Mining the Depths of Excel
Case Study in Objects and Arrays
Why Excel?

James Kwak: (prominent financial blogger?)
"...Microsoft Excel is one of the greatest, most powerful, most important software applications of all time..."

"...Excel is everywhere you look in the business world—especially in areas where people are adding up numbers a lot, ... I have a probably untestable hypothesis that, were you to come up with some measure of units of software output, Excel would be the most-used program in the business world..."

(http://www.economonitor.com/blog/2013/02/the-importance-of-excel/)
Excel at BCA Research...

- Lists - keeping track of..., eg. user profiles, data retrieval codes, publication files, etc.
- Data sources - (Bloomberg, ThomsonReuters...) make data available as .xlsx or .csv
- Interfaces for data collection - downloads and analytical tools are driven by Excel Add-ins
- Charts - if no other way to produce
- Statistics - if no better way to calculate
- Reports - presentation of analyses; lists of things that need attention, etc.
- Process control - "table-driven" tasks - determine what to do based on worksheet contents

In short, Excel is extremely important, and EVERYWHERE
Integration with APL

NEED for SPEED?
Automation?

Dyalog’15 - Mining Excel
Richard Procter
Glitches & Gotchas

- Installation / Incompatibility issues - APL/Excel
- Resource exhaustion - Excel
- Automation failures, eg. pop-ups

Excel cannot complete this task with available resources error, Excel 2010
Excel and XML

Office Open XML (OOXML)
en.wikipedia.org/wiki/Office_Open_XML

- around 2000 - MS began to develop xml-based format for Office documents
- 2000-2006 - standardization process (ECMA/ISO/IEC)
- MS Office 2007 - MS adopts the format as default
- *.xlsx, not *.xls

Office Open XML (also informally known as OOXML or OpenXML) is a zipped, XML-based file format developed by Microsoft for representing spreadsheets, charts, presentations and word processing documents. The format was initially standardised by Ecma (as ECMA-376) and, in later versions, by ISO and IEC (as ISO/IEC 29500). Starting with Microsoft Office 2007...
Support for Office Open XML (non-MS)

and
en.wikipedia.org/wiki/Comparison_of_Office_Open_XML_software#Spreadsheet_documents

- Many direct products
- Other (C#/VB) programming components
Why not COM? (OLE, ActiveX...)

(http://ramblingcookiemonster.github.io/PSExcel-Intro/)

"Chances are you have worked with Excel through COM....this isn’t supported if you want to use it in an automated solution..."

...from Microsoft: (https://support.microsoft.com/en-us/kb/257757)
All current versions of Microsoft Office were designed, tested, and configured to run as end-user products on a client workstation. They assume an interactive desktop and user profile. They do not provide the level of reentrancy or security that is necessary to meet the needs of server-side components that are designed to run unattended.

"Microsoft does not currently recommend, and does not support, Automation of Microsoft Office applications from any unattended, non-interactive client application or component (including ASP, ASP.NET, DCOM, and NT Services), because Office may exhibit unstable behavior and/or deadlock when Office is run in this environment."

So #1 = Automation; Plus - above-mentioned issues...
Options beyond COM/OLE

Application?; or API?
- Who does the UI?
- Is there a UI?; etc.
- (there's also ODBC?; ADO.Net?)

Applications
- www.openoffice.org - "Calc", formerly by Sun Microsystems, now Apache
- Google Docs; Android Apps - work but are features limited?
- (MS-OneDrive = Excel Online?)
API / Utilities

Syncfusion XlsIO

Introduction to Essential XlsIO

Essential XlsIO is a 100% native .NET library that generates fully functional Microsoft Excel Spreadsheets in native Excel format without depending on Microsoft Excel. Essential XlsIO is a perfect solution for those who need to read and write Microsoft Excel files from code. It does not require MS Excel to be installed in the report generation machine or server.

Essential XlsIO library can be used in any .NET environment including C#, VB.NET, and managed C++. It is a Non-UI component that can be used in ASP.NET, Windows Forms, WPF, Silverlight, ASP.NET MVC, WinRT, Windows Phone 8 applications, without any change in the API.
sfExcel

- excellent toolkit for Dyalog APL (no need to study XlsIO)
- Namespace
- speed issues if (1000's < 1↑MAT)
- "DataTable" feature for speedups
- some formatting options
- data/text only (not "inclusions")
- Pierre Gilbert
But what does the C# / VB / VisualStudio crowd do?

MS Open XML SDK

Welcome to the Open XML SDK 2.5 for Office

This content set provides documentation and guidance for the strongly-typed classes in the Open XML SDK 2.5 for Office.

Last modified: March 10, 2015

Applies to: Office 2013 | Open XML

The SDK is built on the System.IO.Packaging API and provides strongly-typed classes to manipulate documents that adhere to the Office Open XML File Formats Specification. The Office Open XML File Formats specification is an open, international, ECMA-376, Second Edition and ISO/IEC 29500 standard. The Open XML file formats are useful for developers because they are an open standard and are based on well-known technologies: ZIP and XML.
Open XML SDK - Download

API and "Tool"

Open XML SDK 2.5 for Microsoft Office

Language: English

Download

Note: There are multiple files available for this download. Once you click on the "Download" button, you will be prompted to select the files you need.

Version: RTW
File Name:
OpenXMLSDKV25.msi
OpenXMLSDKToolV25.msi

Date Published: 11/20/2012
File Size:
2.5 MB
24.9 MB
Open XML SDK - Tool

- drill-down into XML structure
- structure, not content?

file:///C:/Program Files (x86)/Open XML SDK/V2.5/tool/HelpPage.htm

"...The tool integrates the following functionalities: Automatically generate Open XML SDK code based on document content. User will be able to directly copy, compile and run the code to re-generate the same document or specific parts of the documents..." (? not tried yet)
Open XML - Structure


See also: "Open XML Explained e-book"
Open XML - Structure (from APL)

:USING System.IO.Compression (zip.dyalog by DanB)
Open XML SDK - .Net Library

```
USING,←⊂'C:\Program Files\Open XML SDK\V2.5\lib\DocumentFormat.OpenXml.dll'
```
Open XML SDK (or similar) - Product Integration

numerous products built with this, or in similar fashion to it by making use of the Open Office Xml format, some free, some not, some out-of-date, eg.

- EPPLus - epplus.codeplex.com - uses System.IO.Packaging namespace
- github.com/dfinke/ImportExcel - uses EPPLus.dll
- GemBoxSoftware - www.gemboxsoftware.com/support/articles/read-write-excel-spreadsheet-net
- EASYXLS - www.easyxls.com/
- ExcelPackage - excelpackage.codeplex.com - (last update 2007?)


- popular Blog: 
  http://openxmldeveloper.org/
**Issues** (XlsIO and Open XML SDK)

1. Speed - for sfExcel (XlsIO) - using a "DataTable" may help, **BUT...** what about datatype?

**DataSet & DataTable** (...represents one table of in-memory relational data):
https://msdn.microsoft.com/en-us/library/t31h6yhs%28v=vs.110%29.aspx
XlsIO - DataTable usage

http://docs.syncfusion.com/winrt/xlsio/working-with-data#exporting-from-worksheet-to-data-table

Exporting from Worksheet to DataTable

It is easy to export the sheet data to a data table by using the ExportDataTable method of IWorksheet. This method allows selection of various data table options such as include column names, export formula calculated values, styles, and types through the ExcelExportDataTableOptions enumeration. It has the following values.

<table>
<thead>
<tr>
<th>Members</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Indicates default export to datatable.</td>
</tr>
<tr>
<td>ColumnNames</td>
<td>Indicates to export ColumnNames to datatable.</td>
</tr>
<tr>
<td>ComputedFormulaValues</td>
<td>Indicates to export ComputedFormulaValues to datatable.</td>
</tr>
<tr>
<td>DetectColumnTypes</td>
<td>Indicates that XlsIO should try to detect column types.</td>
</tr>
<tr>
<td>DefaultStyleColumnTypes</td>
<td>When DetectColumnTypes is set and this flag is set too, it means that the default column style must be used to detect style. If this flag is not set, but DetectColumnTypes is set, then the first cell in the column is used to detect column type.</td>
</tr>
</tbody>
</table>
DataTable I-Beams - 2010منذ، 2011منذ

Dyalog Technical Note #1: (circa 2010)

SetDT←2010منذ Set DataTable contents
GetDT←2011منذ Get DataTable contents

"Optimisations to Support the Microsoft.Net System.Data.DataTable Class"

System.Data.DataTable is a central object in the ADO.NET library in the Microsoft.Net Framework. It is used by DataSet and DataView objects as a container for data which has been extracted from, or will be written to, an ADO.NET data source.

"...because we need to “loop” on each row – doing a noticeable amount of work each time. The new I-Beam does all the looping in compiled code..."

- Rapid conversion of source data to APL arrays
- compromises on datatype, easily overcome in APL
- sfExcel fns: GetNumber2, GetText2, SetText2...
- dt.Select(query) = select rows "on the way in" (to APL)?
Open XML SDK - Issues

• not many, really, other than complexity, since it offer lots of options

• *guessing*) similar performance issues as above for large tables *(when used in "classic OO" manner)*

• finding online examples is easy (C#), but adapting such code is challenging (for me)

• for really large files, blogs indicate workarounds using "Streaming" methods (?)

• is it really necessary? ie. if we are just pulling out entire worksheet xml vectors...

• may be helpful on inclusions (charts, etc.) - to be investigated; or just sorting out what the workbook contains if unknown?

• work in progress
Using Open XML with APL - what is the best strategy?

since we "have" the data...

\[ \text{xml} \leftarrow \{ \text{Excel/OpenXML worksheet object} \} \ldots \text{Worksheet.OuterXml} \]

(159478)

(2000↑xml)

(\text{thanks DanB...})
Open XML → APL

extract items from the "XML vector"... insert "shared strings"...

The row element defines a new row. Normally the first row in the sheetData is the first row in the visible sheet. There are optimizations that can alter this behavior, which is discussed later on in the chapter. Inside the row you create new cells using the c element. Values for cells can be provided by storing a v element inside the cell. Usually the v element will contain the current value of the worksheet cell. To store a numeric value in the spreadsheet, all you have to do is to include its value in the v element.

```
<worksheet xmlns="http://.../spreadsheetml/2006/main">
  <sheetData>
    <row>
      <c>
        <v>42</v>
      </c>
    </row>
  </sheetData>
</worksheet>
```

Open XML → APL - issues and further work

- XML vs. direct string manipulations? regular expressions?
- & etc. ie. xml escapes
- missing cell references, special-case dates, N/As, etc.
- inclusions (charts, shapes, tables, etc.)
- selection on (named?) ranges, worksheets, etc.
- clever coding for DataTable and datatype issues (several options)
- APL write to Excel - more to be investigated
Thank you!