Thoughts about OLE Automation & Flexcel

Introduction: “Friends don't let friends use OLE Automation”

Before starting I would like to warn that this article is mostly about my personal views and opinions. While all articles are subjective and influenced by what the writer is thinking, it is easier to write a purely technical article where you explain how formulas are updated when you insert a row, than an article where technical facts are mixed with your own opinions and ideas. I took a large amount of time checking all facts are correct as far as I know, and checking than when I state my opinion I also state possible alternative opinions too. This made the article quite large, but I hope that still worth a read.

I am going to analyze issues, conceptions and misconceptions about creating or reading Excel files from your own applications. All I will be writing here is the result of my own experience as a former OLE Automation user myself, but more important, the result of more than five years hearing horror stories from other former OLE users all around the world who became our customers after losing a lot of time and resources trying to make the thing work. Even so, most of what I will write in this article is not related specifically to our third party solution; I am covering general ideas not related to the third party provider (if any) you choose.

So let's go. This all started with the article you can find here:

http://www.joelonsoftware.com/items/2008/02/19.html

Joel is one author that I respect a lot, and one of the few people I regularly read, and probably that is one of the reasons that the article linked above hit a nerve when I read it. My initial reaction was something like “What??” and can only be completely explained here, but it has reduced now a little so I hope can write this in a more calm and impartial way.

Analyzing Joel's article

First of all let me say that I actually agree with the main point in the article: Excel files are complex, and this is not because some dark evil plan, but just because Excel itself is complex and this must be reflected in the file format. That's it. It happens with Excel, it happens with ODF, it happens with PDF, it happens with HTML/CSS. And the easier you make the format for writing, the more difficult you make it for rendering it. Have you tried lately to create your own PDF or HTML viewer? Complex things are complex, and if you add the need to backwards compatibility you end up in a file definition that will be no better than xls.

So let's skip to the part that got me so upset:

OK, I promised some workarounds. The good news is that for almost all common applications, trying to read or write the Office binary file formats is the wrong decision. There are two major alternatives you should seriously consider: letting Office do the work, or using file formats that are easier to write.

There is nothing actually wrong in that sentence; in fact, the problem is not what it says but what it doesn't. Yes writing the office files directly is not normally a good idea. Yes you should consider those two alternatives. But what isn't mentioned here is that there is other big alternative you should be considering, and I know from my job many people are unaware of.

There is a quote I still remember more than ten years after hearing it. It was from a professor in one of our last courses before graduation, and it was something like this:

"I know what many of you are thinking. When you graduate, first thing you will do is go to the beach and grab some sand. From the sand, you are going to extract silicon. With it, make some transistors that you will use to assembly your own computer. You will then write your own drivers, your own operating system, and finally your application. I have bad news for you: real life is not like that"

I couldn't help to remember that quote while reading the article. The solutions proposed here, the same as the solution "to get a computer start with the sand" miss the most evident real-life solution: Get it from someone else who specializes in the problem. If you need a computer and you are not a computer maker, the best you can do is go and buy a computer and concentrate on your own problem. If you need to create office files and you are not in the business of creating office files, then the best you can do is to get a solution from some people whose main objective is creating office files.

You will get a much better, tested and cheaper solution if you get it from someone else than if you start from the sand. (Even when stating from the sand might be more rewarding intellectually speaking)

I will admit the reason I still remember that sand thing is because I was one of those guys who was making a list with the best beaches to go get sand after graduation. And in those years after graduating, I have seen myself and lots of other people falling once and again in the NIH trap, because it has that intellectually rewarding factor.

So let's dig a little deeper.

The proposed alternatives

1) Let Office do the work for you

Most of what I can say here is actually mentioned in Joel's article itself, but downplayed and mentioned as some "little gotchas". Just go to http://support.microsoft.com/default.aspx?scid=kb;EN-US;257757 and make sure you give it a thoughtful read. Really. Don't worry, I'll wait.

For starters, a suspicious mind might see something wrong in the fact that Microsoft itself makes a statement like this over its own product:

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, DCOM, and NT Services), because Office may exhibit unstable behavior and/or deadlock when run in this environment.

Yes it can be just that they are trying to scare us, but what if there is something wrong after all? Maybe Microsoft has a reason to publish (and regularly update) that article? I will tell you what; I believe the reason that article is there is because Server Side Ole Automation does not actually work. There, I said it. After a customer has lost thousands of dollars and time and effort trying to implement such solution, they can be directed to that page where they will learn the thing wasn't supposed to work from the beginning. And that even if they actually managed to make it work, they didn't have the legal rights anyway:

Besides the technical problems, you must also consider the feasibility of such a design with respect to licensing. Current licensing guidelines prevent Office Applications from being used on a server to service client requests, unless those clients themselves have licensed copies of Office. Using server-side Automation to

provide Office functionality to unlicensed workstations is not covered by the End User License Agreement (EULA).

And this is my main gripe with Joel's article.

You cannot expect that every user will find this somehow hidden msdn article and be aware of those "little gotchas" before implementing an Ole Automation solution. And an article like Joel's is not making people any favors. When someone gets to point 3 on the implementation:

3. Same as above, but you need to scale. Throw a load balancer in front of any number of boxes that you built in step 2. No code required.

He will find out that the thing doesn't actually scale, not as it should. No matter how much boxes (and dollars) he throws at it. And he will find that Microsoft had warned him on it, especially on the Reentrancy and Scalability section of that article he didn't found or choose to ignore.

2) Use a simpler format

Different from the previous alternative, this is actually a valid option, and actually used by companies like Google or Amazon to deliver reports, but it is not without its own gotchas too.

a) Use CSV.
While the obvious issue with CSV is that it doesn't support any formatting, or images, or charts, or well, anything but raw data, there is a more subtle issue too. Many countries (I would say most non English ones, but I might be wrong) use a "," and not a "." as decimal separator. This means they can't use "," as field separator, and they use something else instead. (We use ";" in Spanish). And this means that when I open my Amazon s3 reports I get all my data in the first column as a big string. Not nice, let me tell you. I personally was so annoyed with it that created a small application using FlexCel to convert those reports into nice formatted ones, but I don't think normal users should be expected to do that. So if you need to go csv, consider going tsv instead. Tab separated spreadsheets will load better in any locale, even when not 100% perfect. If you have any floating number like "2.5" in your file a Spanish Excel will still load it wrong (it won't be converted to the number "2,5"), but at least it will be imported as a string in its own column.

b) Use WK1 or older Excel files.
I wouldn't go that way, simply because Microsoft is stopping support for those older file formats, and I can understand their reasons. So again, after you spend a couple of months and a couple thousand dollars programming your own wk1 exporter, you might find that Excel 2007 users (or Excel 2003 users with the latest service pack) are not able to read them.

c) For word, use RTF.
Well, I can't actually comment on this one because I mainly work with Excel, so as a good cobbler I will stick to my last. It might be a valid workaround, if you don't care for larger files especially when you embed images in it.

The missing alternative

3) Use a proven third party solution from a provider you trust. Whatever language you are programming in, there are probably a dozen of native solutions to create office files. And you know what? There is a reason for that. If any of the other two solutions actually worked well you would not see so many of us rewriting office again, especially when as correctly stated in the article, it is far from a simple task. I know we would be out of business very soon if solutions 1) and 2) were viable solutions for most customers.

Now let me get things straight. This third alternative is not the Holy Grail either, and it has its share of issues too. In some cases you might want to use alternative 2), and in some you might even need to <gasp> use 1). But I think 3) is an alternative you should have present when you decide what you are going to do, and should have been mentioned in the article along with the others two.

And as I told you about the drawbacks of the others solutions, it wouldn't be fair if I didn't tell you about the drawbacks of a third party solution.

So let's start with the biggest issue: Third party dependence. In fact, you always have third party dependence. Unless you are in the sand-silicon-os-application business, your work stands on the shoulders of so many giants to even try to mention. But depending on Microsoft somehow feels safer than depending on FlexCel. And I can't argue this point (even when some vb6 users could). The only thing we can do (and actually do) to mitigate the problem is to give you the full source code in all versions of FlexCel, so even if we disappear from the face of the earth tomorrow you still have the code. But I know this is not enough in many cases, so as always, the decision is yours. Is the benefit you get from a third party solution (namely nice spreadsheets with formatting that work reliably) worth the risk of your third party provider disappearing in thin air? How bad would it be for you if they actually disappeared? Only you can answer those questions.

Besides this biggest problem, there are two other problems that I see popping up from time to time, but they are not normally a big deal, and there are normally misconceptions about them too. So to round up this article I will discuss them here.

Creating complex files

Note: While most of what I wrote in this article is speaking about third party solutions in general, on this section I am going to speak specifically about FlexCel. The reason being not that other solutions don't do the same or better, but just that I don't know, and I normally prefer to speak of things I know about.

One common reason people will tell you why you can't really use a third party tool is because a third party tool will never support all the features Excel supports. Joel will tell you (and I strongly agree with him here) that people need 100% of the features, not 20%. And they are right. Kind of.

You know, for all the little love those “evil” binary file formats get, they have a very nice property that is seldom mentioned: You can modify a part of a file without invalidating other parts. In fact, you can even modify a part of a file without even knowing what you have on other parts of that file.

And that's what we do.

In our case, our approach from the beginning has been to modify existing files. FlexCel cannot even create a new blank file in its own. It can however modify an existing blank file in a lot of ways. It even has a method “NewFile” you can use to create a blank file, but behind the curtains what this method does is just to load an empty file from an embedded resource.

And what this means is that you can have a file with macros, a pivot table, some ActiveX components and even an embedded word document, open it with FlexCel, insert some rows, change cells values, and save it back. And the macro, the pivot table, the word document, everything will still be there, even when FlexCel does not understand those things.

Even better, if you inserted rows in the range where the pivot table was, now the pivot table will be updated to reflect the new data.

So you can actually create very complex files reliably, having features that are not even supported. You cannot modify those not supported features with our API (for example you cannot add a macro), but for most customers there is no need either. They just need to include an existing macro. It might not be the "perfect" solution for every customer and some people might need to go with OLE anyway, but I know we have thousands customers who are very happy with it, and I have seen spreadsheets from my customers complex besides what I imagined possible.

The myth of "100% Excel compatible"

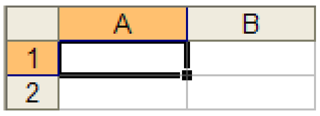
Other common saying you are likely to hear, this time when rendering existing files instead of creating them, is that "Only Excel can be 100% compatible with Excel".

At first glance it sounds logical. Any independent implementation of a complex application like Excel is going to have its own share of bugs, problems and non implemented features. And as the Excel implementation is by definition "the right one", it is the only one that is going to be 100% faithful.

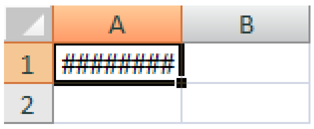
Thing is, it might actually sound logical, it might even look self-evident, but it is wrong. Not even Excel is 100% compatible with Excel. Two reasons:

1) The first reason is that there is no "one true version" of Excel. Excel 2007 will display things different from Excel 2003, which in turn will be different from Excel XP.

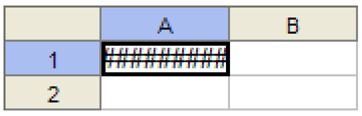
Want some examples? Try entering -1 inside a cell, formatting it as "Custom", entering "[>10]" in the format definition and pressing enter. You will see the following:



Excel 2003



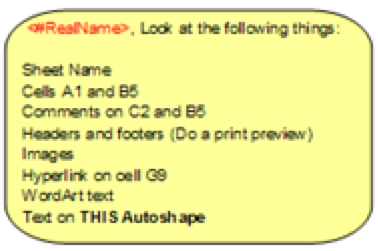
Excel 2007



FlexCel .NET Viewer

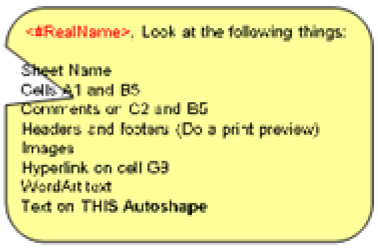
For completeness I added the FlexCel output too. In this particular case, since we couldn't display the thing as Excel 2003 and Excel 2007 at the same time, we settled for the Excel 2007 way.

You can also try to open the file "Getting started Reports.template.xls" from the FlexCel.NET "Getting Started Reports" demo. This is a normal Excel 2003 file never touched with FlexCel or any third party tool (so you can't blame the problem on us). Look at the yellow autoshape there. This is how it displays:


<#RealName>, Look at the following things:

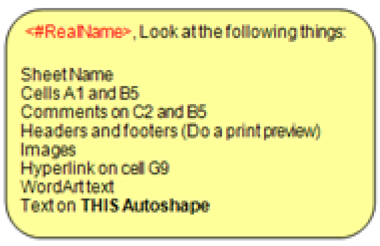
Sheet Name
Cells A1 and B5
Comments on C2 and B5
Headers and footers (Do a print preview)
Images
Hyperlink on cell G9
WordArt text
Text on THIS Autoshape

Excel 2003


<#RealName>, Look at the following things:

Sheet Name
Cells A1 and B5
Comments on C2 and B5
Headers and footers (Do a print preview)
Images
Hyperlink on cell G9
WordArt text
Text on THIS Autoshape

Excel 2007

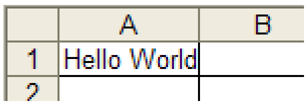

<#RealName>, Look at the following things:

Sheet Name
Cells A1 and B5
Comments on C2 and B5
Headers and footers (Do a print preview)
Images
Hyperlink on cell G9
WordArt text
Text on THIS Autoshape

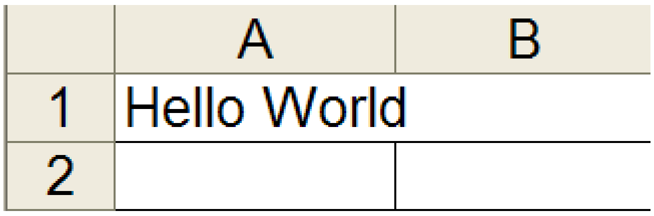
FlexCel .NET Viewer

I could spend a lot of time showing you tons of little Excel 2007 incompatibilities we find here while creating our tests, but I hope those are enough to make you an idea: **Different Excel versions render Excel files differently.**

2) And the other reason is that Excel is "resolution dependent", and will display different things with different resolutions, as you can see in the following screenshots:



File at 100% Zoom



Same file at 200% zoom

What is worse, this applies not only to the screen. If you ever change the default printer of your document all things will change. Not much, but just enough so that last row does not fit anymore in the current page and will appear alone in a blank shiny new page. **Not even the same version of Excel will always display the same.**

So why did I wrote this? Am I trying to make fun of poor old Excel by pointing and some very minor bugs? Well, yes I am, but the point I am trying to make is that even Excel can't escape the "100% faithful" problem, and it will have too its share of bugs and issues.

What actually happens is that we don't care, as long as it is good enough. As humans we are quite used to know perfection doesn't exist. So yes, FlexCel will not display things exactly and up to the pixel as Excel will do. Neither will OpenOffice. And neither will Excel. And it doesn't matter.

Other possible issues

I won't pretend I have covered all issues with third parties, there are lots of them, and some I might not even be aware of. But I think I covered the most important things. Note that I didn't even mention cost above, since the cost of a royalty-free license of FlexCel is much less than one single license of Excel. And please come back to tell me your numbers once you start adding those servers to "scale" an Ole Automation solution.

Conclusion

As promised this has been a highly subjective and personal article. And while myself, I wouldn't touch OLE automation with a 10 feet pole, I am aware that things are not so black and white. OLE has its place, especially for light use, controlled environments, client-side applications where you can guarantee everybody uses Excel and not other thing like Open Office. CSV has a big place too, but in my opinion more like a "we also offer you this" format than as the main format, unless you are generating files for machine consumption. But for machine use, normally xml is better anyway since it doesn't have all the locale problems.

There is more. As I have said earlier, I hear lots of customer histories from people who failed to make Ole Automation work (and many times lost lots of money and time and effort trying to do so). But that is because those people are the ones who actually become our customers, so I am more likely to hear the negative experiences from the people who couldn't make it work. It might be that while I hear those thousands of stories I don't hear all those millions of stories of people using Ole without issues because they are not our customers and they will not write to me. What can I say? This might be right. Me, while I cannot actually prove it, I have that gut feeling that it is not.

Think I can finally go to sleep now.

Take care,
Adrian Gallero - FlexCel Developer.