Clinical Research Informatics Core

OpenClinica Export Utility

Copyright 2010 Women & Children's Health Research Institute. All rights reserved.

Contents

Disclaimer & License

This document, the application it describes and any associated test data have been developed by the WCHRI Clinical Research Informatics Core for internal purposes. The application and this documentation are made available to the OpenClinica user community as is with no warrantee, implied or otherwise. The application is still under development, is unsupported and un-validated. Use at your own risk!

Licensed under LGPLv2.1. This program is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License version 2.1 as published by the Free Software Foundation.

Introduction

Effective management of a clinical trial requires appropriate reporting and access to information. OpenClinica is a maturing product but provides limited reporting capability. The export capability provided by OpenClinica is adequate for small, simple studies but has its own limitations (the structure of the exported data varies based on the data content, number of records in a group, etc.) CDISC ODM appears to provide an appropriate export mechanism, however ODM is not fully implemented in OpenClinica 2.x, and the ODM implementation in SAS is also problematic.

The Clinical Research Informatics Core at the University of Alberta has elected to resolve this issue by developing a utility that provides the following functionality:

- Simplicity of use by data management staff (not and end user tool).
    - Invoked from the command line.
    - Creates an XML 'dump' of all critical study data
    - Fast – not encumbered by a user interface or web based delivery
    - Output XML readable directly via the SAS Libname engine (requires minimal coding to import the data into SAS).
- Generates XML output directly from the PostgreSQL database.
- Leverages existing study metadata to generate SAS labels and format catalog.
    - Translates response options to SAS formats/informats
    - Translates OpenClinica null values to SAS special missing values.

The Approach

No two studies are alike and the structures required for a SAS based data management database will vary from study to study. The export utility must require minimal to no configuration and must be able to handle all legal OpenClinica data. It must therefore be capable of translating complex OpenClinica data structures and names into simple SAS data sets and formats. OpenClinica CRFs have a hierarchical structure consisting of CRF, sections, groups and items. Items and their response options map directly to SAS variables (columns) and formats. The relationship between groups and sections is more complex. In order to accommodate CRFs, Sections and Groups consistently from study to study they are processed as follows:

CRFs

There is no mapping between the CRF on which data items originate. However CRF names and versions are written into the exported data. If the data manager requires to combine date from multiple CRFs and versions this can be done in SAS following the export. The CRF name and version may be retained in the data for data management purposes.

Sections

We have assumed that CRF sections are used to logically group data items on a CRF. CRF sections therefore map to an individual SAS data set.

- One CRF containing multiple sections results in one output data set per section.
- If section names are duplicated across multiple CRFs within the same study one SAS data set will result from each CRF/section combination. Under these circumstances numbers will be appended to the output data set name to make the name unique within the study.

Data set names are derived from the OpenClinica section_label.

Data set labels are derived from the OpenClinica section_title.

Groups

We have assumed that logical CRF design practices will group related data within a CRF section. For this reason CRF groups are handled as follows:

- Ungrouped data and grouped data are 'merged' into the same output record. (All the data items on the section are represented in the output data set).
- Ungrouped data repeat for each record from the groups. (A technique commonly used by clinical data managers).
- Rows from groups are numbered based on their position (row number) in the group.
- Where multiple groups are present in the same section data from the groups is merged based on the row number.
- Where there are unequal numbers of rows in a sections groups records will be 'padded' with null values.

Mapping sections and groups in this way will give the user freedom to recombine data in SAS to best reflect complex relationships between multiple groups.

Response Options and Values

The export utility builds an output table called fmtlib. This table is designed to be used by PROC FORMAT as an 'input control data set' and can therefore be used to automatically generate SAS formats and informats directly from the data contained in OpenClinica. OpenClinica however does not follow the same naming conventions as SAS and only enforces unique naming within a CRF, not across the study as a whole. Since SAS requires format names to be unique within a catalog and since each study will usually only use a single format catalog the export has to guarantee unique naming for all the formats in a study.

- SAS format names are based on the CRF item's response_label.
- Illegal SAS format names are modified to make them legal.
- Format names are de-duplicated by appending an ordinal followed by an underscore (since format names cannot end in a digit).
- If two different CRFs include identical response options (based on name and response values) only one format is produced.

Null Values

Informats

OpenClinica null values are translated into SAS 'special missing values'. This is achieved by building informats (in the fmtlib data set) that provide a translation between the textual null value representation and the SAS notation for a numeric special missing value as follows:

Informat BESTNULL, used to read numeric CRF items into SAS.

OpenClinica Null Value	SAS Missing Value
ASKU	.K
NA	.A
NASK	.D
NI	.I
NP	.P
OTH	.O
UNK	.U
OTHER	BEST10.

Informat CRFDATE, used to read CRF date items into SAS.

OpenClinica Null Value	SAS Missing Value
ASKU	.K
NA	.A
NASK	.D
NI	.I
NP	.P
OTH	.O
UNK	.U
OTHER	IS8601DA.

Formats

In order to format the SAS data correctly formats are built from OpenClinica response values. These include the values specified during CRF design but also include translations for all possible OpenClinica null values (or more specifically the SAS missing values created from them).

For example, a yesno format:

0	No
1	Yes
.U	UNK
.A	NA
.D	NASK
.I	NI
.K	ASKU
.O	OTH
.P	NP

Multi-select Items

One of the current challenges of the OpenClinica export is that data for multi-select items are exported into a single column. Whilst some statistical packages (Stata?) can handle this relatively easily the average data manager requires separate columns in the database to represent each possible data value. For this reason muloti-select and checkbox items are expanded.

- A new column is created for each possible response value
- Columns are named based on the item name but with a digit appended to ensure uniqueness.
- SAS labels are applied to the columns based on the original item_label plus the response_option_text for the response option. This makes it clear to the SAS user where the data originated.
- If the indicated response option is selected then the column will be populated with a 1. If it is not selected then the column is populated with a 0 (zero).

PHI

Data flagged as Protected Healthcare Information is exported as it occurs in the database. (No specific code has been written to handle PHI data.)

Usage

Installation, Execution & File Location

The utility is written in Java and requires access to a copy of the JDBC PostgreSQL driver package. No installation is required. The simplest was to run the utility is to save it to your hard drive and then execute it from a command prompt. Currently a copy of the PostgreSQL jar file must be located in the same folder.

The utility is launched as follows:

```
C:> java -jar export.jar
```

It will prompt for connection details, displaying and using defaults where appropriate. The defaults should work for a default OpenClinica installation on the local machine but will almost certainly need to be modified to run against a production database.

You may need to check that Postgres will accept connections from the machine on which the utility is run and that the appropriate ports are open through any firewall that may be in place.

Command Line Options

The application now supports a number of command line options. For a full list of currently supported options issue the following command:

```
C:> java -jar export.jar -?
```

At time of writing supported usage and options are:

```
Usage: java [-javaoptions] -jar export.jar [-option value] [-option value]...

Where -javaoptions typically needed:
        -Xmx###m        Total memory to allow this export to use during runtime
        -Xms###m        Total memory to initially allocate

Where -options value include:
         -h | -host     host machine where database is located (default: localhost)
                        -eg: 129.128.111.11 or med.servername.com

        -p | -port      port where database is located (default: 5432)

          -soid
          -study_oid    study to export based on oid as shown in OpenClinica

          -sname
          -study_name   study to export based on study name
                        -if study name contains whitespace or special characters, it must be enclosed in
                       quotes. eg: -study_name "my study name"

          -sid
          -study_id     study to export based on primary key id in database

          -U | -user    username used to access database containing study for export

          -P
          -password     users password to access database

          -D
          -database     database name containing OpenClincia (default: openclinica)

          -mapFile      map file name created during export

          -xmlFile      export file name to create

          -nomap        instructs export to not create a map file

          -?
          -help   print this help message
```

Reading Exported Data with SAS

The exported data is structured to be read using the SAS XML libname engine. The process is as follows:

1. Define a libname to represent the exported data. This must specify the location of the export data file and the XML map used to interpret it.
```
Libname ocdata      xml 'r:\export\study1.xml'
                    xmlmap="r:\export\study1.map.xml"
                    access=readonly;
```

2. Define the location of the SAS format catalog
```
libname library     'r:\export\formats';
```

3. Create the SAS format catalog.
```
proc sort data=ocdata.fmtlib out=work.fmtlib;
       by fmtname type start;
run;
proc format cntlin=work.fmtlib library = library fmtlib;
run;
```
4. Reassign the XML library. This makes sure the newly created informats are accessible by the XML libname engine.
```
Libname ocdata      xml 'r:\export\study1.xml'
xmlmap="r:\export\study1.map.xml"
access=readonly;
```
5. Open and use the data. (For instance in a data step, PROC DATASETS, etc). In this example we copy all of the data (except for the fmtlib data which was used to create the format library) out of the XML file into native SAS data sets in the work library.
```
proc datasets library=ocdata;
       copy out=work;
       exclude fmtlib;
quit;
```

Note that the SAS 9.1.3 version of the XML engine does not support data set labels (they will be empty in the SAS data), the 9.2 version does. However the 9.2 XML engine does not allow the user to view the data set interactively (which is useful during testing). The example above uses the 9.1.3 engine. To use the 9.2 engine replace 'xml' in the libname statement with 'xml92'.

Known Issues

Memory Usage

There is a known issue relating to the way the Postgres JDBC drivers return the data and application memory usage. Although we have taken steps to mitigate the problem you may get an out of memory error when working on relatively large studies (thousands of subjects with multiple visits and/or CRFs.) If this is the case you can increase the maximum Java heap size using a command line option:

```
C:\ java –Xmx256m -jar export.jar
```

By default the Java heap has a maximum size of 128 MB. In the above example the maximum heap size is doubled to 256 MB. This is adequate for our largest study but may not be sufficient in all environments.

CRF Status

The application currently exports only completed CRFs. It is intended ultimately to export all CRF data and to include a table of CRF status information.

Execution Example

```
C:> java -jar export.jar
----------------------------------------
             Export Output:
----------------------------------------
          MAP FILE: export.map.xml
       EXPORT FILE: export.xml
----------------------------------------
Postgresql driver loaded

Enter Database url (default: localhost):
Database port (default: 5432):
Database name (default: openclinica):
username (default: clinica):
password:

Enter Export file name (default: derived from study):
Enter Map file name (default: derived from study):

Successful connection to database openclinica on jdbc:postgresql://localhost:5432/

Please choose a study:
----------------------
 1) Study1
 2) Study2
 3) Study3
 4) Study4
==> 1

Retrieving study metadata
Creating subject table
Writing formats to .xml file
Writing subjects to .xml file
Retrieving study item data
Writing study item data to file
Complete
Files generated:   study1.map.xml
                   Study1.xml
```

SAS Program Example

```
/* Define the location of format library (the formats will be created from the XML file)
*/

libname library     'r:\export\test2';

/* Define the location of the definition for the XML file that contains the exported data

   The xml92 libname is an enhanced version released with SAS 9.2. It supports data set
   labels (SAS 9.1.3 did not), however it does not support interactive viewing of the
   data. If you want to view XML based data sets or if you do not have SAS 9.2 then you
   will need to use the xml engine rather than the xml92 engine.
*/

libname ocdata clear; /* not required but useful during testing */

libname ocdata xml92  'r:\export\satudy1.xml' xmlmap="r:\export\study2.xml.map"
                       access=readonly;

/* Read the formats in from the exported XML and create the format library */

proc sort data=ocdata.fmtlib out=work.fmtlib;
    by fmtname type start;
run;

proc format cntlin=ocdata.fmtlib library=library fmtlib;
run;

/* Reassign the library then copy the data to the work library (by way of example). This
   will copy all the 'tables' from the XML with the exception of the one we used to create
   the format library. */

libname ocdata        xml92  'r:\export\satudy1.xml' xmlmap="r:\export\study2.xml.map"
                              access=readonly;

proc datasets library=ocdata;
       copy out=work;
       exclude fmtlib;
quit;

/* That's it! */
```

Source Code

Program source is available on request. Should there be demand from the OpenClinica community we will establish an online code repository and open the application up for community development.

Additional Information

To provide feedback or for additional information contact:

Rick Watts B.Sc, FICR, CSci
Team Lead, Clinical Research Informatics Core
Women & Children's Health Research Institute
University of Alberta
1047 RTF, 8308 114 Street
Edmonton, AB Canada T6G 2V2
Tel: (780) 248 1170
Fax: (780) 407 1982
Email: rick.watts@ualbeta.ca

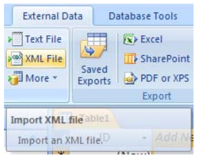
Appendix

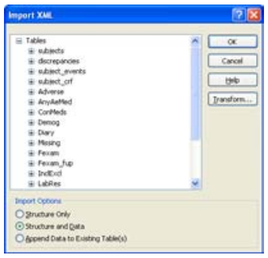
Importing Data into MS Office Applications

Coincidentally the XML data file generated by this application can be imported directly into both MS Access and Excel, although labels and formats are not carried through to these applications. The notes below provide examples but are not intended to be exhaustive.

Access 2007

1. Start Access and create a new, empty database.
2. Select the 'External Data' tab
3. In the 'Import' section click 'XML file'
4. Browse to your xml file and clock OK.
5. Click OK on the 'Import XML' dialog.
6. Click 'Close' to 'Save Import Steps'.
7. Access opens the XML file and creates a new table for each table defined in the XML (one per CRF section).



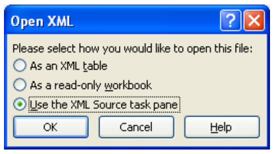


Tables
- subjects
- discrepancies
- subject_events
- subject_crf
- Adverse
- AnyAeMed
- ConMeds
- Demog
- Diary
- Missing
- Fexam
- Fexam_fup
- IndExcl
- LabRes

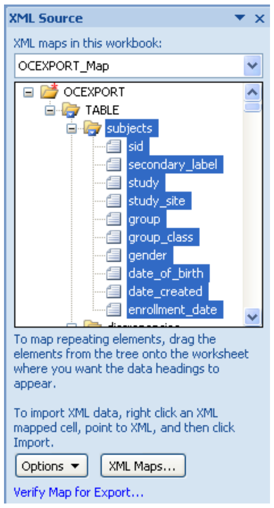
Import Options
- Structure Only
- [✓ Structure and Data]
- Append Data to Existing Table(s)

Excel 2007

1. Click File -> Open and select your XML file. Click 'Open'.
2. Select 'Use the XML Source Task pane
3. With your mouse drag a table from the XML Source task pane onto an empty sheet in your workbook.
4. Excel will paste the column headers into the sheet and display the 'Design' tab.
5. Click 'Refresh' on the design tab. Excel will pull the data for the selected table into your sheet.

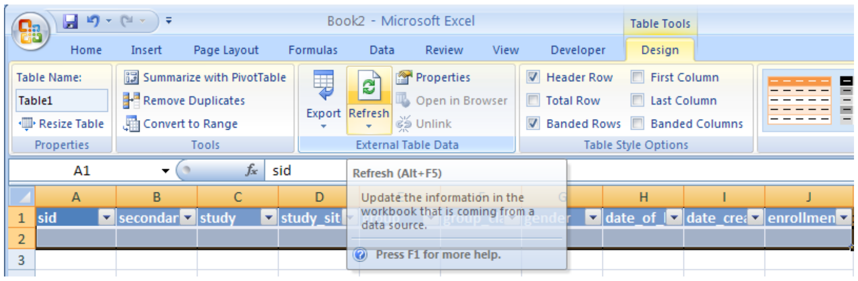


Please select how you would like to open this file:
- As an XML table
- As a read-only workbook
- [✓ Use the XML Source task pane]



To map repeating elements, drag the elements from the tree onto the worksheet where you want the data headings to appear.

To import XML data, right click an XML mapped cell, point to XML, and then click Import.



Update the information in the workbook that is coming from a data source.