

Introduction

Working in early phase drug development where many varied and disparate information stakeholders want immediate access to the study data, it can seem that I spend more of my time dealing with Excel rather than SAS. There have been many articles over the years dealing with transferring data from SAS® to Microsoft Excel ® (DDE, ODS, ADO, SAS v9, SAS Microsoft Office Add-in, XML, text files, SAS/IOM, SAS/SHARE), but I wish to focus on one aspect of these different methods which perhaps hasn't received enough attention. When left to it's own devices, Excel compromises data integrity by applying it's own formats.

The Problem

According to the Microsoft Knowledge Base Article ID 214233, Excel applies it's own formats to the data, based on the following criteria:

1. If a number contains a slash mark (/) or hyphen (-), it may be converted to a date format.
2. If a number contains a colon (:), or is followed by a space and the letter A or P, it may be converted to a time format.
3. If a number contains the letter E (in uppercase or lowercase letters; for example, 10e5), or the number contains more characters than can be displayed based on the column width and font, the number may be converted to scientific notation, or exponential format.
4. If a number contains leading zeros, the leading zeros are dropped.

The end result is corrupt data. So how can the SAS programmer safeguard the integrity of their data? Using Excel's "Text" format ensures the data is treated as just text and preserves the original values. Below are some examples of using the "Text" format using different import methods.

Table 1. Genuine Examples of Corrupted Data

Actual Intension	SAS Value	Excel Interpretation
% positively stained cells (categorical score)	1-5	01 – May *
questionnaire responses / number issued	60 / 90	2 / 3
protocol schedule time	01:30	01:30:00

* or more truly 38473 the Excel date serial value

via DDE

```
data _null_;
file xlcmds;
put '[select("r1c1:r10c2",
"r1c1")]';
put '[format.number("@")]';
run;
```

```
data _null_;
set sastable;
file sasdata;
put var1 '09'x var2 ;
run;
```

- Pre-select the region to be populated with SAS data
- Set the format to "Text" using the *format.number* function and "@" as the text format identifier.
- Row 1, column 1 to row 10, column 2 have been formatted to "Text" in readiness for the population by SAS data in the next DATA step.

via ODS

```
ods html file='c:\report.xls';
proc report data=sastable
nowindows;
column var1;
define var1 / display
style={htmlstyle="mso-number-
format:@"};
run;
quit;
ods html close;
```

- Use the STYLE option to set the mso-number-format to "@" as the text format identifier.
- Variable specific, enabling all output for var1 to be correctly formatted.
- Other variables can have other formats defined.

via VBA using ADO

```
obConnection.Provider =
"sas.LocalProvider.1"

obConnection.Properties("Data
Source") = "... folder
containing SAS tables ..."

obConnection.Open

obRecordset.Open " SAS table
name ", obConnection,
adOpenDynamic, adLockReadOnly,
ADODB.adCmdTableDirect

'format the cells to text
Range(Cells(1, 1),
Cells(obRecordset.RecordCount +
1,
obRecordset.Fields.Count)).Numb
erFormat = "@"

'add header row
'add detail rows
```

- Pre-select the region to be populated with SAS data.
- Cell A1 will contain the start of the data, but the end row and column depends on the number of records and variables in the SAS table.
- Use the *obRecordset.RecordCount* and *obRecordset.Fields.Count* properties to select the cells for formatting
- Import the data into formatted cells

via VBA and text files

```
Workbooks.OpenText Filename:=
"C:\temp\import.txt",
Origin:=xlWindows, StartRow:=1,
DataType:=xlDelimited,
TextQualifier:=xlDoubleQuote,
ConsecutiveDelimiter:=False,
Tab:=False, Semicolon:=False,
comma:=False, Space:=False,
Other:=True, OtherChar="|",
FieldInfo:=Array(Array(1, 2),
Array(2, 2), Array(3, 2), ...
Array(256, 2))
```

- The FIELDINFO argument is an array of two-element arrays, with each two-element array specifying the conversion options for a particular column.
- The first element is the column number (1-based)
- The second element is a constant specifying how the column is parsed. The number 2 means parse as text.
- If less array statements were defined than there are SAS variables, Excel still imports all the variables but the additional variables are open to data corruption issues.

Conclusion

While the knowledge of getting data from SAS to Excel is widely disseminated, it's limitations are not so widely known. Hopefully this poster has not only highlighted this issue, but has also put forward some simple solutions. The programmer must be proactive in ensuring data integrity and not leave this vital step to Excel.

Abbreviations

ADO – ActiveX Data Objects
DDE – Dynamic Data Exchange
IOM – Intergrated Object Model
ODS – Output Delivery System
VBA – Visual Basic for Applications
XML – Extensible Markup Language

Recommended Reading

Koen Vyverman (2001) "Using Dynamic Data Exchange to Export Your SAS® Data to MS Excel — Against All ODS, Part I", *Proceedings of the Twenty-sixth Annual SAS Users Group International Conference*, Paper 11.

Darren Key (2002) "Using SAS® Data To Drive Microsoft Office", *Proceedings of the Twenty-seventh Annual SAS Users Group International Conference*, Paper 123.

Vincent DelGobbo (2003) "A Beginner's Guide to Incorporating SAS Output in Microsoft Office Applications". *Proceedings of the Twenty-eighth Annual SAS Users Group International Conference*, Paper 52.

Vincent DelGobbo (2004) "Moving Data and Analytical Results between SAS and Microsoft Office", *Proceedings of the Thirtieth Annual SAS Users Group International Conference*, Paper 136.

Brian Fairfield-Carter (2004), "Instant SAS Applications With VBScript, Jscript, and dHTML ", *Proceedings of the Thirtieth Annual SAS Users Group International Conference*, Paper 9.

Ted Conway (2005) "Make Bill Gates and Dr. Goodnight Run Your SAS Code: Using VBA, ADO and IOM to Make Excel and SAS Play Nice", *Proceedings of the Thirtieth Annual SAS Users Group International Conference*, Paper 157.

SAS Institute Inc., "Welcome to the SAS 9.1 Data Providers: ADO/OLE DB Cookbook ", <http://support.sas.com/md/eai/oledb/index.htm>, Cary, NC: SAS Institute Inc.