PhUSE 2012

Paper TS04

# Let SAS handle your CDISC compliance check: automating OpenCDISC Validator in SAS

Edwin J. van Stein, Chiltern International, The Netherlands

## ABSTRACT

OpenCDISC Validator is a free, open source tool for checking clinical data compliance with CDISC standards. The input is data that you can create in SAS®, the tool can be called from the command line, and the output can be read in SAS, so running a compliance check can be handled completely by SAS on a range of operating systems. The purpose of this paper is to explain how to run OpenCDISC Validator from within SAS and automatically read the results in SAS for further processing. The examples that are given are tested on SAS on Microsoft Windows and Sun Solaris.

## INTRODUCTION

OpenCDISC Validator is a free, open source tool for checking clinical data compliance with CDISC standards, such as SDTM, ADaM, SEND, and Define.xml, as well as custom data definitions. The tool is written in 100% Java and includes a command line interface, which makes it usable on just about any operating system that can run Java. The input data can be either SAS V5 XPORT files or delimited files. The output of compliance checking is captured in Excel or CSV files when using the graphical user interface. Since the input is data that you can create in SAS, the tool can be called from the command line, and the output can be read in SAS, running a compliance check can be handled completely by SAS on a range of operating systems. In an environment with for instance a UNIX SAS server and a Windows client this can save valuable time because often the server has more computational power than the client and the data to be checked doesn't have to be transferred over a network.

This paper will focus on checking compliance of SAS XPORT files containing clinical data against Define.xml and the SDTM standards from the Clinical Data Interchange Standards Consortium. OpenCDISC Validator can also validate and generate Define.xml, but that is out of scope for this paper and will only be briefly mentioned. Validation of ADaM, SEND, or custom tabular data is similar to checking SDTM data, but just with another configuration file.

Please note that the examples in this paper were written on Microsoft Windows, which affects the directory delimiter used in the examples (\ as opposed to / on Sun Solaris (UNIX)). However, the overall idea of invoking the OpenCDISC Validator command line interface from within SAS will work on any operating system that can run SAS and Java. The latest release of OpenCDISC Validator during the time of writing this paper was version 1.3.

## OPENCDISC COMMAND LINE INTERFACE

OpenCDISC Validator is shipped with a graphical user interface as shown in Figure 1.

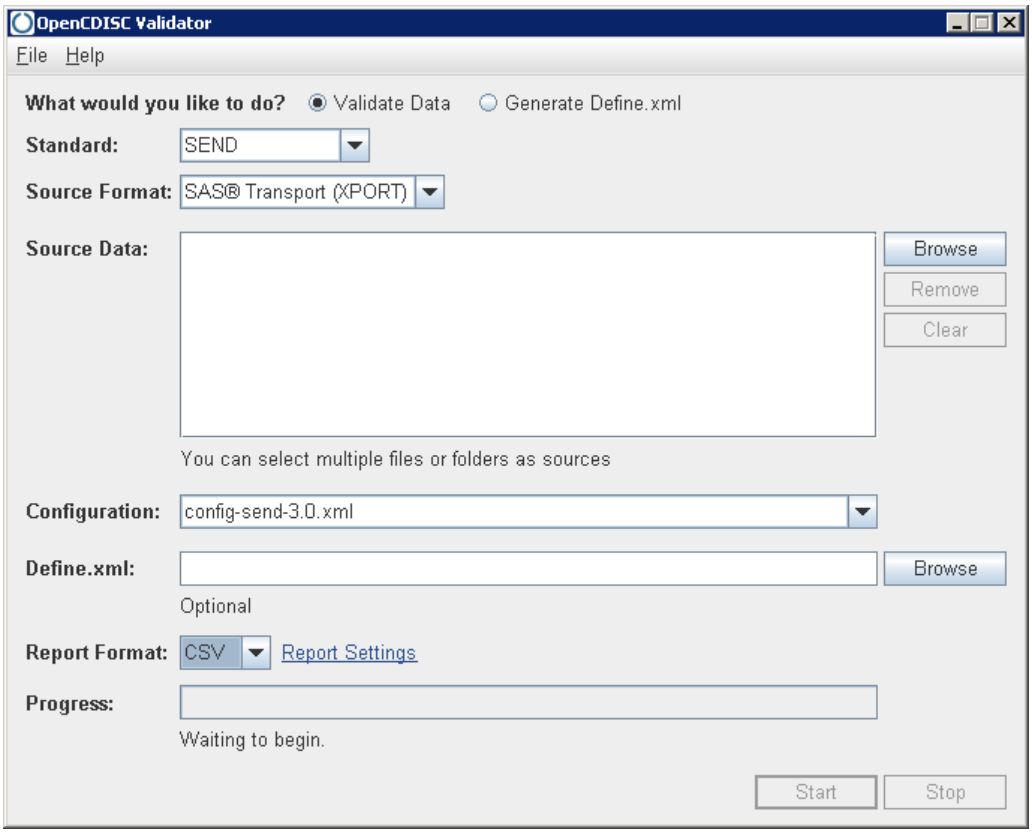



**Figure 1 Validator graphical user interface**

All parameters that can be set using the graphical user interface can also be supplied at the command line. The command line interface can be invoked by going to the directory where OpenCDISC is installed (on most operating systems done using the CD command) and then calling the Java application launcher (for instance java.exe on Windows) whilst specifying the Java archive (.jar) file of the OpenCDISC command line interface and any Java options or OpenCDISC parameters needed:

```
cd [location of OpenCDISC]
[Java application launcher location\]java [options] -jar [OpenCDISC jar] [parameters]
```

If Java is properly installed then the location of the Java application launcher is often not needed. Some Java options that are normally set if you run the graphical user interface on Windows are "-XX:+HeapDumpOnOutOfMemoryError -Xms256m -Xmx1024m". See the Performance section below for more information on these options. OpenCDISC Validator has a number of parameters that can be specified at the command line. The parameters relating to the 3 tasks that OpenCDISC Validator can perform can be found in table 1.

**Table 1 OpenCDISC Validator parameters**

| Parameter and default value | Task * | | | Values |
|---|---|---|---|---|
| | 1 | 2 | 3 | |
| -task=validate | ✓ | ✓ | ✓ | 'validate' or 'generate' |
| -type=sdtm | ✓ | ✓ | ✓ | 'sdtm', 'define' or 'custom' (custom includes ADaM, SEND and custom tabular data when using the command line interface) |
| -source= | ✓ | ✓ | ✓ | absolute or relative path to the source data |
| -source:type=sas | ✓ | | ✓ | 'sas' or 'delimited' |
| -source:delimiter=, | ✓ | | ✓ | only relevant if source:type is set to delimited |
| -source:qualifier=" | ✓ | | ✓ | only relevant if source:type is set to delimited |
| -config= | ✓ | ✓ | ✓ | absolute or relative path to the configuration file |
| -config:define= | ✓ | | | absolute or relative path to Define.xml |
| -config:codelists= | | | ✓ | absolute or relative path to an Excel file with codelists |
| -report= | ✓ | ✓ | | absolute or relative path to the report |
| -report:type=excel | ✓ | ✓ | | 'excel', 'csv' or 'xml' † |

| Parameter and default value | Task * | | | Values |
|---|---|---|---|---|
| | 1 | 2 | 3 | |
| -report:cutoff=1000 | ✓ | ✓ | | maximum number of times a message should be reported when -report:type=excel ‡ |
| -report:overwrite= | ✓ | ✓ | | 'yes' or 'no' |
| -output= | | | ✓ | absolute or relative path to output |
| -output:overwrite= | | | ✓ | 'yes' or 'no' |

* Task 1 stands for validation of data sets; task 2 stands for validation of Define.xml; task 3 stands for generating Define.xml

† Up to version 1.2.1 of OpenCDISC Validator 'html' has also been an option for -report:type, but support was dropped because rendering the output in web browsers was too resource intensive

‡ Up to version 1.2.1 of OpenCDISC Validator there was a bug involving -report:cutoff where one message less would be printed than specified (e.g. if cutoff was set to 10 then only a maximum of 9 messages would be displayed)

A special note should be made about the -source:qualifier parameter. If the default value (double quote) should be used then this parameter should not be specified in the command. The command line parser is not set up to handle specifying double quote explicitly and thus OpenCDISC Validator will fail.

When validating data sets the -source parameter can be a single file (for instance an XPORT file for a single SDTM domain), by specifying the path to that specific XPORT file, or all files in a directory (for instance all XPORT files for all SDTM domains for a clinical study), by using asterisk as wildcard (path-to-the-files*.xpt). Assuming Java has been installed in C:\sas\java, OpenCDISC Validator 1.3 was installed in C:\sas\opencdisc and the data to test can be found in C:\sas\testdata, OpenCDISC Validator can be run with the following command:

```
cd C:\sas\opencdisc & C:\sas\java\bin\java -XX:+HeapDumpOnOutOfMemoryError -Xms256m -Xmx3072m -jar lib\validator-cli-1.3.jar -source=C:\sas\testdata\*.xpt -config=config\config-sdtm-3.1.2.xml
```

This checks compliance of all XPORT files found against the SDTM 3.1.2 standard. The output is stored in an Excel file called opencdisc-report-yyyy-mm-ddThh-mm-ss.xls (where yyyy-mm-ddThh-mm-ss is the date and time in ISO 8601 format with colons replaced by dashes) with a report cutoff of 1000 in the report directory where OpenCDISC Validator was installed.

## CALLING IT FROM SAS

One of the first things to decide when running OpenCDISC Validator from within SAS is whether SAS or the tool itself should do parameter checking. If the parameters result in OpenCDISC Validator not being able to perform the run then it will fall over fairly quickly. So to make the SAS program easier it can be considered to leave it up to the tool. However, if parameter checking is done in SAS then you can customize what needs to be checked and what feedback should be given to the user. For simplicity in this paper parameter checking will be done by the tool itself.

There are multiple ways of invoking system commands from within SAS like X statements, X commands, FILENAME PIPE statements, the CALL SYSTEM routine, SYSTASK statements and %SYSEXEC calls. The two most commonly used ways are X statements and FILENAME PIPE statements (or SAS supplied macros %xlst and %xlog, which use FILENAME PIPE statements). The advantage of using a FILENAME PIPE statement is that the command line feedback can be read by SAS and thus error message from OpenCDISC Validator can be shown.

There are some things to take into account when using FILENAME PIPE (or any of the other methods) to execute a system command:

- The system option XCMD should be set (as opposed to NOXCMD). Otherwise none of the ways to invoke system commands from within SAS will work.
- When running multiple commands (for instance first a change directory command and then Java) the separator between commands depends on the operating system. On Windows & (ampersand) is the separator whereas on UNIX ; (semicolon) is the separator.
- The directory delimiter is operating system dependent as mentioned in the introduction. On Windows \ (back slash) is the delimiter to use and on UNIX / (front slash) is the delimiter.
- If an application that uses standard input, output and error commands on Windows generates an error then the message will be redirected to STDERR, which means the SAS log as opposed to for instance the DATA step that invoked the FILENAME PIPE. To ensure that error messages are sent to STDOUT for further processing a redirection sequence can be added to the FILENAME PIPE statement. The redirection sequence 2>&1 ensures that STDERR output (file handle 2) is redirected to STDOUT (file handle 1).

When writing a program that only ever will be run on a single operating system then the directory delimiter, command separator and redirection sequences can be hardcoded in the program. When a program should be able to run on different systems then these can be made dependent of the value for macro variable &SYSSCP.

When using X statements, X commands, CALL SYSTEM routines or %SYSEXEC calls on Windows the XSYNC and XWAIT options are also important. If XSYNC is set then SAS will wait for the command to complete execution. If NOXSYNC is set then SAS will not wait, but will immediately continue processing the rest of the program. If XWAIT is set then EXIT will need to be typed in the command prompt to return to SAS. If NOXWAIT is set then the command prompt will close automatically upon finishing execution.

An example of going into a specific directory and then invoking Java to check the version whilst ensuring that regular output and errors are captured in a DATA step can be seen below. In this case any error messages from the cd command end up in the log (no redirection sequence specified), and error messages from the Java command end up in the DATA step (redirection sequence specified).

```
filename javachk pipe "cd C:\sas\opencdisc & C:\sas\java\bin\java -version 2>&1";

data _javachk;
   infile javachk;
   input;
   length cmd $2000;
   cmd=_infile_;
run;
```

The resulting data set produced by the code above contains all feedback from invoking the Java binary with the -version option as shown in Figure 2.

VIEWTABLE: Work._javachk

| | cmd |
|---|---|
| 1 | java version "1.6.0_31" |
| 2 | Java(TM) SE Runtime Environment (build 1.6.0_31-b05) |
| 3 | Java HotSpot(TM) Client VM (build 20.6-b01, mixed mode, sharing) |

**Figure 2 Data set containing feedback from Java binary**

Similar to SAS invoking the Java binary to get the Java version it can run the command that was used earlier to run OpenCDISC Validator.

```
filename runoc pipe "cd C:\sas\opencdisc & C:\sas\java\bin\java -
XX:+HeapDumpOnOutOfMemoryError -Xms256m -Xmx3072m -jar lib\validator-cli-1.3.jar -
source=C:\sas\testdata\*.xpt -config=config\config-sdtm-3.1.2.xml 2>&1";

data _runoc;
   infile runoc;
   input;
   length cmd $2000;
   cmd=_infile_;
run;
```

The feedback from OpenCDISC Validator, which would normally appear in the command prompt, now ends up in a data set as shown in Figure 3.

VIEWTABLE: Work._runoc

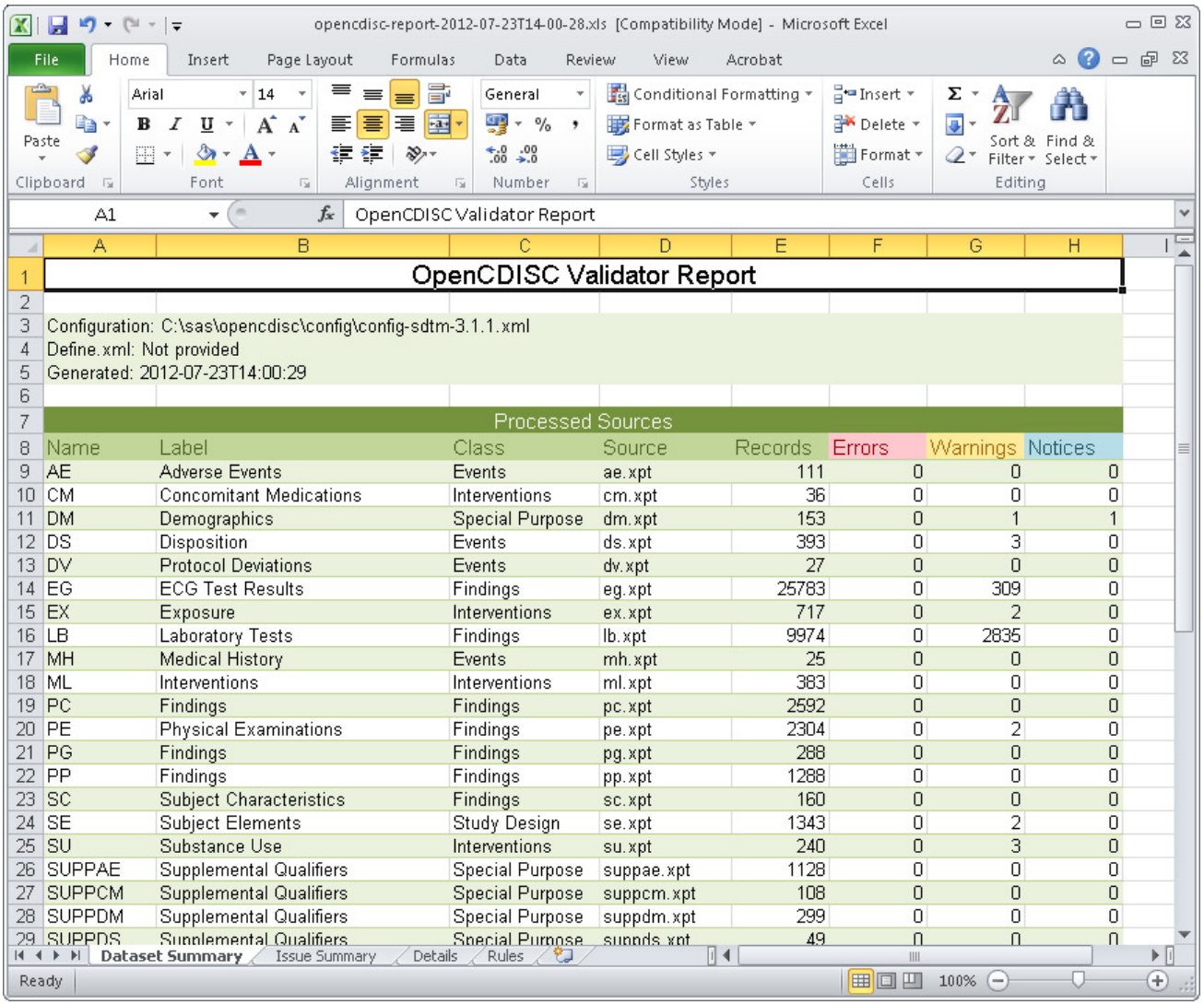
| | cmd |
|---|---|
| 1 | Beginning validation, please wait... |
| 2 | The validation has completed. |

**Figure 3 OpenCDISC Validator feedback**

## OUTPUT

The default output from OpenCDISC Validator is an Excel file with a summary by data set, a summary by issue, a detailed report of the issues and the rules that were used. The Excel file is very useful and looks good, but for further

processing in SAS on an operating system like UNIX a license for SAS/ACCESS to PC Files® and a corresponding PC Files Server would be needed. This is where the CSV and XML output options come in.



opencdisc-report-2012-07-23T14-00-28.xls [Compatibility Mode] - Microsoft Excel

A1: OpenCDISC Validator Report

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | OpenCDISC Validator Report | | | | | | | |
| 2 | | | | | | | | |
| 3 | Configuration: C:\sas\opencdisc\config\config-sdtm-3.1.1.xml | | | | | | | |
| 4 | Define.xml: Not provided | | | | | | | |
| 5 | Generated: 2012-07-23T14:00:29 | | | | | | | |
| 6 | | | | | | | | |
| 7 | Processed Sources | | | | | | | |
| 8 | Name | Label | Class | Source | Records | Errors | Warnings | Notices |
| 9 | AE | Adverse Events | Events | ae.xpt | 111 | 0 | 0 | 0 |
| 10 | CM | Concomitant Medications | Interventions | cm.xpt | 36 | 0 | 0 | 0 |
| 11 | DM | Demographics | Special Purpose | dm.xpt | 153 | 0 | 1 | 1 |
| 12 | DS | Disposition | Events | ds.xpt | 393 | 0 | 3 | 0 |
| 13 | DV | Protocol Deviations | Events | dv.xpt | 27 | 0 | 0 | 0 |
| 14 | EG | ECG Test Results | Findings | eg.xpt | 25783 | 0 | 309 | 0 |
| 15 | EX | Exposure | Interventions | ex.xpt | 717 | 0 | 2 | 0 |
| 16 | LB | Laboratory Tests | Findings | lb.xpt | 9974 | 0 | 2835 | 0 |
| 17 | MH | Medical History | Events | mh.xpt | 25 | 0 | 0 | 0 |
| 18 | ML | Interventions | Interventions | ml.xpt | 383 | 0 | 0 | 0 |
| 19 | PC | Findings | Findings | pc.xpt | 2592 | 0 | 0 | 0 |
| 20 | PE | Physical Examinations | Findings | pe.xpt | 2304 | 0 | 2 | 0 |
| 21 | PG | Findings | Findings | pg.xpt | 288 | 0 | 0 | 0 |
| 22 | PP | Findings | Findings | pp.xpt | 1288 | 0 | 0 | 0 |
| 23 | SC | Subject Characteristics | Findings | sc.xpt | 160 | 0 | 0 | 0 |
| 24 | SE | Subject Elements | Study Design | se.xpt | 1343 | 0 | 2 | 0 |
| 25 | SU | Substance Use | Interventions | su.xpt | 240 | 0 | 3 | 0 |
| 26 | SUPPAE | Supplemental Qualifiers | Special Purpose | suppae.xpt | 1128 | 0 | 0 | 0 |
| 27 | SUPPCM | Supplemental Qualifiers | Special Purpose | suppcm.xpt | 108 | 0 | 0 | 0 |
| 28 | SUPPDM | Supplemental Qualifiers | Special Purpose | suppdm.xpt | 299 | 0 | 0 | 0 |
| 29 | SUPPDS | Supplemental Qualifiers | Special Purpose | suppds.xpt | 49 | 0 | 0 | 0 |

Dataset Summary | Issue Summary | Details | Rules

Ready

**Figure 4 OpenCDISC Validator default Excel output**

The CSV output from OpenCDISC Validator is, as its name implies, a comma separated list of the issues found with no additional summaries. The columns that are included in the output are: Name, Record, Variable, Value, Rule ID, Message, Category, Type and Severity. Most of these are self explanatory and also appear on the Details tab in the default Excel output. One thing to note, however, is that currently the Type column actually contains the value from the Severity column. Reading a CSV file in SAS can be done in multiple ways. The easiest is using PROC IMPORT using GETNAMES and GUESSINGROWS statements as shown below.

```
proc import datafile="C:\sas\opencdisc\reports\report.csv" out=_report dbms=csv
replace;
   datarow=2;
   getnames=yes;
   guessingrows=32676;
run;
```

A disadvantage of using PROC IMPORT can be that SAS decides, based on the data, what the type and name of the variable will be. If more control is needed a DATA STEP with INFILE statement can be used. However, the disadvantage of that is that the report may change in future releases, in which case the programs also need updating.

When using the command line interface XML is also an option for the report. Converting an XML report to SAS data sets can be done using an XMLMap file combined with a LIBNAME XML statement. Quite a lot of papers have been published about doing this, so this will not be discussed further in this paper.

The XML report has some different information from the Excel and CSV reports. The main advantage of Excel and CSV is that for a single issue they can contain values for multiple variables so that the report contains all relevant information for the issue found. In XML it only says which record the problem occurs in, but not the value for all relevant variables for that record. However, the relevant data can be merged in SAS using the record number. An advantage of the XML report is that a slightly longer description of the issue is given. However, a major disadvantage of the XML report is that in some cases not all information is present to find the actual issue. For instance an issue reported in the CSV report can read:

```
"EG",,"VARIABLE","EGEVAL","SD0057","SDTM Expected variable not
found","Metadata","Warning"
```

However, that same issue is reported in the XML report as:

```
<issue code="SD0057" time="2012-07-23T14:26:46">
      <source name="EG"
              location=" C:\sas\testdata\eg.xpt"
              label="ECG Test Results"
              class="Findings">
         <details/>
      </source>
      <message>SDTM Expected variable not found</message>
      <description>Variables described in SDTM as Expected should be included in the
dataset</description>
      <type>Warning</type>
      <category>Metadata</category>
   </issue>
```

In this case the name of the variable that is missing is not given in the XML report. This means that the CSV report is currently more complete. Additional information can be added in SAS by linking to the actual clinical data and adding information about the validation rules used by reading in the configuration file, which is simply another XML file.

As mentioned before support for HTML output has been dropped in OpenCDISC Validator 1.3. This used to be created from the XML output using SAXON XSLT, but rendering the output in browsers was too resource intensive.

## PERFORMANCE

By default OpenCDISC Validator will use a single core. However, OpenCDISC Validator can use multiple cores, which even basic computers have nowadays. To use multiple cores open \lib\properties\settings.properties and edit the line containing the setting called Engine.ThreadCount. This will allow OpenCDISC Validator to validate multiple data sets simultaneously. On very powerful machines, with 10+ cores, allowing OpenCDISC Validator to use all cores does not necessarily make it run faster than assigning fewer cores, because for most studies you will be waiting for the largest data set to be validated. This is illustrated in Figure 5, where assigning more than 5 cores on a 10+ core machine did not make the run faster, because during validation of the largest data set the other 4 cores were enough to validate all the other data sets.

When running the graphical user interface on a Windows machine some default Java options are used: "-XX:+HeapDumpOnOutOfMemoryError -Xms256m -Xmx1024m". The Xms argument tells the Java Virtual Machine to allocate a certain amount of memory when the program starts. The benefits of decreasing or increasing Xms depend on your system. The Xmx argument tells the Java Virtual Machine the maximum amount of memory that the heap can reach, so this should never be lower than Xms. If you're getting OutOfMemory exceptions then the value for Xmx is too small for what you're doing and should be increased. When OpenCDISC Validator is not performing as fast as expected then trying different values for Xms could be considered. However, the impact may be low as shown by the tests performed for Figure 5. In case of doubt a safe bet is using the default values and only increasing Xmx on OutOfMemory exceptions.

The XX:+HeapDumpOnOutOfMemoryError option does what its name implies, it writes the heap to a file when an OutOfMemory exception occurs. This is useful if you are a Java programmer debugging OpenCDISC Validator, but for a simple validation run it's not relevant and can be safely ignored.

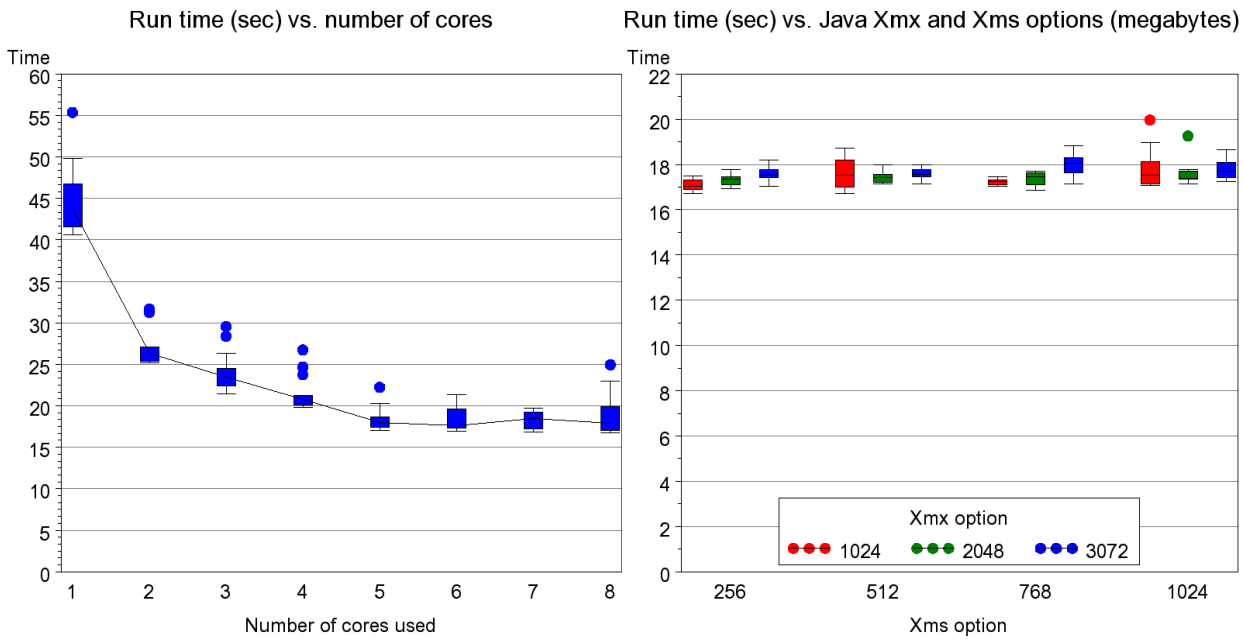



**Figure 5 OpenCDISC Validator performance benchmarks**

For the run time versus number of cores graph in Figure 5 per number of cores 16 measurements were performed. For the run time versus Java Xmx and Xms options graph per combination of Xmx and Xms 12 measurements were performed. In both graphs the whiskers represent the most extreme values within 1.5 interquartile ranges and individual symbols are the values outside of 1.5 interquartile ranges.

## TROUBLESHOOTING

When something goes wrong with running OpenCDISC Validator from SAS one of the first things to check is whether all parameters were entered correctly and whether everything points towards the correct directories. If exotic Java exceptions are shown, then it can be useful to first check whether a recent version of Java has been installed. In some cases moving to the most recent Java version can solve speed issues as well.

If problems persist and they also occur when running OpenCDISC Validator using the graphical user interface then the OpenCDISC community may be able to help. The OpenCDISC website has a support forum where developers and users post and answer questions on a regular basis.

## OTHER STATISTICAL SOFTWARE PACKAGES

Similar to SAS other statistical software packages can also run system commands and thus OpenCDISC Validator. Some examples of running system commands in other packages are:

- R-project using the system() function;
- Stata using the shell command;
- MATLAB using the system() function or simply the exclamation point character (!).

If a software package can run system commands and it's installed on an operating system supporting Java then it can run OpenCDISC Validator for you.

## CONCLUSION

Running OpenCDISC Validator from SAS is very easy and can save valuable time by not having to transfer XPT files between servers and clients and by leveraging multiple cores on servers. The type of report to use depends on whether further processing of the report is needed. If not, then the default Excel report can be used. If so, then the CSV report is the most complete and can be linked to the validation rules from the configuration file and to the actual clinical data in SAS.

## REFERENCES

- http://www.cdisc.org The Clinical Data Interchange Standards Consortium
- http://www.opencdisc.org OpenCDISC - An open source community of CDISC developers and users
- http://www.java.com/en/ Java
- http://www.phusewiki.org/wiki/index.php?title=Let_SAS_handle_your_CDISC_compliance_check:_automating_OpenCDISC_Validator_in_SAS PhUSE Wiki page with example code

## CONTACT INFORMATION

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

Edwin J. van Stein
Chiltern International
Crown Business Center Key-Point
Schipholweg 103
2316 XC Leiden
The Netherlands
Email: edwin.vanstein@chiltern.com / ejvanstein@gmail.com
Web: http://nl.linkedin.com/in/ejvanstein