

# CASE STUDIES

## Aspose.Cells for .NET Case Study



Digital Synergy Solutions

Using Aspose.Cells to Create a Configurable Excel Reader

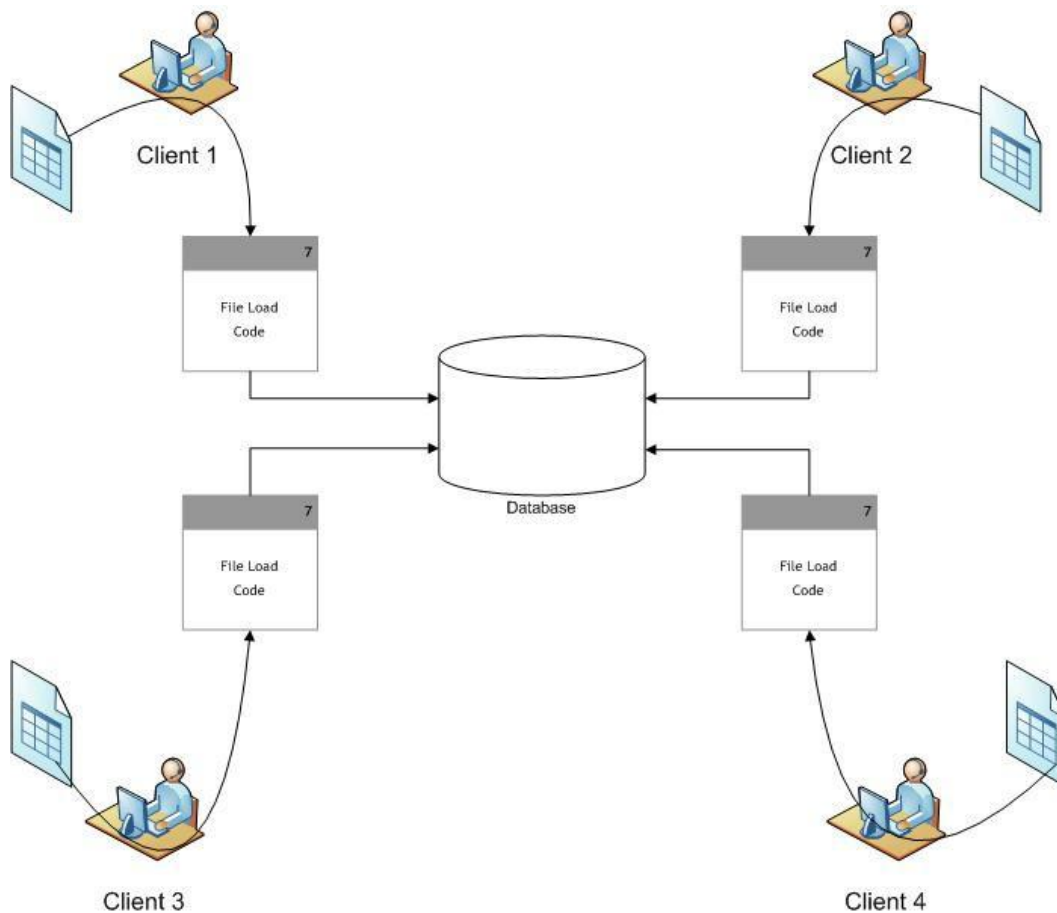
Craig Jones, Technical Consultant, February 3, 2011

## About Digital Synergy Solutions

Digital Synergy Solutions specializes in web application development and vendor software integration for clients within the financial industry. When architecting a solution for a client, we first always look to the market for “best in breed” products offering the most extensive feature set, as well as solid and consistent quality and support.

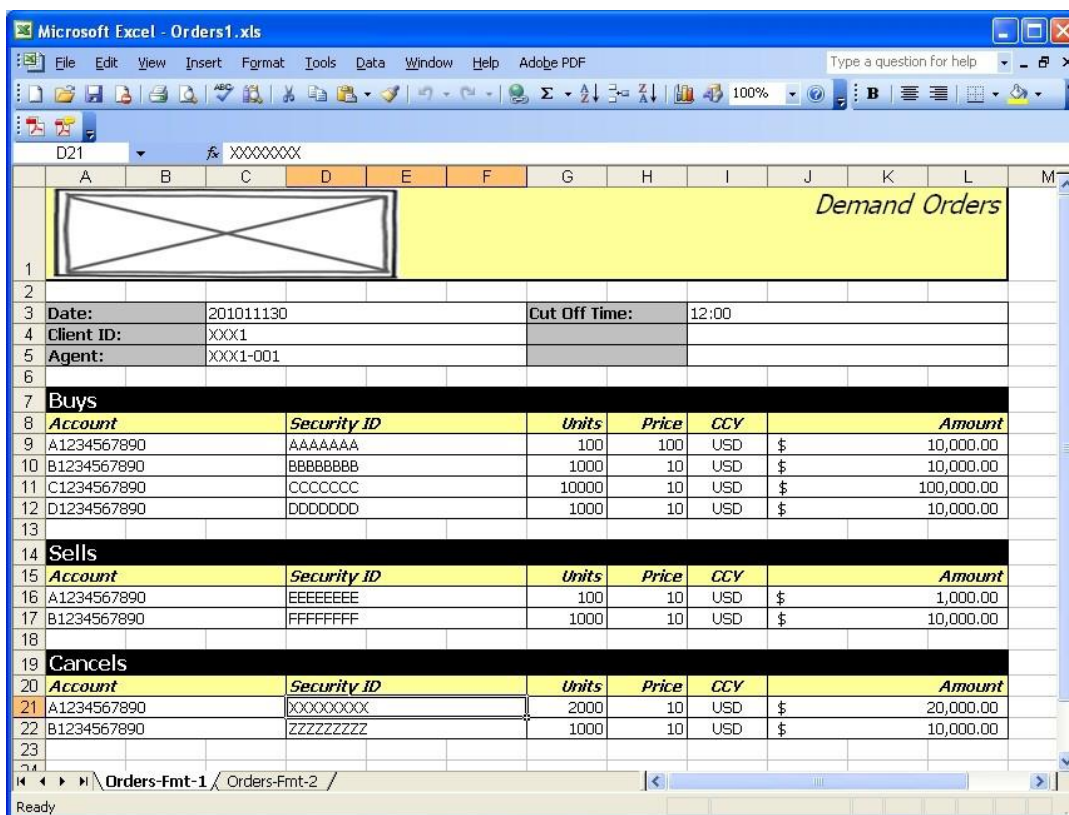
## Problem

Our client built a prototype application and eventually used it to service their clients’ needs. As they signed on new clients they built bespoke conversion code to read their clients’ Excel files by either using Excel Automation or ADO.



As more clients signed on to use the services of their application, they sought to deploy it to a production server-based environment. Shortly thereafter, they found Excel automation on the server troublesome and found ADO access difficult to navigate, especially when reading form-like formatted Excel worksheets, where data regions could start and end on any row for a certain column.

Figure 1: Example Client Format 1 – Form, where data regions start and end on any row.



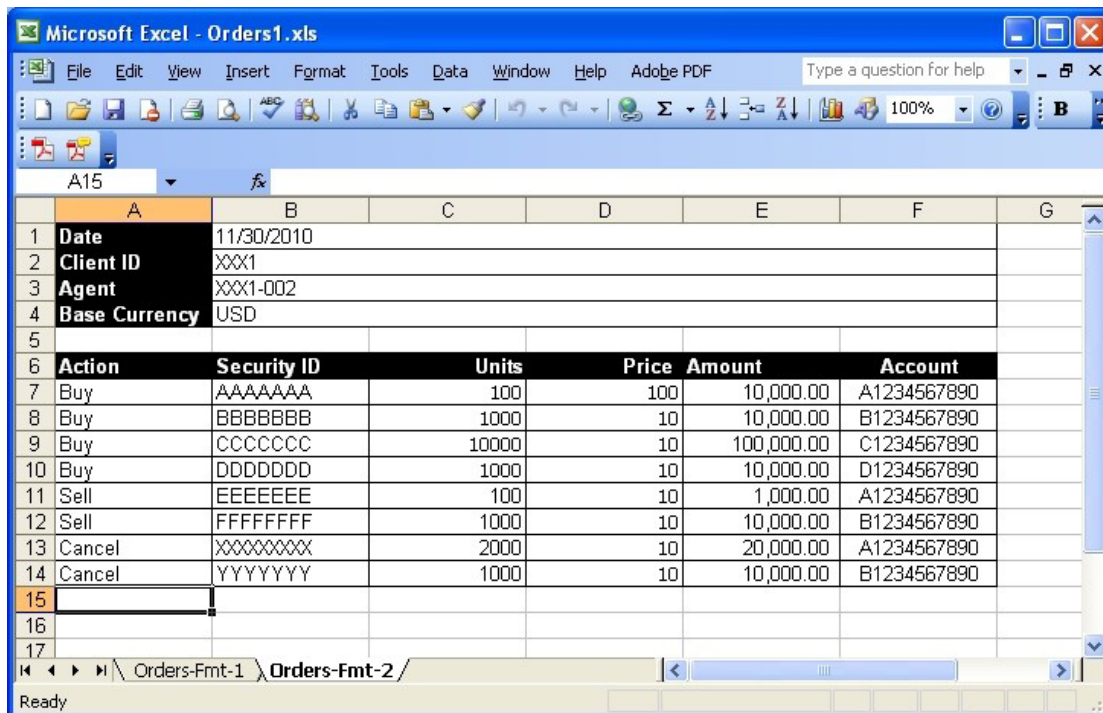
The screenshot shows an Excel worksheet titled "Orders1.xls" with a menu bar (File, Edit, View, Insert, Format, Tools, Data, Window, Help) and a toolbar. The worksheet contains a form with several sections:

- Header Section (Row 1):** A large yellow cell spanning from column D to L, labeled "Demand Orders".
- Form Fields (Rows 3-6):**
  - Row 3: Date: 201011130, Cut Off Time: 12:00
  - Row 4: Client ID: XXX1
  - Row 5: Agent: XXX1-001
- Buys Section (Rows 7-13):**
  - Row 7: Section header "Buys".
  - Row 8: Table header with columns: Account, Security ID, Units, Price, CCY, Amount.
  - Row 9: A1234567890, AAAAAAA, 100, 100, USD, \$, 10,000.00
  - Row 10: B1234567890, BBBB8888, 1000, 10, USD, \$, 10,000.00
  - Row 11: C1234567890, CCCCCC, 10000, 10, USD, \$, 100,000.00
  - Row 12: D1234567890, DDDDDDD, 1000, 10, USD, \$, 10,000.00
- Sells Section (Rows 14-18):**
  - Row 14: Section header "Sells".
  - Row 15: Table header with columns: Account, Security ID, Units, Price, CCY, Amount.
  - Row 16: A1234567890, EEEEEEEE, 100, 10, USD, \$, 1,000.00
  - Row 17: B1234567890, FFFFFFFF, 1000, 10, USD, \$, 10,000.00
- Cancels Section (Rows 19-22):**
  - Row 19: Section header "Cancels".
  - Row 20: Table header with columns: Account, Security ID, Units, Price, CCY, Amount.
  - Row 21: A1234567890, XXXXXXXX, 2000, 10, USD, \$, 20,000.00
  - Row 22: B1234567890, ZZZZZZZZ, 1000, 10, USD, \$, 10,000.00

The status bar at the bottom shows "Ready" and "Orders-Fmt-1 / Orders-Fmt-2 /".

Our client also sought the ability to on-board more of their clients with the shortest lead time as possible.

Figure 2: Example Client Format 2: Simple Form



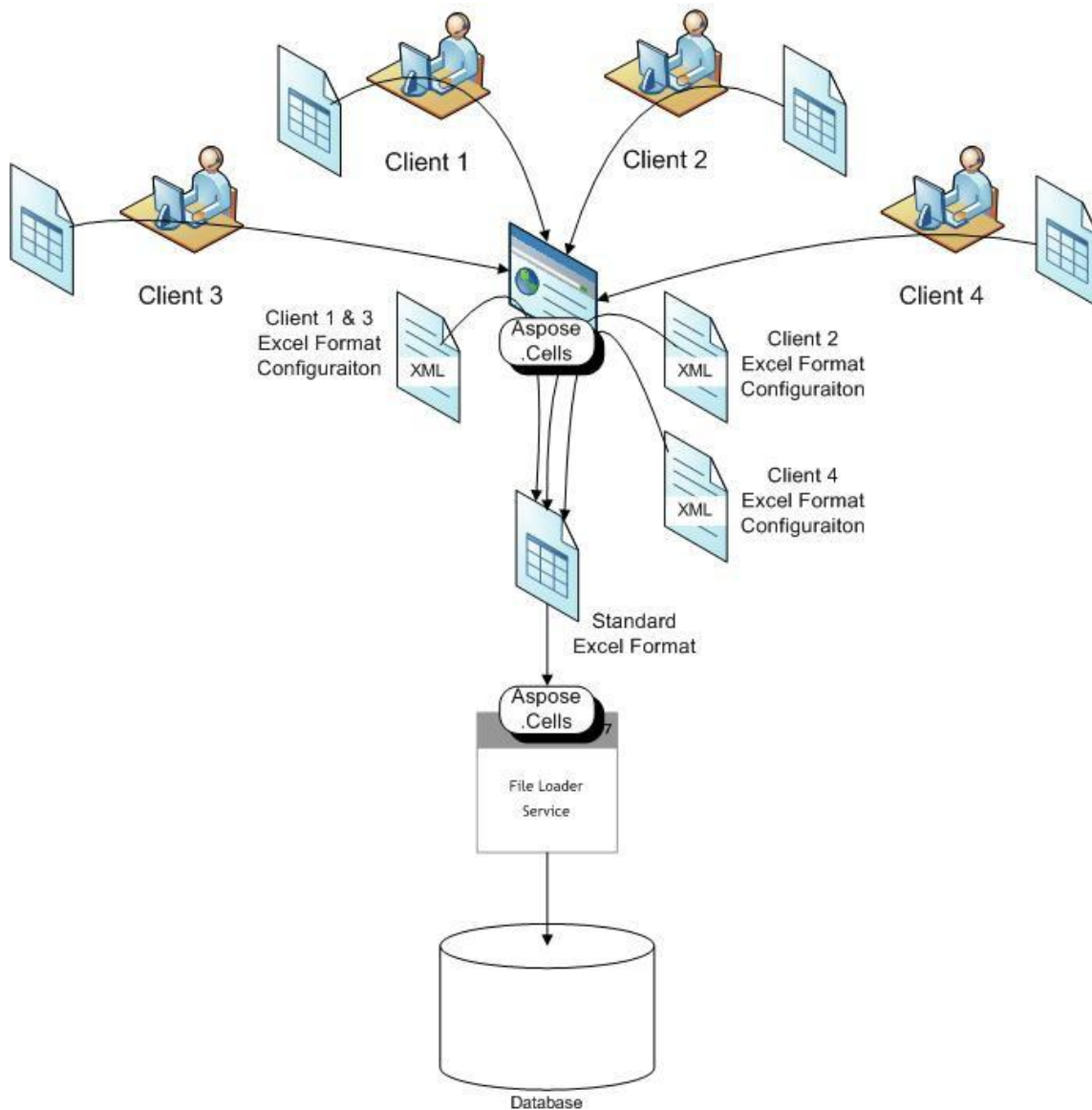
	A	B	C	D	E	F	G
1	<b>Date</b>	11/30/2010					
2	<b>Client ID</b>	XXX1					
3	<b>Agent</b>	XXX1-002					
4	<b>Base Currency</b>	USD					
5							
6	<b>Action</b>	<b>Security ID</b>	<b>Units</b>	<b>Price</b>	<b>Amount</b>	<b>Account</b>	
7	Buy	AAAAAAA	100	100	10,000.00	A1234567890	
8	Buy	BBBBBBB	1000	10	10,000.00	B1234567890	
9	Buy	CCCCCCC	10000	10	100,000.00	C1234567890	
10	Buy	DDDDDDD	1000	10	10,000.00	D1234567890	
11	Sell	EEEEEEE	100	10	1,000.00	A1234567890	
12	Sell	FFFFFFF	1000	10	10,000.00	B1234567890	
13	Cancel	XXXXXXXXX	2000	10	20,000.00	A1234567890	
14	Cancel	YYYYYYY	1000	10	10,000.00	B1234567890	
15							
16							
17							

## Solution

Our solution was to service enable the load and conversion of varying client formats into a standard format before load and processing into the application's database.

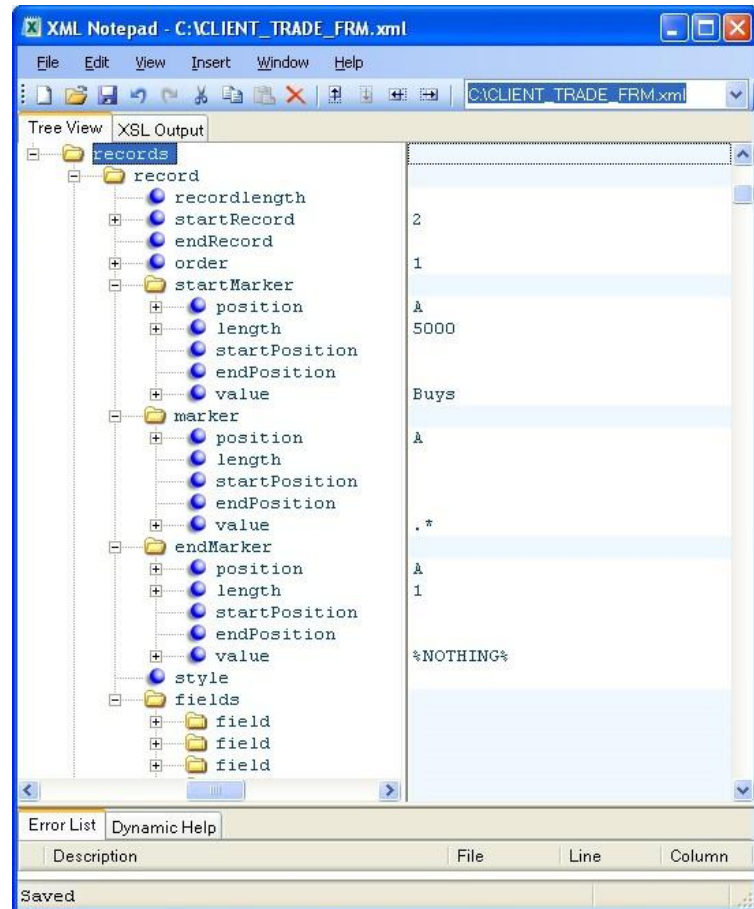
We evaluated several products on the market supporting server side Excel automation, and enabling cell level reading/writing. Aspose.Cells scored the highest for available workbook, worksheet and cell level access and feature support. With Aspose.Cells documentation, we were able to quickly demonstrate the ease of its usage and supportability to our clients.

We centralized file conversion into a standard format using a .NET web service. To address the requirement for a more streamlined client format on-boarding process, we made the mapping from the client format to the standard format configurable using XML.



Our biggest challenge was how to configure the read of client formats supporting varying data region start and end on any row. Our configuration supported the notion of a record Start Marker, Marker and End Marker.

	A	B	C	D
1				
2				
3	Date:	201011130		
4	Client ID:	XXX1		
5	Agent:	XXX1-001		
6				
7	Buys			
8	Account			Security ID
9	A1234567890			AAAAAAA
10	B1234567890			BBBBBBBB
11	C1234567890			CCCCCCC
12	D1234567890			DDDDDDD
13				
14	Sells			
15	Account			Security ID
16	A1234567890			EEEEEEEE
17	B1234567890			FFFFFFF
18				
19	Cancels			
20	Account			Security ID
21	A1234567890			XXXXXXXX
22	B1234567890			ZZZZZZZZ



For each record definition, we used Aspose.Cells to test the cell in the defined column position ("A") for the Start Marker Value ("Buys"). We used the Start Marker Length ("1000") to define how many record cells would be checked before determining that the data region was not provided within the worksheet. If found the record pointer is offset by the start record value ("2"), which indicates where the data records should actually start.

The record Marker was then used to test the cell in the defined column position ("A") for the Marker Value (".\*", a regular expression) to ensure it was a valid record. The fields for each valid record were then mapped into the standard format. We used a similar configuration concept and Aspose.Cells to create and write the standard Excel format (which is beyond the scope of this case study).

The End Marker was then used to test the cell in the defined position ("A") for End Marker Value ("%NOTHING%" which we used to indicate an empty or null value cell). The End Marker Length

("1") defined how many times the End Marker could be found before the end of the data region was determined.

The whole procedure is then repeated for each data region defined as a record in the configuration file.

Several other features of Aspose.Cells have proven useful in supporting varying source file formats, namely Excel Version support ranging from CSV and tab delimited format to Excel 97, 2000, XP, 2003 and 2007.

## Experience

Since implementation in 2009, there has not been one production issue involving the read of varying formatted files. From almost daily issues to no issues at all is definitely attributable to the stability of Aspose.Cells.

Our client has now on-boarded over 20 different source formats using our solution and have streamlined source file format on-boarding from 21-40 days to a repeatable 5-10 day end to end cycle. They are now also using the solution to convert their own internal Excel file formats to other standard formats.

## Next Steps

We have since extended the solution for our client to generate varying formatted outbound files supporting configurable formulas, conditional cell values and extensive style support.

We are currently planning to extend the solution to support configurable conditional formula and/or style placement (When <Some Condition> Then Use <Formula 1> When <Some Other Condition> Then Use <Formula 2> Otherwise Use <Formula 3> / When <Some Condition> Then Use <Style 1> When <Some Other Condition> Then Use <Style 2> Otherwise Use <Style 3>).

## Summary

The integration of Aspose.Cells for .NET into our file management solution offering multi source format support contributed to a successful implementation and operational phase for

our client. The usability, stability and supportability of Aspose.Cells won confidence in our client's application and ultimately our client's confidence in us to continue to deliver solid and robust solutions.