td>
<td>N/A</td>
<td>Michael</td>
<td>N/A</td>
<td>Clare</td>
<td>Clare</td>
<td>Clare</td>
</tr>
<tr>
<td>DateWithin</td>
<td>Tadas</td>
<td>Tadas</td>
<td>Tadas</td>
<td>Abigail</td>
<td>Taura (No_CP_batch)</td>
<td>Chris</td>
<td>Sara</td>
<td>Tadas</td>
<td>Tadas</td>
<td>Clare</td>
</tr>
<tr>
<td>EDShortcut</td>
<td>Tadas</td>
<td>Tadas</td>
<td>Tadas</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>Tadas</td>
<td>N/A</td>
<td>Tadas</td>
</tr>
<tr>
<td>GreaterThan</td>
<td>Michael</td>
<td>Michael</td>
<td>Clare</td>
<td>Michael</td>
<td>Taura</td>
<td>Abigail</td>
<td>Sara</td>
<td>Clare</td>
<td>Clare</td>
<td>Clare</td>
</tr>
<tr>
<td>GreaterThanOrEqual</td>
<td>Michael</td>
<td>Michael</td>
<td>Clare</td>
<td>Michael</td>
<td>Taura</td>
<td>Abigail</td>
<td>Sara</td>
<td>Clare</td>
<td>Clare</td>
<td>Clare</td>
</tr>
<tr>
<td>HideIf(&quot;Role&quot;)</td>
<td>Tadas</td>
<td>Abigail</td>
<td>Clare</td>
<td>Clare</td>
<td>N/A</td>
<td>Chris</td>
<td>N/A</td>
<td>Tadas</td>
<td>Tadas</td>
<td>Clare</td>
</tr>
<tr>
<td>HideIfNot(&quot;Role&quot;)</td>
<td>Clare</td>
<td>Clare</td>
<td>Clare</td>
<td>Clare</td>
<td>N/A</td>
<td>Chris</td>
<td>N/A</td>
<td>Tadas</td>
<td>Tadas</td>
<td>Clare</td>
</tr>
<tr>
<td>LegalRange</td>
<td>Clare</td>
<td>Michael</td>
<td>Clare</td>
<td>Abigail</td>
<td>Taura (No_CP_batch)</td>
<td>Michael</td>
<td>Sara</td>
<td>Clare</td>
<td>Clare</td>
<td>Clare</td>
</tr>
<tr>
<td>LessThan</td>
<td>Michael</td>
<td>Michael</td>
<td>Clare</td>
<td>Michael</td>
<td>Taura</td>
<td>Abigail</td>
<td>Sara</td>
<td>Clare</td>
<td>Clare</td>
<td>Clare</td>
</tr>
</tbody>
</table>
Step 3: Testing and Documentation

• Testing edit check specifications
  o Conditions under which the edit check is expected to execute and not expected to execute

• Testing all variations of each edit check (e.g. batch-only version, customizable version, etc.)

• Written by Clare, interns, and Data Coordinator
  o Trained in edit check testing
<table>
<thead>
<tr>
<th>Edit Check Name</th>
<th>Purpose</th>
<th>Specifications</th>
<th>Testing Plan</th>
<th>Testing Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReqS</td>
<td>Fires a query if the parameter field has a specified value and the current field is blank.</td>
<td><strong>Query dialog box with appropriate query language should appear if:</strong> 1. Parameter field has specified value and current field is blank (tested on choice, string, number, date, and check fields).</td>
<td>Log into &quot;global edits 1&quot; study in IDatefax 2016.0.0, <strong>Query dialog box with appropriate query language should appear if:</strong> 1a. Create Plate 490 for Participant 9999320. Mark 'other, specify' for item 2 and tab through specify field, leaving it blank. 1b. Enter &quot;OTHER&quot; for item 2 specify field and tab through item 4, leaving it blank. 1c. Delete the response for item 2 specify field. Enter &quot;01&quot; for item 3 AE page # and tab through item 4, leaving it blank. 1d. Delete the response for item 3 AE page #. Enter &quot;01/JAN/17&quot; for item 2 Date of death and tab through item 4, leaving it blank. 1x. Create Plate 250 for Participant 9999320. Leave item 6 WBC Not reported blank. Leave Item 6 Severity Grade blank and tab off.</td>
<td>Passed</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Query dialog box should not appear if:</strong> 2. Parameter field has specified value and current field is completed. 3. Parameter field does not have specified value. 4. Parameter field has specified value and current field has a missing value code applied.</td>
<td><strong>Query dialog box should not appear if:</strong> 2a. On Plate 490 for Participant 999920, mark Item 2 'other, specify'. Enter a response for specify field and tab off. 2b. Enter &quot;OTHER&quot; for item 2 specify. Enter a response for item 4 and tab off. 2c. Enter &quot;01&quot; for item 3 AE page #. Enter a response for item 4 and tab off. 2d. Enter &quot;01/JAN/17&quot; for item 2 Date of death. Enter a response for item 4 and tab off. 2x. On Plate 250 for Participant 9999320, leave item 6 WBC Not reported blank. Enter a response for Item 6 Severity Grade and tab off. 3a. On Plate 490 for Participant 999920, mark 'death' for Item 2 and tab through specify field, leaving it blank. 3b. Enter &quot;TEST&quot; for item 2 specify field and tab through Item 4, leaving it blank. 3c. Delete the response for item 2 specify field. Enter &quot;04&quot; for item 3 AE page #: and tab through item 4, leaving it blank. 3d. Delete the response for item 3 AE page #:. Enter &quot;01/FEB/17&quot; for item 2 Date of death and tab through item 4, leaving it blank. 3e. On Plate 250 for Participant 9999320, mark Item 6 WBC Not reported. Leave Item 6 Severity Grade blank and tab off. 4a. On Plate 490 for Participant 999920, mark 'other, specify' for item 2 and apply a missing value code to specify field and tab off. 4b. Enter &quot;OTHER&quot; for item 2 specify field and apply a missing value code to item 4 and tab off. 4c. Delete the response for item 2 specify field. Enter &quot;01&quot; for item 3 AE page #: and apply a missing value code to item 4 and tab off.</td>
<td>Passed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Step 3: Testing and Documentation

• Edit check testing plans updated and retested anytime code changes
  o With each iteration, trying to make the testing more detailed (e.g. testing the behavior when a missing value code is added, testing all combinations of field types)

• Updates to SOPs to clarify process for:
  o Implementing global edit checks on new servers
  o Formal testing requirements for global edit checks for new DataFax/DFdiscover products and software upgrades

• Formal testing plans stored for Data Manager access
Step 4: Development of User Resources

• **Goal**: Anyone using the global edit checks won’t have to go into the edit check files and try to understand code (i.e. could apply edit checks directly to the fields in setup)

• **Resources**:
  1. Global Edit Check User Guide
  2. Edit check code stored externally (outside of server)
  3. Survey for internal staff to submit feedback, report issues, and track the status of this feedback
II. Global Edit Checks Structure

Variations
Each Global Edit Check may have up to 5 variations which function differently in iDataFax but all share the same general purpose.

1. Basic Edit Check: Runs edit check in both the live study and in batch reports and is limited to field references on the same plate on which it is applied.
2. Cross-Plate Variation: Runs edit check in both the live study and in batch reports and allows references to fields on visits and/or plates other than the plate on which it is applied.
3. Batch-Only Variation: Only runs edit check in batch reports and is limited to field references on the same plate on which it is applied.
4. Batch-Cross Plate Variation: Only runs edit check in batch reports and allows references to fields on visits and/or plates other than the plate on which it is applied.
5. Customizable Variation: Performs the same as basic edit check with the code packaged to allow for copy and pasting into EDits file for customization of the code. (NOTE: Customizable variations need to be tested in study-specific edit checking across the code is modified in EDits for that study)

Supporting Files and Functions
The following files are required to be imported into the database as a local or global file for global edit function in iDataFax and affect how queries and edit checks function within each study.

EDCShortcut
EDCShortcut is the function that allows the study to recognize and apply Queries onto fields in a consistent manner, depending on whether the study is Paper or EDC/Hybrid. Studies will directly apply a query, whilst all EDC/Hybrid studies will prompt the user an error, and provide the choice to either apply the Query or fix the mistake.

TheQuery
TheQuery is the function that evaluates the settings and variables defined via the trigg global edit check and EDCShortcut to generate the proper query type and apply it to the For EDC studies, a warning message will fire with the action to ‘Fix Now’ (return to file ‘Apply QC’ open the query dialog box for editing and application). For paper studies, dialog box automatically opens and no warning message fires.

For studies with the multiple QC function turned on iDataSetup, if a pre-existing query...
g_AlwaysSkip

#|AlwaysSkip|Skips field during data entry and in batch.|
#Apply to String, Number, Date, Choice, and Check Fields
#Updated: 7/18/16
#Author: Clare

#NOTE: Always apply edit check on Field Enter

#Basic (Can also be used as the Source Code for Custom EC)
edit g_AlwaysSkip() {
    if (dfdirection() > 0) dfmoveto(@T+1);  #Skips field when tabbing forward
    if (dfdirection() < 0) dfmoveto(@T-1);  #Skips field when tabbing backward
    if (dfbatch()) dfmoveto(@T+1);  #Skips field if in batch
}

g_AutoFill

#|AutoFill|Auto fills field with specified data.|
#Apply to String, Number, Date, Choice, and Check Fields
#Updated: 7/20/16
#Author: Tadas

#Basic Version (Can also be used as the Source Code for Custom EC)
#Ex1: AutoFill("01/JAN/90") auto-fills the current field with 01/JAN/90 upon tabbing into field (IF edit check is placed on Field Entry)
#Ex2: AutoFill("Version 3.0") auto-fills the current field with the text "Version 3.0"
edit g_AutoFill(string fill) {
    dfautoreason(0);
    $T = fill;
}
# Global Edit Check Feedback Form

Please enter your feedback or request for a global edit check below and Clare or Tadas will follow up as soon as possible.

**Global Edit Check Feedback/Issue (include specific example, PTID, if possible)**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name &amp; Date</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Errors popping up in studies where 'visit' is the Field Name for the visit code field</td>
<td>Pete, 05/30/17</td>
<td>Changing 'visit' variable within global edit check code to 'visittal' to avoid variable name mixing</td>
<td>TL, 6/8/17</td>
</tr>
<tr>
<td>In testing the application of the new global edit checks in the Stay study, I'm noticing that when a query fires, I can't back-tab (shift-tab) to move backward <em>past</em> the variable in question. We want to be able to back-tab to a previous variable to correct its value so the subsequent edit check doesn't fire. In the old hybrid checks we always included this:</td>
<td>Pete, 06/02/17</td>
<td>Direction functionality added (Each study has a “study setting” at the top of their edit check file that defines whether reverse tabbing avoids queries or not called “backTab”. 0 = Feature offline, 1 = Feature online. So if people want to allow backTabbing, just change the 0 to a 1</td>
<td>TL, 6/8/17</td>
</tr>
<tr>
<td>if (dfDirection()&lt;0) return;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>which allows the user to tab backward without queries popping.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Lessons Learned

1. There are trade-offs between time & resources available and testing & documentation requirements
   - Can’t test every possible situation under which edit check is applied

2. Provide documented and transparent mode of communication for feedback and updates of global edit checks
Next Steps and Future Improvements

- Continue retesting global edit checks with more detailed tests
- Work with software development team to investigate ways to automate testing process
- Development of more HTML-friendly query messages
- Further implementation in new studies and continuing discussion on study-specific needs and opportunities to make global edit checks more flexible/adaptable
Thanks for listening!

Questions?