Presentation Quality Spreadsheets From RPG

Using POI or CGIDEV2 to Build Beautiful Spreadsheets

By Martin Cytrynbaum

The iSeries is a great platform for data processing, but it is not best suited for preparing presentation quality business documents. This is why many iSeries shops will typically export selected extracts of their iSeries data to PC based spreadsheet programs like Microsoft Excel for preparing presentation documents. In this article I will outline some commonly used methods of importing data into Excel and propose some new more powerful techniques that permit better formatting of ownloaded data. Here are four common ways of moving iSeries data to Microsoft Excel:

- Use IBM iSeries Access (Client Access) to download an iSeries file (or results of an SQL statement) to a CSV (comma separated variable) file on your PC or network drive that can be easily opened by Excel. This can be done either interactively or part of a batch job.
- Import an iSeries file (or results of an SQL statement) directly into Excel using the Client Access add-in for Excel.
- Using the Copy to Import File (CPYTOIMPF) command to copy an iSeries file to a CSV file on the IFS, making certain to specify appropriate record, string and field delimiters.
- Use a utility to capture and parse spooled file data and save it in an Excel compatible file.

These four methods are good for extracting raw data, but they offer few additional ways to process or format the data. Data typically arrives in Excel with a default font and column width and without headers, as per the example in **Figure 1**.

If a download is repeated regularly and the data is used as part of a presentation document, then why should the user be forced to adjust fonts, formatting, width, height, colours and add headers and subtotals each time? Why not let the iSeries do this processing for them? In addition to being time consuming, human intervention in formatting a spreadsheet may bring along with it the risk of accidental modification of the data.

Microsoft Excel - SalesReport20.csv

File Edit View Insert Format Tools Data W

	A	B	C	D	E
1	7633	John Russ	5	2124.34	
2	7652	William W	3	4949.44	
3	7654	Edward Ba	4	6262.64	
4	7678	Spencer C	1	1234.34	
5	7679	Samuel Fc	2	3455.34	
6					

Figure 1. A typical CSV file when imported into Excel.

```
// Create 50 cells (0-49) (the += 2 becomes apparent later).
            for (short cellnum = (short)0; cellnum < 50; cellnum += 2) {
                // Create a numeric cell.
                cell = row.createCell(cellnum);
                cell.setCellType(HSSFCell.CELL_TYPE_NUMERIC);
                // Do some goofy math to demonstrate decimals.
                cell.setCellValue(rownum * 10000 + cellnum   + ((( double ) rownum / 1000)
                        + (( double ) cellnum / 10000)));
                // Create a string cell (see why += 2 in the for loop (2 cells per loop).
                cell = row.createCell((short)(cellnum+1));
                cell.setCellType(HSSFCell.CELL_TYPE_STRING);
                // Set the cell's string value to "TEST".
                cell.setCellValue("TEST");
                // Make this column a bit wider.
                sheet.setColumnWidth((short)(cellnum+1), (short)((50*8) / ((double)1/20)) );
```

Figure 2. Extract of RPG/POI code that generates a native .xls Excel file

Writing to Excel .xls Files Using POI and RPG

If you are java savvy, then there is an open source API collection from Apache called POI that may interest you. (In case you were wondering, POI stands for "Poor Obfuscation Implementation" which is a humorous jab at Microsoft's complex file formats.) POI is part of the Apache Jakarta Project and provides pure Java libraries for reading and writing files in Microsoft Office formats including Excel.

If you are not ready to tackle this project in java, you can use the RPG *object* data type introduced in V5R1 and use POI classes and methods within your RPG program. (See **Figure 2** for an example of RPG/POI code.) To generate Excel files with POI, you first have to download the POI jar from Apache and install it in your home directory. You also have to run the Java Virtual Machine (JVM) on the iSeries. POI code is very powerful but tends to be quite verbose.

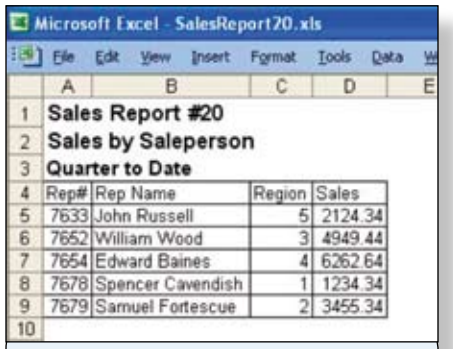

Figure 3. A sample formatted Excel spreadsheet that may have been created using POI, stream file API's or CGIDEV2

Writing to Excel .xls Files Using iSeries API

Another option is to use native iSeries APIs to create an Excel compatible file. Although there is no iSeries API to write directly to native Microsoft file formats, there is a very effective work-around.

It is commonly known that recent versions of Microsoft Excel can read from, and write to the popular Hypertext Markup Language (HTML) and Extensible Markup Language (XML). What is not well known is that if you save an HTML file with the Excel .xls extension, Excel will read in the HTML data and do its best to convert the contents to an Excel spreadsheet complete with cell width and most formatting. Similarly if you create an XML file with an .xls extension, using appropriate Excel XML tags, the resulting file should read into Excel as a regular Excel file.

The iSeries QHFWRTSF (Write to Stream File) API will permit you to write to a stream file to the iSeries IFS. By using the QHFWRTSF API in conjunction with the open and close stream file APIs (QHFOPNSF and QHFCLOSF), you can use an RPG program to create a file with XML or HTML contents and an .xls extension on the IFS from where it may be opened by Microsoft Excel as an Excel spreadsheet. (See **Figure 3**.)

```
c* define an HTML template file
c                   eval      IfsMultIndicators = gethtmlifsmult(
c                             '/CgidevExt/SalesReport20Template.html':
c                             '<AS400>')
 * Clear the HTML buffer
c                   callp     ClrHtmlBuffer

 * Open data file, read file and write sections of HTML.
c                   Callp     wrtsection('top')
c                   Open      SALES20PF
c                   Callp     wrtsection('Header1')
c                   Callp     wrtsection('TopOfTable')
c     *loval        Setll     SLSFILERC
c                   Read      SLSFILERC
c                   DoW       not %EOF
c                   Callp     updHTMLvar('Rep#'    :SLSM7)
c                   Callp     updHTMLvar('Name'    :SLMN7)
c                   Callp     updHTMLvar('Region'  :REG07)
c                   Callp     updHTMLvar('Sales'   :har(SLS07))
c                   Callp     wrtsection('TableDetail')
c                   Read      SLSFILERC
c                   EndDo
c                   callp     wrtsection('BottomOfTable')
c                   Close     SALES20PF
c*  write as a stream file on the IFS and return
c                   callp     wrtsection('endhtml')
c                   Eval      Stmf= '/CgiDevExt/SalesReport20.xls'
c                   Eval      rc = WrtHtmlToStmf(%trimr(Stmf):CodePage)
c                   return
```

Figure 4. Extract of RPG/CGIDEV2 code that generates an HTML file saved as an .xls Excel file

Although XML files are typically much larger than HTML files and initially harder to work with, Microsoft has announced that the next version of Microsoft Office (Office 12) will start using XML as the default format for saving Excel and other Office documents. XML also permits greater precision in cell formatting and additional functionality such as multiple worksheets. To get a quick introduction to the XML format used by Excel, try saving an Excel file in XML format, and open the resulting document in a text editor.

Writing to .xls Files Using CGIDEV2

If you feel that dealing with stream file APIs in RPG is a bit too technical for your comfort level, you can use CGIDEV2, the popular and free IBM iSeries web toolkit which provides a simplified interface to these APIs. (See **Figures 4**, **5**, and **6**.) Although best known for generating interactive HTML web pages, CGIDEV2 is equally useful in helping you stream an HTML or XML document to a file in your IFS using the CGIDEV2 WrtHtmlToStmf procedure.

```
<AS400>top
        <html>
        <body>
        <h2> Sales Report #20</h2>
        <h3> Sales by Saleperson</h3>
        <h4> Quarter to Date</h4>
<AS400>TopOfTable
        <table border=1>
        <tr>
        <td> Rep# </td>
        <td> Rep Name </td>
        <td> Region </td>
        <td> Sales </td>
        </tr>
<AS400>TableDetail
        <tr>
        <td> /%Rep#%/ </td>
        <td> /%Name%/ </td>
        <td> /%Region%/ </td>
        <td> /%Sales%/ </td>
        </tr>
<AS400>BottomOfTable
        </table>
        <AS400>endhtml
        </BODY>
        </HTML>
```

Figure 5. HTML template file generated by CGIDEV2 (SalesReport20Template.html)

```
<AS400>TableDetail
  <Row>
  <Cell><Data ss:Type="Number">/%Rep#%/</Data></Cell>
  <Cell><Data ss:Type="String">/%Name%/</Data></Cell>
  <Cell><Data ss:Type="Number">/%Region%/</Data></Cell>
  <Cell><Data ss:Type="Number">/%Sales%/</Data></Cell>
  </Row>
```

Figure 6. A fragment of the same template using XML instead of HTML

Whichever method you may choose to create formatted spreadsheets from your RPG programs, the results will be equally impressive and the time savings for the end-users, improved accuracy and standardization of documents will likely quickly pay back for your investment in time and effort. TUG

Martin Cytrynbaum *is an analyst/programmer on midrange systems with Liberty and Associates, a Montreal based IBM Business Partner specializing in modernizing legacy applications. He can be reached at mcytrynbaum@liberty-i.net.*

Some Links of Interest:

The Apache POI Home Page >	http://poi.sourceforge.net
Example of using POI with RPG >	http://www.foundation.be/articles_001.htm
CGIDEV2 Home Page >	http://www.easy400.net
Article on writing HTML to a stream file >	http://www.easy400.net/cgidev2o/exhibiu7.htm
More information on Martin's Web site >	http://as400.liberty-i.net/CgiDevExt/index.html