Paper 10841-2016

# Data Review Listings on Auto-Pilot: Using SAS® and Windows Server to Automate Reports and Flag Incremental Data Records

Victor A. Lopez, Samumed, LLC; Heli Ghandehari, Samumed, LLC;
Bill Knowlton, Samumed, LLC; Christopher J. Swearingen, Samumed, LLC

## ABSTRACT

During the course of a clinical trial study, large numbers of new and modified data records are received on an ongoing basis. Providing end users with an approach to continuously review and monitor study data, while enabling them to focus reviews on new or modified (incremental) data records, allows for greater efficiency in identifying potential data issues. In addition, supplying data reviewers with a familiar machine-readable output format (for example, Microsoft Excel) allows for greater flexibility in filtering, highlighting, and retention of data reviewers' comments.

In this paper, we outline an approach using SAS® in a Windows server environment and a shared folder structure to automatically refresh data review listings. Upon each execution, the listings are compared against previously reviewed data to flag new and modified records, as well as carry forward any data reviewers' comments made during the previous review. In addition, we highlight the use and capabilities of the SAS® ExcelXP tagset, which enables greater control over data formatting, including management of Microsoft Excel's sometimes undesired automatic formatting. Overall, this approach provides a significantly improved end-user experience above and beyond the more traditional approach of performing cumulative or incremental data reviews using PDF listings.

## INTRODUCTION

Clinical trial data review, performed by subject matter experts (for example, medical monitors, data managers, etc.), is an important part of monitoring subject safety and data integrity. Data integrity is critical in statistical analysis and interpretations. Ongoing and early review of data helps identify existing errors however, more importantly, can help prevent future errors by allowing early identification of systemic causes. Data review can be challenging as a large number of new and modified data records are received on an ongoing basis, which raises a need for more efficient approaches. Traditional approaches have focused on cumulative or incremental data reviews using PDF listings, typically printed and manually annotated for follow-up actions. Enabling end users control over filtering, highlighting, sorting, and note taking at the record level significantly improves end user experience and overall efficiency in conducting clinical trials.

In this paper, we outline a data listing programming approach using SAS® in a Windows server environment with the following key features:

- Automatic data review listing's re-execution
- Flagging new or modified (incremental) data records
- Outputting in machine readable spreadsheet format using SAS® ExcelXP tagset features
- Recording and retention of data reviewers' comments from one review to another

## OVERVIEW OF DATA LISTING APPROACH

Figure 1 is a high level diagram of the Master Program. The Figure provides a general overview of the standard Folder Structure and Naming Conventions for each file. It illustrates the use of the three Supporting Macros that are used repetitively for each listing and describes the function of individual Listing Program(s). Details on each component are described in the subsequent sections.

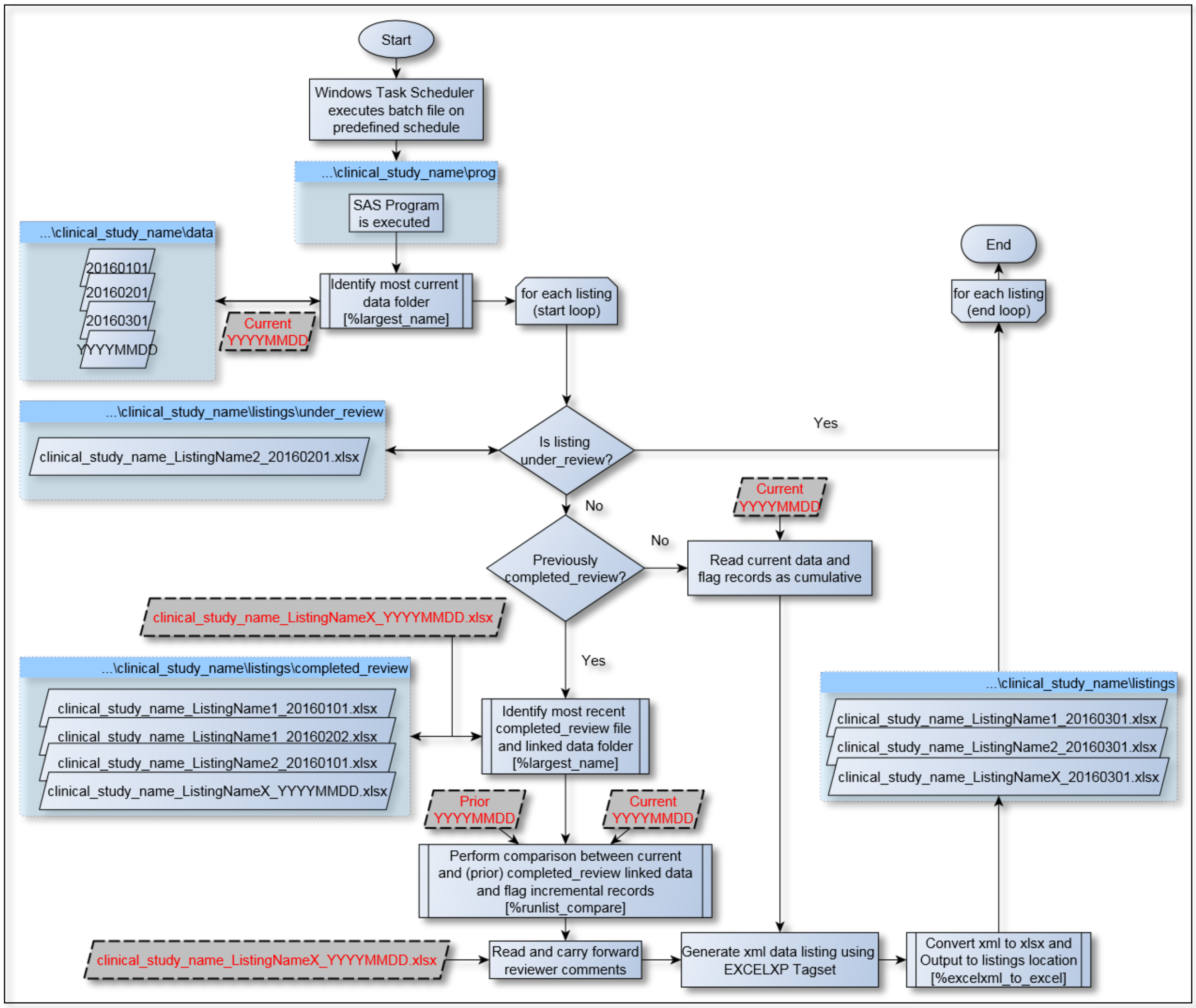

Start
Windows Task Scheduler executes batch file on predefined schedule
...\clinical_study_name\prog
SAS Program is executed
...\clinical_study_name\data
20160101
20160201
20160301
YYYYMMDD
Current YYYYMMDD
Identify most current data folder [%largest_name]
for each listing (start loop)
End
for each listing (end loop)
...\clinical_study_name\listings\under_review
clinical_study_name_ListingName2_20160201.xlsx
Is listing under_review?
Yes
No
Current YYYYMMDD
Previously completed_review?
No
Read current data and flag records as cumulative
clinical_study_name_ListingNameX_YYYYMMDD.xlsx
Yes
...\clinical_study_name\listings\completed_review
clinical_study_name_ListingName1_20160101.xlsx
clinical_study_name_ListingName1_20160202.xlsx
clinical_study_name_ListingName2_20160101.xlsx
clinical_study_name_ListingNameX_YYYYMMDD.xlsx
Identify most recent completed_review file and linked data folder [%largest_name]
...\clinical_study_name\listings
clinical_study_name_ListingName1_20160301.xlsx
clinical_study_name_ListingName2_20160301.xlsx
clinical_study_name_ListingNameX_20160301.xlsx
Prior YYYYMMDD
Current YYYYMMDD
Perform comparison between current and (prior) completed_review linked data and flag incremental records [%runlist_compare]
clinical_study_name_ListingNameX_YYYYMMDD.xlsx
Read and carry forward reviewer comments
Generate xml data listing using EXCELXP Tagset
Convert xml to xlsx and Output to listings location [%excelxml_to_excel]

**Figure 1. High Level Diagram of Folder Structure and Program Function**

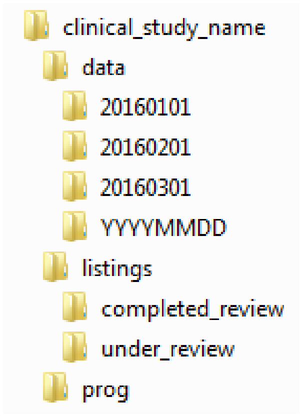
## FOLDER STRUCTURE AND NAMING CONVENTIONS

In order to enable automation, a standard folder structure must be established. The folder structure used in this paper is simplified and illustrative; it can be easily adapted into existing data repository structures and business needs.

The folder structure, illustrated in Figure 2, must consist of:

- **data** folder containing dated subfolders with all cumulative study data (YYYYMMDD, where YYYY indicates 4 digit year, MM indicates 2 digit month, and DD indicates 2 digit day; if data is received more than once per day, time may be appended). When new data is received it should be placed into this location within a new dated folder.
- **prog** is used to contain SAS® program(s) and Windows Batch files.
- **listings** folder will be used to contain the output files, see Figure 3. End users must have access to this location. Within the listings folder, two subfolders are required:

- **under_review** will contain files the end user is currently reviewing. End user is expected to move ready for review files contained in the **listings** folder into the **under_review** folder.
- **completed_review** will contain files for which the end user has competed their review. End user is expected to move reviewed files from **under_review** to the **completed_review** folder.


clinical_study_name
data
20160101
20160201
20160301
YYYYMMDD
listings
completed_review
under_review
prog

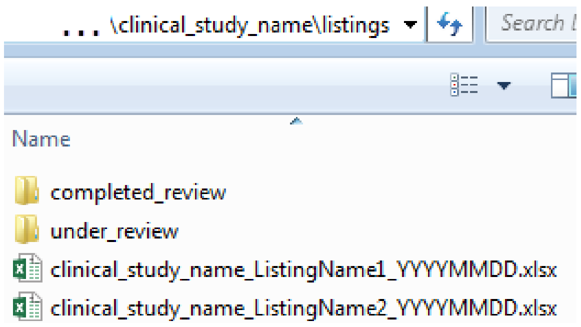

. . . \clinical_study_name\listings
Search
Name
completed_review
under_review
clinical_study_name_ListingName1_YYYYMMDD.xlsx
clinical_study_name_ListingName2_YYYYMMDD.xlsx

**Figure 2. Example Folder Structure**

**Figure 3. Example Contents of listing Folder**

File naming conventions for all output files should also be standard (e.g. <Study Identifier>_<Listing Identifier>_<Data Input Folder>.xlsx). It is required that the last portion of the file name contain the data input folder name (i.e. YYYYMMDD).

## SUPPORTING MACROS

There are three repeating tasks performed throughout the Master Program that should be setup as macros. They are described as follows:

### %LARGEST_NAME

The %largest_name macro identifies the 'largest' (based on ASCII-code order) name of the items (folders or files) in the user specified folder. Should be set up with two parameters:

PATH - Folder location in which to identify 'largest' name of items

TXTMATCH – string pattern of items we wish to identify

```
%macro largest_name(path=, txtmatch=) ;
  %*** initialize and set macro variable to null ;
  %global largest_name ;
  %let largest_name  = ;

  %*** create dataset with the names of the folder items ;
  data largest_name ;
    rc = filename('path',"&path.") ;
    did = dopen('path');
    num_items = dnum(did) ;
    if (num_items gt 0) then
      do i = 1 to num_items ;
        filename=dread(did,i) ;
        output ;
        rc=fclose(did) ;
      end ;
    rc=dclose(did) ;
  run;
  %*** Compute largest based on ASCII-code order;
  proc sql noprint ;
    select strip(scan(max(strip(filename)),1,'.')) as filename
      into :largest_name
    from largest_name
    where filename like "&txtmatch." escape '^'
   order by filename desc ;
  quit;
%mend largest_name ;
```

**%EXCELXML_TO_EXCEL**

The ExcelXP Tagset generates XML output which can be opened directly from Microsoft Excel. However, the XML file is significantly larger when compared to the same amount of data stored in the Microsoft Excel Open XML format (XLSX). Additionally, to enable easier opening, commenting, and re-saving the output file, the Excel native format is preferred. Lastly, we wanted to create output files with the ‘Share Workbook’ Excel feature enabled to allow multiple users to simultaneously review and comment.

The %ExcelXML_to_Excel macro is set up with two parameters:

FILE_PATH – Relative or absolute path to location of file

FILE_NAME – File name excluding extension

```
%macro excelxml_to_excel(file_path=, file_name=) ;
  ** Set options;
  option noxwait;
```

Relative paths cannot be used outside of SAS. Hence, regardless of whether FILE_PATH contains a relative or absolute path the following 2 lines will standardize FILE_PATH input into an absolute path that can be passed outside of SAS and properly execute the subsequent PowerShell command.

```
  %*** determine absolute path of file_path ;
  libname _temp_ "&file_path." ;
  %let abs_path = %sysfunc(pathname(_temp_)) ;
```

The PowerShell command is one long command broken into several lines by enclosing within quotes for easier viewing.

```
  %*** convert XML file to native XLSX Format ;
  x powershell
  -command "$xlXMLSpreadsheet = 51; "
           "$Excel = New-Object -Com Excel.Application; "
           "$WorkBook = $Excel.Workbooks.Open('&abs_path.\&file_name..xml');"
           "$Excel.Application.DisplayAlerts=$False; "
```

The enablement of the ‘Share Workbook’ feature is the 7th parameter of the Workbook.SaveAs Method (2 turns this feature on, 1 will leave it off). The details of each parameter is available from the Workbook.SaveAs Method documentation.

```
"$WorkBook.SaveAs('&abs_path.\&file_name..xlsx',$xlXMLSpreadsheet,[Type]::Mis
sing,[Type]::Missing,$false,$false,2);"

           "$Excel.Quit()" ;

  %*** delete the XML file ;
  x del "&abs_path.\&file_name..xml" ;
%mend excelxml_to_excel ;
```

**%RUNLIST_COMPARE**

The %runlist_compare macro executes the listing data preparation macro against the current data. If applicable, it runs the data preparation macro a second time against the previously reviewed data and performs the comparison between the two data sources in order to flag incremental data records. The macro is set up with one parameter:

LISTNO – Listing number, 3 digit numeric value assigned to each unique listing, e.g. 001

```
%macro runlist_compare(listno);
  %*** Run listing macro for current data ;
  %l&listno.(run=cur);

  %*** If listing has not been previously reviewed - set change flag to NA ;
  %if &&l&listno = NA %then %do;
    data l&listno._final;
      attrib CHGFL label='New or Changed?' format=$2.;
      set l&listno._cur;
      chgfl = 'NA';
    run;
    footnote;
  %end;

  %*** If listing has been previously reviewed, use the previous review date;
  %*** and run listing macro against older data and compare ;
  %else %do;
    %l&listno.(run=lst); /* Re-Execution of listing macro on older data */

    %*** 'Computing' maximum length of matching variables in the current and;
    %*** prior datasets to load into program data vector in next step;
    proc sql;
      create table lengths (where=(0)) as
        select *
        from l&listno._cur
        outer union corr
      select *
      from l&listno._lst;
    quit;

    %*** Performing compare and setting New or Changed flag ;
    data l&listno._final;
    merge lengths l&listno._cur (in=a) l&listno._lst (in=b);
      by _all_;
      /* Set change flag for new/updated records */
      if a = 1 and b = 0 then CHGFL = 'Y';
      else CHGFL = 'N';

      if b = 1 and a = 0 then delete;
      format chgfl $8.;
      label chgfl='New or Changed?';
    run;

    %let fulldt = %substr(&&l&listno.,%index(&&l&listno.,201),8);

      %*** If listing has been previously reviewed, use the previous review
      date;
    footnote "Last execution date of listing: %substr(&fulldt,1,4)-
              %substr(&fulldt,5,2)-%substr(&fulldt,7,2)";
  %end;
%mend;
```

## LISTING PROGRAM(S)

The following is an example listing named ***ListingName1***, which has been assigned unique listing number ***001***. For each listing a macro is setup in identical fashion. Highlighted or **bolded** text represents code that is unique for each listing, all other code would remain identical across all listings. For example, in a second listing, **002**, would replace all green highlighted locations and the descriptive name would replace the yellow highlight (e.g. ***Medical_History***):

```
%macro ListingName1;
  %*** Determine if listing is currently under review ;
  %largest_name(path=../listings/under_review
               ,txtmatch=clinical_study_name_ListingName1_2%)
  %let ListingName1_ur = %sysfunc(ifc(&largest_name. = ,N ,Y)) ;

  %*** Create listing if applicable ;
  %if &ListingName1_ur. = N %then %do ;
    %*** identify most recent completed_review file and linked data folder.;
    %largest_name(path=../listings/completed_review
                 ,txtmatch=clinical_study_name_ListingName1_2%) ;
    %let ListingName1_lst    = &largest_name. ;
    %let ListingName1_lst_dt = %sysfunc(tranwrd(&largest_name.
                               ,clinical_study_name_ListingName1_,)) ;
    %let L001                = %sysfunc(ifc(&ListingName1_lst_dt. = , NA
                               ,&ListingName1_lst_dt.));
```

For each listing macro there is a sub macro that contains the data preparation code, as follows:

```
    %macro L001(run);
      /* Read in required dataset from last execution date */
      %if &&l&listno ^= NA and &run = lst %then %do;
        libname lst "../data/&L001." access=readonly;
      %end;
```

Regardless of the number of intermediate steps, the final dataset must be of the form LXXX_&run in order for the %runlist_compare macro to read and compare files. In this section, the programmer prepares the input data by merging applicable files, creating new variables, and ensuring a null Reviewer_Notes variable is created. The &***run*** macro variable is used by %runlist_compare to execute against current and, if applicable, against prior execution. Example code for data preparation using an ***ae*** input dataset:

```
      proc sql;
        create table L001_&run as
        select  usubjid
               ,aespid
               ,aeterm
               ,aedecod
               ,aebodsys
               ,aesev
               ,aestdtc
               ,aeendtc
               ,'' as Reviewer_Notes label='Reviewer Notes' length=5000
        from &run..ae ;
      quit;

      proc sort data = L001_&run;
        by _all_;
      run;
    %mend;
```

For each listing macro there is a sub macro that contains the data presentation code and the reviewer notes carry forward feature, as follows:

```
    %macro L001_p;
      %*** - merge end-user notes from previous completed review;
      %*** - if applicable ;
        %if &L001. ~= NA %then %do ;
          proc import dbms=xlsx replace
            datafile="../listings/completed_review/&ListingName1_lst..xlsx"
            out=L001_notes ;
            getnames=yes ;
          run ;

          proc sql ;
            create table L001_final_notes as
            select
               a.*
              ,b.Reviewer_Notes
            from
              L001_final (drop=Reviewer_Notes) as a left join
              L001_notes as b
              on
```

The following variables used to join the previous reviewer notes are unique to this example listing. The programmer will need to identify the variable or set of variables that define uniqueness for each record and is not expected to change as new data is received. Most databases and electronic data capture systems have unique record identifiers. If using standardized data such as Clinical Data Interchange Standards Consortium (CDISC) Study Data Tabulation Model (SDTM) datasets, there is always a set of keys or USUBJID + --REFID or --SPID that can be used depending on study implementation. Additionally, the sort order will vary depending on end user requirements.

```
                b.Subject_ID  = a.usubjid and
                b.Record_ID   = a.aespid
            order by
                a.usubjid, a.aedecod;
          quit ;

        %end ;
        %else %do ;
          proc sort data = L001_final out = L001_final_notes;
            by usubjid aedecod;
          run;

        %end ;

        proc report data=L001_final_notes nowindows split='|';
```

List of columns is unique for each listing however, chgfl and reviewer_notes must remain. Define statements along with ODS "tagattr" attributes should be used, as needed, in order to have full control over Microsoft Excels auto formatting. For example, in order to ensure Microsoft Excel does not convert the ISO 8601 (YYYY-MM-DD) date formats into numeric date fields in the AESTDTC and AEENDTC variables, the define statement uses the ODS "tagattr" text attribute. Please see reference for DelGobbo's ExcelXP Tagset Paper Index which contains references to solutions of all things ExcelXP related and details on the TAGSETS.EXCELXP options.

```
          columns usubjid aespid aeterm aedecod aebodsys aesev aestdtc
                  aeendtc chgfl Reviewer_Notes;

            define usubjid  / 'Subject ID'   style={tagattr='format:text'};
            define aespid   / 'Record ID'    style={tagattr='format:text'};
            define aestdtc  /                style={tagattr='format:text'};
            define aeendtc  /                style={tagattr='format:text'};
        run;
    %mend;

    ** Execute Run Listing macro on current listing;
    %runlist_compare(listno=001);

    ** Prepare output;
    ods listing close;
    ods tagsets.excelxp
      style=analysis
      file="../listings/clinical_study_name_ListingName1_&cur_data..xml" ;

    ods tagsets.excelxp options(sheet_name = 'Adverse Events'
                     autofilter          = 'yes'
                     frozen_headers      = 'yes'
                     embedded_footnotes  = 'yes'
                     absolute_column_width = '12,12,15,12,18,12,12,12,8,25');

    ** Execute listing specific printing macro;
    %L001_p

    ods _all_ close;

    %*** - convert xml to xlsx ;
    %excelxml_to_excel(file_path=../listings
                     ,file_name=clinical_study_name_ListingName1_&cur_data.);
  %end;

%mend ListingName1;

%ListingName1;
```

## MASTER PROGRAM

The following presents the general structure of the 'Master Program' that brings all code together into a single program.

```
%*** Master Listing Program: clinical_study_name_listings.sas ;
%*** (1) Load Supporting Macros ;
  %macro largest_name;
    ...
  %mend;

  %macro excelxml_to_excel;
```

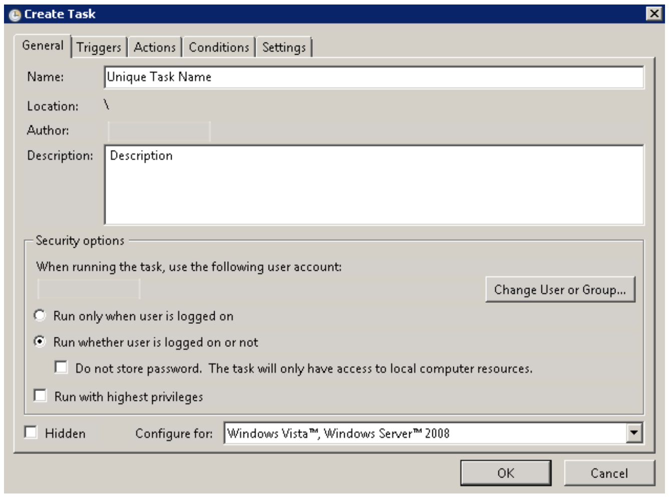
```
    ...
  %mend;

  %macro runlist_compare;
  ...
  %mend;

%*** (2) Identify and load most current data folder;
  %largest_name(path=../data, txtmatch=20%)
  %let cur_data = &largest_name.;

  libname cur "../data/&cur_data.";

%*** (3) Setup and execute each data listing;
  %macro ListingName1;
    ...
  %mend;
  %ListingName1;

  %macro ListingName2;
    ...
  %mend;
  %ListingName2;

  ...

  %macro ListingNameX;
        ...
  %mend;
  %ListingNameX;
```

## BATCH FILE AND AUTOMATIC SCHEDULER

A Windows batch file is a text file with a .bat extension. The following is the contents of an example batch file: clinical_study_name_listings.bat:

```
rem   The rem command is used to enter remarks. Absolute paths must
rem   be used. cd is Change Directory command. del is the delete command

rem   Prior to execution of program, the listings which have not been
rem   moved to under_review or completed_review are removed
cd "C:\[<insert absolute path>]\clinical_study_name\listings"

rem   The following command deletes files with xlsx suffix, quietly (the
rem   prompt to confirm deletion of each file is suppressed)
del *.xlsx /q

rem   Change directory to the program location
cd "C:\[<insert absolute path>]\clinical_study_name\prog"

rem   Execute SAS program
rem   Example location of SAS Executable file is shown below
rem   Example sas program referenced in Master Program Setup Section
"C:\Program Files\SASHome\SASFoundation\9.4\sas.exe" -rsasuser -
nosyntaxcheck -sysin clinical_study_name_listings.sas
```

Once the windows batch file is setup the task can be scheduled in Windows Task Scheduler. Task Scheduler will automatically run the SAS program at the desired intervals. Display 1 shows the recommended 'General' tab options that should be selected. The frequency of re-execution is set in the Triggers tab as shown in Display 2. In the Actions tab the "Start a program" and the windows batch file should be selected in the "Program/script" section as shown in Display 3.


Create Task

General | Triggers | Actions | Conditions | Settings

Name: Unique Task Name

Location: \

Author:

Description: Description

Security options

When running the task, use the following user account:

Change User or Group...

Run only when user is logged on

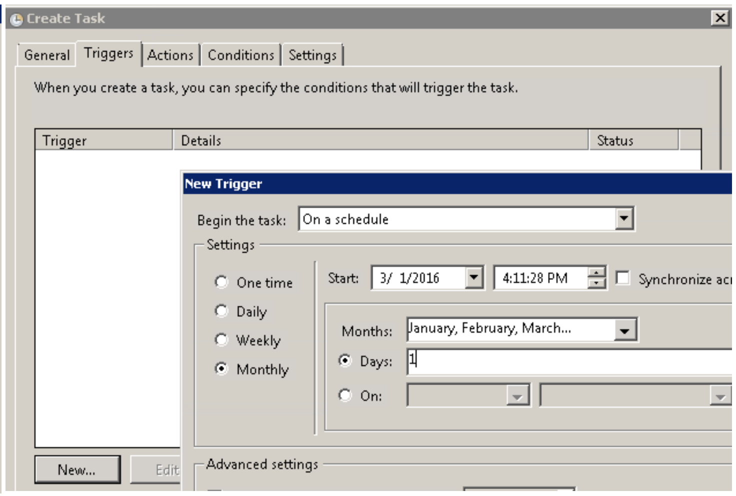
[✓ Run whether user is logged on or not]

Do not store password. The task will only have access to local computer resources.

Run with highest privileges

Hidden  Configure for: Windows Vista™, Windows Server™ 2008

OK  Cancel


Create Task

General | Triggers | Actions | Conditions | Settings

When you create a task, you can specify the conditions that will trigger the task.

| Trigger | Details | Status |
|---|---|---|

New Trigger

Begin the task: On a schedule

Settings

One time / Daily / Weekly / [✓ Monthly]

Start: 3/ 1/2016  4:11:28 PM  Synchronize acr

Months: January, February, March...

[✓ Days: 1]

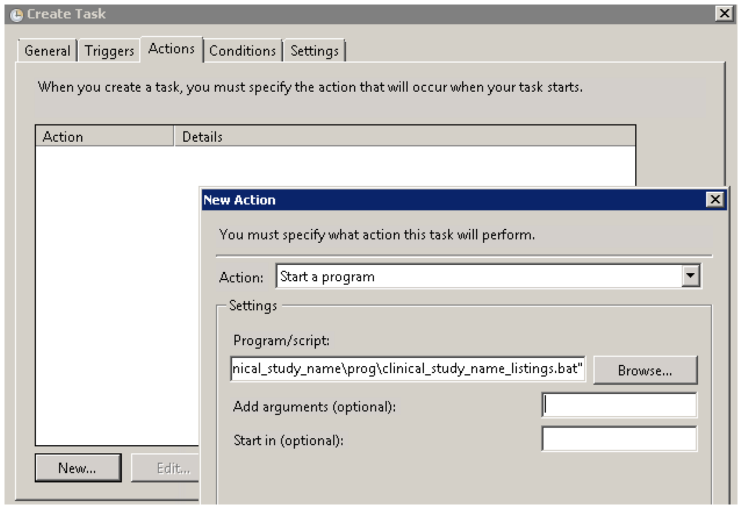
On:

Advanced settings

New...  Edit

Display 1. General tab in Windows Task Scheduler

Display 2. Triggers tab in Windows Task Scheduler


Create Task

General | Triggers | Actions | Conditions | Settings

When you create a task, you must specify the action that will occur when your task starts.

| Action | Details |
|---|---|

New Action

You must specify what action this task will perform.

Action: Start a program

Settings

Program/script:

nical_study_name\prog\clinical_study_name_listings.bat"  Browse...

Add arguments (optional):

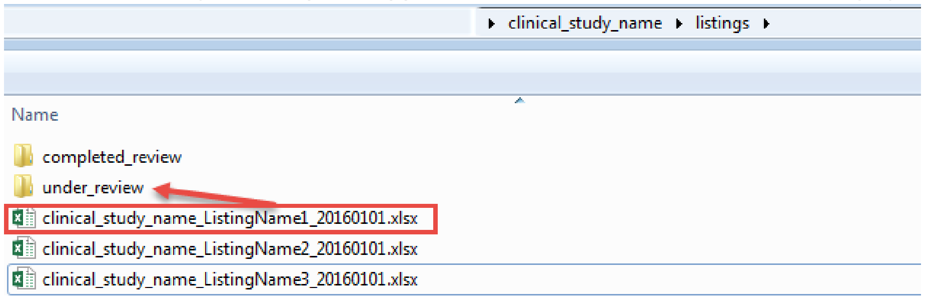
Start in (optional):

New...  Edit...

Display 3. Actions tab in Windows Task Scheduler

## END USER EXPERIENCE

All end users should have read/write access to the **listings** folder. When the end users access the directory they will see all refreshed files with the data date. Once the first person is ready to initiate review, then drag and drop the applicable file into "under_review" folder (Figure 4).


▸ clinical_study_name ▸ listings ▸

Name

completed_review

under_review

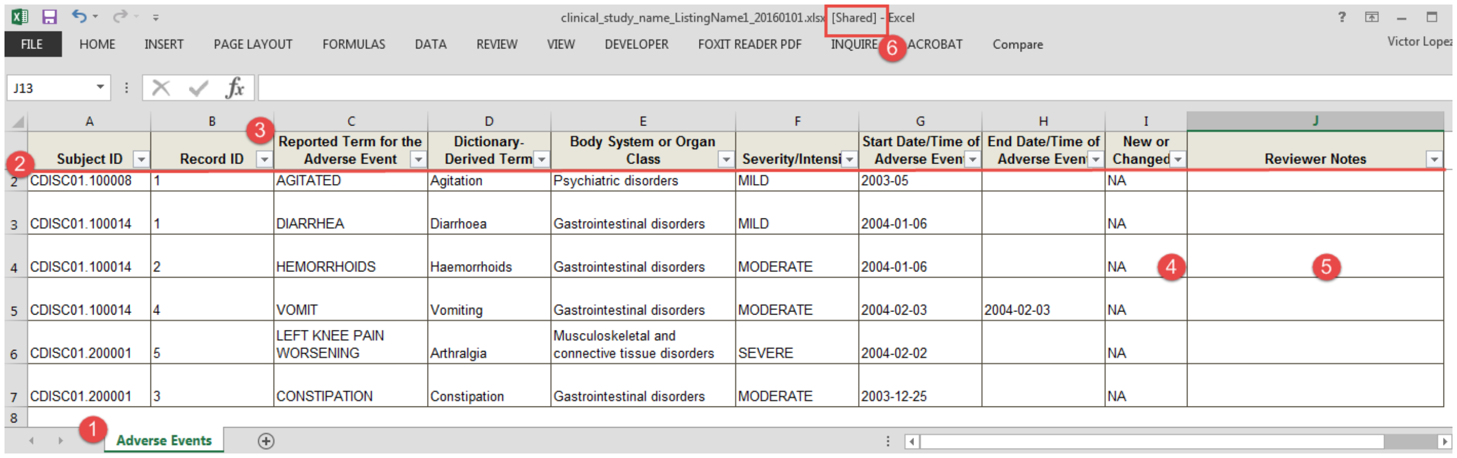
clinical_study_name_ListingName1_20160101.xlsx

clinical_study_name_ListingName2_20160101.xlsx

clinical_study_name_ListingName3_20160101.xlsx

Figure 4. Listings ready for review

The initial execution reveals the spreadsheet features, namely, (1) worksheet name set to “Adverse Event’ (2) frozen top row (3) auto filter is enabled (4) ‘New or Changed’ column is set to NA as this is the first execution (5) “Reviewer Notes” is ready to accept notes and (6) workbook is “Shared” enabled and ready to allow simultaneous reviewers (Figure 5).

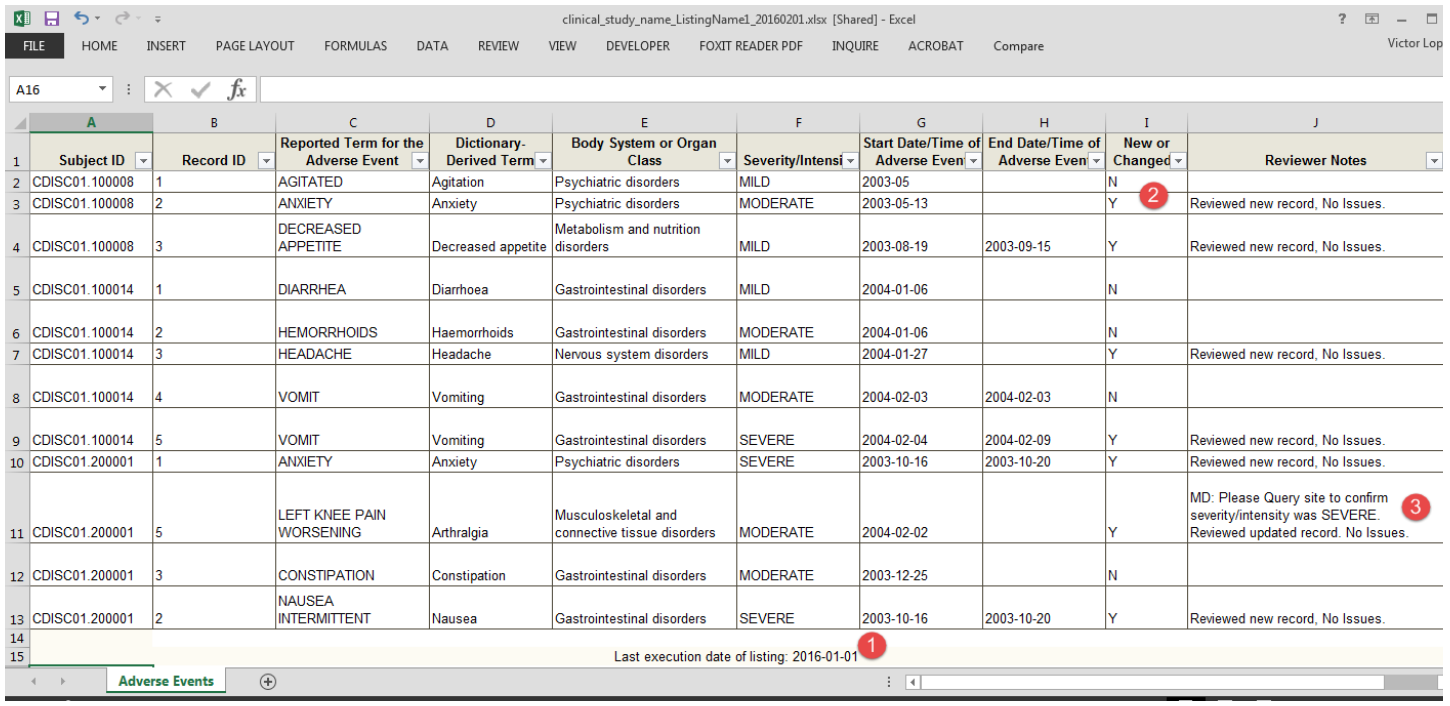

clinical_study_name_ListingName1_20160101.xlsx [Shared] - Excel
FILE HOME INSERT PAGE LAYOUT FORMULAS DATA REVIEW VIEW DEVELOPER FOXIT READER PDF INQUIRE ACROBAT Compare
Victor Lopez
J13
Adverse Events
1
2
3
4
5
6

| Subject ID | Record ID | Reported Term for the Adverse Event | Dictionary-Derived Term | Body System or Organ Class | Severity/Intensi | Start Date/Time of Adverse Even | End Date/Time of Adverse Even | New or Changed | Reviewer Notes |
|---|---|---|---|---|---|---|---|---|---|
| CDISC01.100008 | 1 | AGITATED | Agitation | Psychiatric disorders | MILD | 2003-05 | | NA | |
| CDISC01.100014 | 1 | DIARRHEA | Diarrhoea | Gastrointestinal disorders | MILD | 2004-01-06 | | NA | |
| CDISC01.100014 | 2 | HEMORRHOIDS | Haemorrhoids | Gastrointestinal disorders | MODERATE | 2004-01-06 | | NA | |
| CDISC01.100014 | 4 | VOMIT | Vomiting | Gastrointestinal disorders | MODERATE | 2004-02-03 | 2004-02-03 | NA | |
| CDISC01.200001 | 5 | LEFT KNEE PAIN WORSENING | Arthralgia | Musculoskeletal and connective tissue disorders | SEVERE | 2004-02-02 | | NA | |
| CDISC01.200001 | 3 | CONSTIPATION | Constipation | Gastrointestinal disorders | MODERATE | 2003-12-25 | | NA | |

**Figure 5. Initial execution listing output (data displayed is from the CDISC Pilot Project 1 Study)**

Once review is complete and Reviewer Notes have been entered, the entire file is saved and moved into the completed_review folder.

The subsequent execution reveals additional features, namely, (1) will indicate the last execution date, (2) the ‘New or Changed’ flags now indicate Y/N (end user can filter for incremental records by filtering for only Y) and (3) the Reviewer Notes entered previously have been carried forward (Figure 6). Once this file is moved into the under_review folder, reviewer notes can once again be entered, edited, or deleted and file is placed into the completed_review folder where the process will repeat:

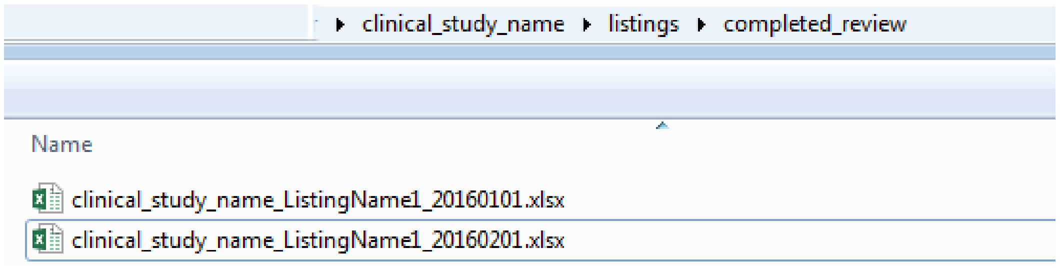

clinical_study_name_ListingName1_20160201.xlsx [Shared] - Excel
FILE HOME INSERT PAGE LAYOUT FORMULAS DATA REVIEW VIEW DEVELOPER FOXIT READER PDF INQUIRE ACROBAT Compare
Victor Lop
A16
Adverse Events
1
2
3

| Subject ID | Record ID | Reported Term for the Adverse Event | Dictionary-Derived Term | Body System or Organ Class | Severity/Intensi | Start Date/Time of Adverse Even | End Date/Time of Adverse Even | New or Changed | Reviewer Notes |
|---|---|---|---|---|---|---|---|---|---|
| CDISC01.100008 | 1 | AGITATED | Agitation | Psychiatric disorders | MILD | 2003-05 | | N | |
| CDISC01.100008 | 2 | ANXIETY | Anxiety | Psychiatric disorders | MODERATE | 2003-05-13 | | Y | Reviewed new record, No Issues. |
| CDISC01.100008 | 3 | DECREASED APPETITE | Decreased appetite | Metabolism and nutrition disorders | MILD | 2003-08-19 | 2003-09-15 | Y | Reviewed new record, No Issues. |
| CDISC01.100014 | 1 | DIARRHEA | Diarrhoea | Gastrointestinal disorders | MILD | 2004-01-06 | | N | |
| CDISC01.100014 | 2 | HEMORRHOIDS | Haemorrhoids | Gastrointestinal disorders | MODERATE | 2004-01-06 | | N | |
| CDISC01.100014 | 3 | HEADACHE | Headache | Nervous system disorders | MILD | 2004-01-27 | | Y | Reviewed new record, No Issues. |
| CDISC01.100014 | 4 | VOMIT | Vomiting | Gastrointestinal disorders | MODERATE | 2004-02-03 | 2004-02-03 | N | |
| CDISC01.100014 | 5 | VOMIT | Vomiting | Gastrointestinal disorders | SEVERE | 2004-02-04 | 2004-02-09 | Y | Reviewed new record, No Issues. |
| CDISC01.200001 | 1 | ANXIETY | Anxiety | Psychiatric disorders | SEVERE | 2003-10-16 | 2003-10-20 | Y | Reviewed new record, No Issues. |
| CDISC01.200001 | 5 | LEFT KNEE PAIN WORSENING | Arthralgia | Musculoskeletal and connective tissue disorders | MODERATE | 2004-02-02 | | Y | MD: Please Query site to confirm severity/intensity was SEVERE. Reviewed updated record. No Issues. |
| CDISC01.200001 | 3 | CONSTIPATION | Constipation | Gastrointestinal disorders | MODERATE | 2003-12-25 | | N | |
| CDISC01.200001 | 2 | NAUSEA INTERMITTENT | Nausea | Gastrointestinal disorders | SEVERE | 2003-10-16 | 2003-10-20 | Y | Reviewed new record, No Issues. |

Last execution date of listing: 2016-01-01

**Figure 6. Subsequent listing output (Data displayed is from the CDISC Pilot Project 1 Study)**

Files stored in the completed_review folder are permanently stored and retrievable if needed (Figure 7):



clinical_study_name_ListingName1_20160101.xlsx
clinical_study_name_ListingName1_20160201.xlsx

**Figure 7. Storage of data listings that have completed review**

## CONCLUSION

In this paper, we have outlined an approach that not only increases end user efficiency but also increases programmer efficiency as it eliminates the need to manually re-execute programs. Since the release of ExcelXP tagset, we can confidently create Excel files with full control over displayed formatting. This provides end users with tabular data in spreadsheets and helps minimize programming requests of new listings. Overall, this approach can be a significant time saver for programming teams.

## REFERENCES

Microsoft Corporation. "Workbook.SaveAs Method (Excel)". Accessed March 3, 2016. https://msdn.microsoft.com/en-us/library/office/ff841185(v=office.15).aspx.

DelGobbo, Vince. "ExcelXP Tagset Paper Index". Accessed March 3, 2016. http://www.sas.com/events/cm/867226/ExcelXPPaperIndex.pdf.

Clinical Data Interchange Standards Consortium. "CDISC SDTM/ADaM Pilot Project". Accessed March 3, 2016. http://www.cdisc.org/cdisc-sdtm/adam-pilot-project.

## ACKNOWLEDGMENTS

The authors would like to thank Jack Ort our colleague at Samumed, LLC for his expertise and assistance with Windows Batch files, PowerShell and establishing the initial groundwork for the use of the ExcelXP tagset for data listings review in our organization.

## CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Victor A. Lopez
Samumed, LLC
9381 Judicial Drive, Suite 160
San Diego, CA 92121
858 926 2973
Victor@samumed.com