Paper 032-2010

Creating Highly Customized, Binary Excel Workbooks Using XML

Alan Churchill, Savian, Colorado Springs, CO

ABSTRACT

This paper will illustrate a new method for generating binary Excel workbooks using SAS ® data. It will illustrate how you can go beyond ODS, proc export, Access engines, etc. to make highly-customized Excel workbooks using SAS data.

INTRODUCTION

In almost every SAS site, there is a need to interface with Excel. A large portion of that need is to export SAS datasets to Excel. However, a simple proc export is rarely what is needed since the data needs to be not only exported to Excel but to also be formatted in Excel. Several techniques are available to get the data from SAS to Excel and format it but they suffer significant limitations:

- ODS
  - Complex
  - Intertwines with business logic code violating UI separation policy
  - Does not have the necessary flexibility that is often needed with Excel
- DDE
  - Deprecated. New features not being added.
  - Hard to work with
  - Requires Excel on the computer and can run into locking issues with Excel.

In the field, significant limitations have been found for the above 2 approaches, especially for consultants. Specifically, speed of quickly replicating a client's existing Excel spreadsheets and how to automate the Excel file generation so that it is ready for production while being easy to replicate the client's existing formatting. Additionally, clients want binary files and not HTML or XML representations of those files.

SAVICELLSPRO

SaviCellsPro (SCP) is a binary executable written in .NET that is designed to address the limitations found in the above mentioned methods. SCP runs on the .NET Framework 3.5 or higher and has a number of advantages over traditional methods used by most SAS programmers:

- SCP does not require Excel to be installed and runs standalone.
- SCP creates true binary Excel files and supports both the Excel 2003 and 2007 file types
- SCP is straight-forward XML code so the Excel workbook can be described using any application that can generate text files
- SCP sits outside of SAS and therefore isolates the business logic from the presentation
- SCP has support for charting, printing, and other more complex Excel items

SCP uses an XML file at its heart to describe how to create a binary Excel file in either Excel 2003 or 2007 format. As a bonus, SCP can generate a PDF file as well as an HTML file of the Excel data.

SCP is installed on a Windows platform using a simple setup.msi file. Once it is installed, it can be called using the X command:

```
options noxwait noxsync;
```

```
x '"C:\Program Files\Savian\SaviCellsPro\scp.exe" "C:\temp\sample.xml"' ;
```

This command calls SCP and uses the contents of sample.xml to create the workbook. Here is an example of what the XML in sample.xml might contain:

```
<Actions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="..\..\XML\SaviCellsMain.xsd"
         xmlns:xi="http://www.w3.org/2003/XInclude"
         log="Log\SaviCells.log"
         appendLog="false">
       <Export>
              <Workbook excelFile="Samples\Output\simple" excelExtension="xls">
                     <Worksheet name="Class">

                            <!-- RAW DATA INSERTION -->

                            <Cell rc="R7C4">Name</Cell>
                            <Cell rc="R7C5">Gender</Cell>
                            <Cell rc="R7C6">Age</Cell>
                            <Cell rc="R7C7">Height</Cell>
                            <Cell rc="R7C8">Weight</Cell>

                            <!-- ColFill can be used to fill in a column with data. Default
                            delimiter is a ; if dlm is not specified -->

                            <ColFill
                     rc="R8C4">Alfred;Alice;Barbara;Carol;Henry;James;Jane;Janet;Jeffrey;John;J
                     oyce;Judy;Louise;Mary;Philip;Robert;Ronald;Thomas;William</ColFill>

                            <!-- RowFill can be used to fill in a row with data. Default
                            delimiter is a ; if dlm is not specified -->

                            <RowFill rc="R8C5">M;14;69;112.5</RowFill>
                            <RowFill rc="R9C5">F;13;56.5;84</RowFill>
                            <RowFill rc="R10C5">F;13;65.3;98</RowFill>
                            <RowFill rc="R11C5">F;14;62.8;102.5</RowFill>
                            <RowFill rc="R12C5">M;14;63.5;102.5</RowFill>
                            <RowFill rc="R13C5">M;12;57.3;83</RowFill>
                            <RowFill rc="R14C5">F;12;59.8;84.5</RowFill>
                            <RowFill rc="R15C5">F;15;62.5;112.5</RowFill>
                            <RowFill rc="R16C5">M;13;62.5;84</RowFill>
                            <RowFill rc="R17C5">M;12;59;99.5</RowFill>
                            <RowFill rc="R18C5">F;11;51.3;50.5</RowFill>
                            <RowFill rc="R19C5">F;14;64.3;90</RowFill>
                            <RowFill rc="R20C5">F;12;56.3;77</RowFill>
                            <RowFill rc="R21C5">F;15;66.5;112</RowFill>
                            <RowFill rc="R22C5">M;16;72;150</RowFill>
                            <RowFill rc="R23C5">M;12;64.8;128</RowFill>
                            <RowFill rc="R24C5">M;15;67;133</RowFill>
                            <RowFill rc="R25C5">M;11;57.5;85</RowFill>
                            <RowFill rc="R26C5">M;15;66.5;112</RowFill>

                            <!-- Cell can be abbreviated as C -->

                            <C rc="R27C7">Total</C>
                            <C rc="R27C8">=SUM(R[-19]C:R[-1]C)</C>
                     </Worksheet>
              </Workbook>
       </Export>
</Actions>
```

Below is a breakdown of the sections to explain what is happening:

XML Header

```
<Actions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="..\..\XML\SaviCellsMain.xsd"
         xmlns:xi="http://www.w3.org/2003/XInclude"
         log="Log\SaviCells.log"
         appendLog="false">
```

- `xsi:noNamespaceSchemaLocation="..\..\XML\SaviCellsMain.xsd"`

  Points to where the XML Schema file is located. This should not need to changed unless the location of SCP.exe is moved.

- `log="Log\SaviCells.log"` and `appendLog="false"`

  Points to the location of the processing log and whether the log should be overwritten or appended to. This is under the program files by default in the Log directory.

Start of workbook creation

```
<Export>
    <Workbook excelFile="Samples\Output\simple" excelExtension="xls">
        <Worksheet name="Class">
```

- `excelFile="Samples\Output\simple”`

  Specifies the location of the resulting Excel file.

- `excelExtension="xls"`

  Specifies the type of Excel file. Xls creates a 2003 and earlier Excel file and xlsx creates an Excel 2007 and later file.

- `<Worksheet name="Class">`

  Specifies the name of the worksheet in the workbook.

Worksheet creation

- `<Cell rc="R7C4">Name</Cell>`

  This is an example of raw data insertion. The word *Name* will be inserted at row 7, column 4. Cell can also be abbreviated as a C so you could have written the above as:

  ```
  <C rc="R7C4">Name</C>
  ```

- ```
  <ColFill
  rc="R8C4">Alfred;Alice;Barbara;Carol;Henry;James;Jane;Janet;Jeffrey;John;Joyce;Judy;Louis
  e;Mary;Philip;Robert;Ronald;Thomas;William</ColFill>
  ```

  A column in Excel can be easily filled with data by using the ColFill element. The items in the above element will be inserted at row 8, column 4 in the order listed. Optionally a dlm attribute can be used to specify an alternate delimiter than the default (semicolon).

- `<RowFill rc="R10C5">F;13;65.3;98</RowFill>`

  A row in Excel can be easily filled with data by using the RowFill element. The items in the above element will be inserted at row 10, column 5 in the order listed. Optionally, a dlm attribute can be used to specify an alternate delimiter than the default (semicolon).
  ```

ADDITIONAL EXCEL FEATURES

SCP comes with a number of additional features to simplify working with Excel. Every possible feature can be found in the Help directory under the SaviCellsPro directory in the Sample called Complete.xml. Some features:

- Styles
  - Styles can be overridden at any level including the cell
  - Style elements that can be modified include
    - Font name, color, size
    - Borders
    - Cell color
    - Number formatting
- Stylesheet support
  - Stylesheets controlling styles can be stored in a separate file
- Printing support
  - Sheet fitting
  - Header and footer support
  - Margins
  - Orientation
  - Workbook level settings and worksheet level settings
- Image insertion
  - Insert an image at any location in the worksheet
- Charting
- Command switches
  - Background processing

COMMAND LINE SWITCHES

SCP comes with a number of command line switches to help with processing:

- /b

  Allows for background processing meaning that no console messages will be written.

- /nolog

  Does not log any messages.

- File overrides

  Overrides the files specified in the XML. Specific overrides allowed are:

  - /excel=
  - /pdf=
  - /html=
  - /license=
  - /log=

If no feedback is desired and speed is critical, set the options for SCP to /b and /nolog as such:

```
options noxwait noxsync;
x '"C:\Program Files\Savian\SaviCellsPro\scp.exe" "C:\temp\sample.xml"' /b /nolog;
```

STYLESHEET SUPPORT

Styesheets enable separation of the cell/worksheet styles from the actual data similar to what is found in HTML. Stylesheets can be created as separate XML files and held on their own. Here is an example stylesheet:

```
<?xml version="1.0" encoding="utf-8" ?>
<Stylesheet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:noNamespaceSchemaLocation="..\..\XML\SaviCellsStylesheet.xsd"
            xmlns:xi="http://www.w3.org/2003/XInclude">
  <Styles name="CalibriHighlight">
    <Style name="Negative">
      <Font>Courier</Font>
      <FontColor>Red</FontColor>
    </Style>
    <Style name="Positive">
      <Font>Calibri</Font>
      <FontColor>White</FontColor>
      <CellColor>Black</CellColor>
    </Style>
    <Style name="Header">
      <Font>Calibri</Font>
      <FontColor>White</FontColor>
      <CellColor>Blue</CellColor>
    </Style>
    <Style name="PlainValue">
      <Font>Calibri</Font>
      <FontColor>Black</FontColor>
      <CellColor>#FFCC00</CellColor>
    </Style>
    <Style name="LargeHeader">
      <Font>Calibri</Font>
      <FontSize>16</FontSize>
      <Bold>true</Bold>
      <FontColor>Black</FontColor>
      <CellColor>White</CellColor>
    </Style>
    <Style name="TOC">
      <Font>Calibri</Font>
      <FontSize>8</FontSize>
      <Bold>true</Bold>
      <FontColor>Black</FontColor>
      <CellColor>Yellow</CellColor>
    </Style>
    <Style name="Highlight">
      <Font>Calibri</Font>
      <FontSize>8</FontSize>
      <Bold>true</Bold>
      <Italic>true</Italic>
      <FontColor>White</FontColor>
      <CellColor>Red</CellColor>
    </Style>
  </Styles>
</Stylesheet>
```

And its usage:

```
<Export>

<!-- ===== WORKBOOK ===== -->


 …hidden xml…

<Workbook excelFile="Samples\Output\sample" excelExtension="xls" use1904Dates="false"
pdfFile="Samples\Output\sample" htmlFile="Samples\Output\sample">

      <!-- Stylesheets -->

      <Stylesheet location="Samples\Common\SampleStylesheet.xml">
        <UseStyle>CalibriHighlight</UseStyle>
      </Stylesheet>
      <C rc="R29C4" style="Header">Gender</C>
    …hidden xml…
```

USAGE WITHIN SAS

SCP can be used in a number of ways within SAS. Anything that can create a text file can be called to create the XML file.

The data step, using simple put statements, is an easy method to demonstrate:

```
filename outFile 'c:\temp\SimpleSample.xml' ;

data _null_ ;
   set sashelp.class end=eof;
   file outFile lrecl=1000;
   retain row 7 ;
   if _n_ = 1 then
      do;
        put
           / '<Actions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                       xsi:noNamespaceSchemaLocation="..\..\XML\SaviCellsMain.xsd"
                       xmlns:xi="http://www.w3.org/2003/XInclude"
                       log="c:\temp\SaviCells.log" appendLog="false">'
           / '   <Export>'
           / '      <Workbook excelFile="c:\temp\simple" excelExtension="xls">'
           / '          <Worksheet name="Class">'
           / '             <!-- RAW DATA INSERTION AT CELL LEVEL -->'
           / '              <Cell rc="R' row +(-1) 'C4">Name</Cell>'
           / '              <Cell rc="R' row +(-1) 'C5">Gender</Cell>'
           / '              <Cell rc="R' row +(-1) 'C6">Age</Cell>'
           / '              <Cell rc="R' row +(-1) 'C7">Height</Cell>'
           / '              <Cell rc="R' row +(-1) 'C8">Weight</Cell>'
           / ' '
           / '          <!-- RowFill can be used to fill in a row with data.
                            Default delimiter is a ; if dlm is not specified --> '
           / ' '
          ;
      end;

   row = row + 1;
   put
        '        <RowFill rc="R' row +(-1) 'C4">' name +(-1) ';' sex +(-1) ';' age +(-1) ';'
                          height +(-1) ';' weight +(-1)'</RowFill>'
        ;

   if eof then
        do;
         put
           / '           <!-- Cell can be abbreviated as C -->'
           //
           / '           <C rc="R27C7">Total</C>'
           / '           <C rc="R27C8">=SUM(R[-19]C:R[-1]C)</C>'
           / '       </Worksheet>'
           / '     </Workbook>'
           / '   </Export>'
           / '</Actions>'
           ;
      end;
run;

options noxwait noxsync;

x '"c:\program files\savian\savicellspro\scp.exe" c:\temp\SimpleSample.xml';
```

UTILITIES

SCP comes with several utilities to make it easier to use.

- **SCP XML Creator**

  Automatically 'discovers' the elements within an Excel workbook and creates the necessary XML for the user that describes that workbook. **This utility should only be used 1 time to create the initial framework**.

  It can also automatically generate the SAS code needed to recreate the selected workbook.

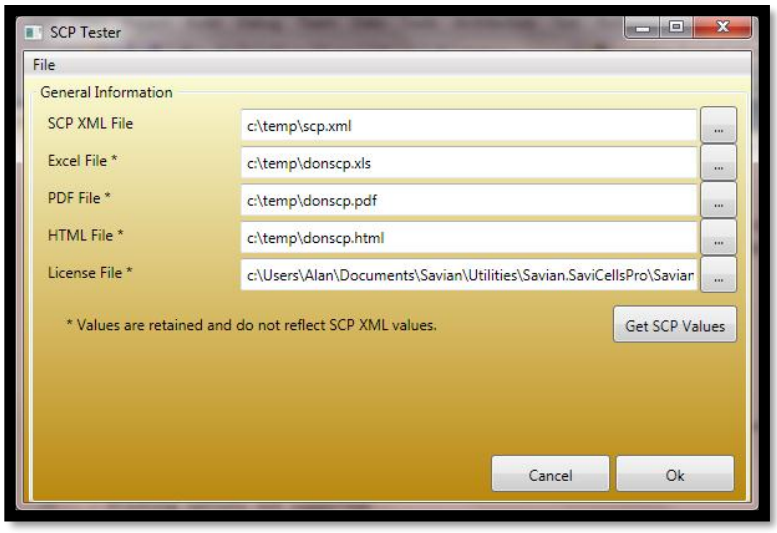

Figure 1 - SCP XML Creator Main Screen

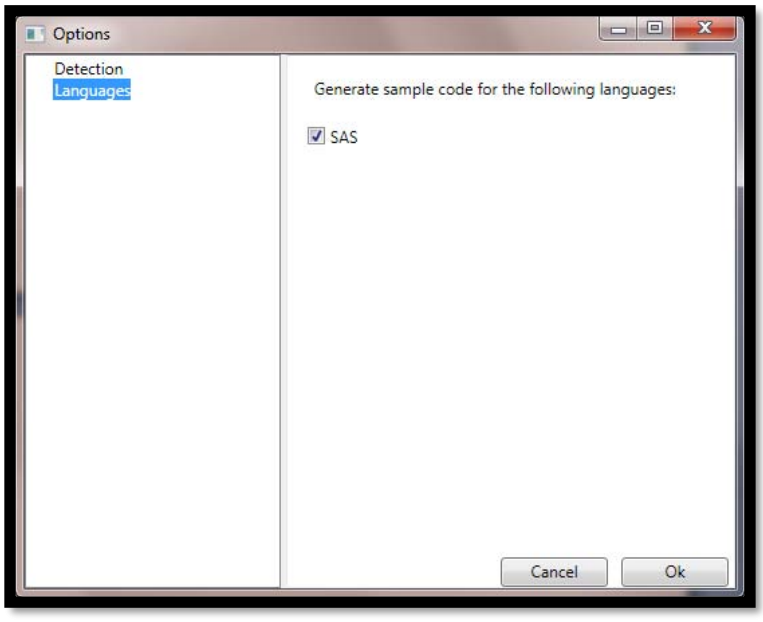

Figure 2 - SCP XML Creator - Option Screen showing SAS generation

- **SCP Tester**

  Provides a UI that allows for easier testing of SCP than using the command line.

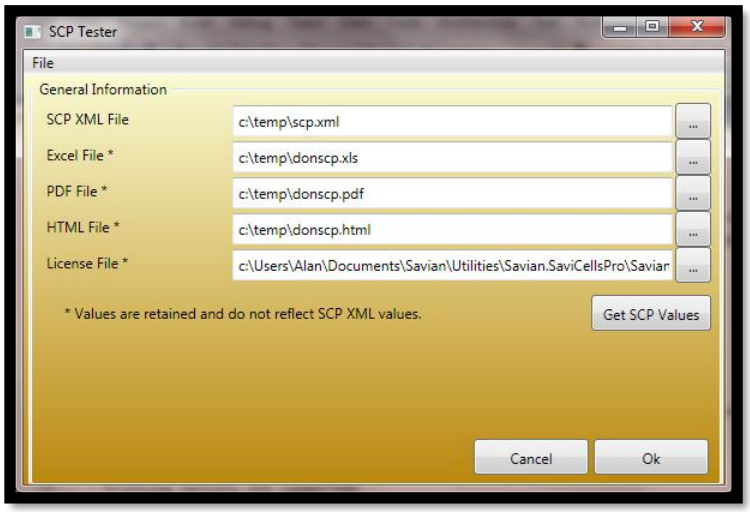

Figure 3 - SCP Tester screen

HELP FILES AND SAMPLES

SCP comes with a full help file and samples collection. Please look under the Samples directory for complete code samples including SAS code.

CONCLUSION

When approaching Excel, SAS programmers should view the problem holistically and consider directions outside of the standard approaches. In this paper, an approach was used that allowed SAS to generate Excel workbooks without using Excel and the approach bypassed the limitations found in ODS, DDE, and proc export. The SCP approach allows for rapid duplication of existing spreadsheets, simplifies the SCP XML creation through the use of various tools, and makes it easy to get a production process ready.

REFERENCES

- SCP online documentation, quick starts, and sample code can be found here:

  http://www.sascommunity.org/wiki/savicellspro

ACKNOWLEDGMENTS

Savian, LLC would like to acknowledge the continued help and support of Don Henderson who has done a lot to make SaviCellsPro better and easier for the SAS community. I would also like to acknowledge the continued support of SAS R&D, specifically Vince DelGobbo, as well as SAS Professional Services.

RECOMMENDED READING

- XML schema help
  - http://www.w3schools.com/schema/default.asp
- XML documentation
  - http://www.w3schools.com/xml/default.asp

CONTACT INFORMATION

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

Name:	Alan Churchill
Enterprise:	Savian, LLC
Address:	68 W Cheyenne Mtn Blvd
City, State ZIP:	Colorado Springs, CO 80906
Work Phone:	719-687-5954
Fax:	N/A
E-mail:	alan.churchill@savian.net
Web:	http://www.savian.net