EXCELREBEL FOR FERTILIZER TONNAGE REPORTING

Updated 2013-12-16
Table of Contents

Introduction .................................................................................................................. 1
Overview ...................................................................................................................... 1
Why the Name ExcelRebel? ......................................................................................... 2
Data Sources ............................................................................................................... 2
What ExcelRebel Is and Isn’t ..................................................................................... 3
A More Integrated Approach .................................................................................... 5
Operational Features ................................................................................................. 5
  Basic Operational Features .................................................................................... 5
  Advanced Operational Features .......................................................................... 6
Introducing the Sample ............................................................................................... 7
Mapping Process ........................................................................................................ 9
  Overview ................................................................................................................. 9
  Cell Mapping Part 1 ............................................................................................... 9
  Abbreviations ......................................................................................................... 9
  Cell Mapping Part 2 ............................................................................................... 10
  Table Mapping .................................................................................................... 11
  Referencing Ancestors ......................................................................................... 12
Manager Features ..................................................................................................... 12
Developer Features .................................................................................................. 14
Frequently Asked Questions ...................................................................................... 16
Introduction

Companies that buy and sell fertilizer in the United States must comply with state requirements to report fertilizer sales and shipments. While much of the content requirements is common across states, each state has specified its own detailed content requirements and format requirements. Such state-by-state specificity leaves fertilizer tonnage reporters with a highly inefficient process of producing and formatting content in state-by-state-specific ways. In 2012, several AgGateway member companies, in coordination with various U.S. State officials and in coordination with The Fertilizer Institute, initiated a project to standardize the fertilizer reporting content and format requirements. The project team collected state reporting requirements for all states and produced an XML schema that specifies data structures and types capable of expressing tonnage reporting information in a way that satisfies requirements for all states. That schema, FertilizerTonnageReport, was published by AgGateway in collaboration with OAGi as part of Ag eStandards 5.3 in October 2013.

While moving to a standard way to express fertilizer tonnage reports is a productivity boost for reporters, state governments that agree to accept reports in the new industry-standard format face the challenge of integrating that new format into their systems. For some states, in-house staff can make quick work of the integration task. For other states, a third-party provider addresses such concerns. For the remaining states, transforming content into a format that is already supported in their workflow is a requirement. It is for states in this situation that ExcelRebel for Fertilizer Tonnage Reporting might be useful.

Overview

ExcelRebel for Fertilizer Tonnage Reporting is a freely-available, lightweight, and easy-to-use data integration tool. While the desired outcome of implementing such a tool is mostly-hands-free operational use, the tool has several features that assist developers and their managers during the mapping and configuration process. In a nutshell, ExcelRebel looks for new tonnage reports in an input folder, and for each one transforms the content from the Ag eStandards format to Excel, and optionally, flat file formats.
Why the Name ExcelRebel?

Excel was designed to enable users to develop and express complex numerical data and models, like mortgages and capital budgeting analysis. ExcelRebel rebels in the sense that it uses Excel as a database file. Of course this is a rather playful use of the term rebel given that Excel is commonly used for all manner of tables and lists.

**Excel as Intended**

![Excel as Intended](image1)

**As Commonly Used**

![As Commonly Used](image2)

Data Sources

Companies that buy and sell fertilizer are required by various U.S. state governments to report fertilizer sales, receipts, and movement. Using AgGateway’s Ag eStandards version 5.3 FertilizerTonnageReport standard,
companies can express the required reporting information in a format that meets each U.S. state’s requirements. The schema specifies data structure and data type requirements for reports. The data-structure parts of the schema specify requirements like: “The report has a header section and a details sections. Each details section can have one or more detail sections.” The data-type parts of the schema specify requirements like: “Dates must be of the form YYYY-MM-DD.” Conforming reports will be expressed in an XML syntax that meets the requirements specified in the schema. In technical XML terms, a FertilizerTonnageReport is valid if it meets all the requirements specified in the XML schema.

At this point all we’ve addressed is that companies deliver industry-standards-compliant reports to state governments. It’s time to dig deeper into what ExcelRebel and XML schemas are and aren’t.

What ExcelRebel Is and Isn’t

End-to-end system integration between two parties requires a solution with several components and capabilities. These include, but are not limited to, the following:

1. An agreement between the parties regarding the process they are automating with their systems integration. This includes, among other things, what data (messages) are exchanged in what format in response to what circumstances.
2. The capability of each party to produce messages in the agreed-upon format in response to the agreed-upon circumstances.
3. An agreement between the parties regarding how messages are delivered from the sending party’s system to the receiving party’s system and how such receipt is acknowledged by the receiving party’s system.
4. The capability of each party to send and receive messages according the agreed-upon approach.
5. The capability of each party to consume messages that conform to the agreed-upon format, and if specified, provide a business-level

---

1 The following discussion should not be considered a recipe for an integration project.
acknowledgement. Such acknowledgements may include various levels of business validation.

6. Components, technologies, and processes that satisfy various security-related requirements.

Within the overall system-to-system integration picture painted above, we can locate ExcelRebel, thereby understanding what it is and is not. ExcelRebel applies exclusively to number five—consuming messages. To more narrowly locate it, we must decompose message consumption, which includes, but is not limited to, the following components and capabilities:

1. Store the messages in a persistent data store (e.g., database, file system, cloud service)
2. Perform security validation, if necessary (e.g., authentication, authorization, integrity, non-repudiation—the first two of which may be implemented as transport-layer functions)
3. Confirm XML well-formedness (the message follows the rules of XML syntax)
4. Confirm that the XML is valid against the schema (the message conforms to the structures and data types specified in the schema)
5. Confirm that message is complete, consistent, and conforms to other business rules.
6. Transform the message into a format (or formats) suitable for ingestion into the receiving system.
7. Ingest the message content.
8. Confirm conformance to additional business rules not validated in number six.
9. Provide business-level acknowledgements to the sender per agreement.

Within this set, ExcelRebel addresses number six. As a side effect of accomplishing number six, ExcelRebel also completes number three. (ExcelRebel can’t transform non-well-formed XML.) In the image below, ExcelRebel addresses the blue-colored boxes.
A More Integrated Approach

At this point, an obvious question might be: “Excel, really? Isn’t there a more integrated approach?” Let’s be clear that ExcelRebel is not a preferred long-term link in the integration chain. To use a popular phrase from Larry the Cable Guy, ExcelRebel is a *git’er done* temporary solution. Some states just need data in Excel or flat-file formats to ensure continuity with existing processes. That said, the process of mapping the XML message to Excel is effort that may be leveraged later. The knowledge and even some of the mechanics of completing the mapping may be applied to a more permanent solution.

Operational Features

Basic Operational Features

The basic process follows. (Configurable variations are described in the Advanced Operational Features section.) When ExcelRebel is executed, it checks for
XML files in a folder named 1.INPUT, which is a child folder of the same folder containing the ExcelRebel executable. ExcelRebel opens each XML file and checks the document element. If the document element is FertilizerTonnageReport, ExcelRebel looks in the 2.TEMPLATES folder for an Excel file configured to process fertilizer tonnage reports. If one is found, the file is transformed into an Excel file and one or more flat files. A zip file containing the Excel file and any flat files are stored in the 3.OUTPUT folder. The input file is zipped and moved to the 4.Archive folder. The directory structure is illustrated in the following image.

When the process is complete, the result is that all of the reports have been transformed and saved, and the input files archived.

**Advanced Operational Features**

The following image shows the content of the Settings worksheet in the template file.

The *Sheets in Output* section provides settings to show, hide, or delete worksheets in the output file. The reason for these options is primarily for allowing deletion of sheets in operational settings in the case where system administrators want to limit file sizes. Once the transformation has been
completed, there is no transformational reason to keep the five worksheets in the Excel file although there may be operational reasons for doing so.

The Export section provides settings that control the output as follows:

- Export sheets: yes/no value to control whether the table-type sheets are exported to flat files
- If Export sheets is set to yes, the following settings apply:
  - Delimiter: The delimiter to be used in the flat-file export. The value can either be a single character or tab, which indicates a tab character.
  - Export first row: Controls whether the first row—the header row—is exported.
  - Zip exported files: Controls whether or not the exported files are zipped, and if so, whether or not the Excel file is included.
  - Include ExcelRebel keys: Controls whether or not the exported files include the transformationally generated primary/foreign keys (used to relate nested “records”).

The Developer section will be discussed later in this document.

**Introducing the Sample**

A sample input file and corresponding sample template file is included with ExcelRebel. The sample input file contains two lists: 1) A list of all NFL teams organized by conference and by divisions within conference; 2) A list of all Super Bowls, their location, contestants, and outcome. An excerpt of the input file is shown in the image below (note that many elements have been collapsed in the editor).
The sample template (i.e., mapping) file corresponding to the sample data above is the source of many of the example images throughout the remainder of this document.
Mapping Process

Overview

Mapping is the process of specifying a transformation of an input format into an output format. The result of the process is a “map” that specifies how data in an input file should be represented in an output file.

ExcelRebel can represent report data in two ways. One way is like a database table with rows and columns. Often the first row contains column names. This type of mapping is typical of the kind of data that repeats, like line items in a purchase order. We’ll refer to this approach as table mapping. Another way is to represent the data in a cover-sheet sort of way—specific values in the source file placed in specific Excel cells in the destination file. We’ll refer to this approach as cell mapping. It’s important to understand that ExcelRebel only exports table-mapped data to flat files.

Cell Mapping Part 1

As mentioned earlier, cell mapping specifies how specific values in the source file are mapped to specific Excel cells. There are two parts to this process:

- Specify the path to the piece of data you want to select in the source file.
- Specify the Excel cell where the selected piece of data should go.

To specify the path to the data you want, list the elements in order that drill down to the element or attribute containing the data. (For those familiar with XML technologies, this is basically an XPath without concern for the namespace—just local names.) This is a good place to point out another useful feature: abbreviations.

Abbreviations

XML element names may sometimes be long and consequently tedious to type and tedious to read—especially when several of them are strung together. A developer may specify abbreviations on the Abbreviations worksheet. For example, FertilizerTonnageReport could be abbreviated FTR,

---

2 Experienced XPath users will note that the statement is not precisely correct in that the index to the value is not specified. The author views the precision of the discussion “close enough” for the intended audience.
ExcelRebel for Fertilizer Tonnage Reporting

_FertilizerTonnageReportBody_ abbreviated _FTRBody_, and _FertilizerTonnageReportDetails_ abbreviated _FTRDetails_. Doing so would change this path:


to this:

_FTR/FTRBody/FTRDetails_

The abbreviations may be any case. It’s important not so use an abbreviation that is the same as an actual element or attribute name. The image below shows the abbreviations defined for the sample content and template supplied with ExcelRebel.

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Element or Attribute Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>League</td>
<td>SportsLeague</td>
</tr>
<tr>
<td>ConfS</td>
<td>Conferences</td>
</tr>
<tr>
<td>Conf</td>
<td>Conference</td>
</tr>
<tr>
<td>Divs</td>
<td>Divisions</td>
</tr>
<tr>
<td>Div</td>
<td>Division</td>
</tr>
<tr>
<td>abbr</td>
<td>abbreviation</td>
</tr>
</tbody>
</table>

_Cell Mapping Part 2_

Back to cell mapping. Once you specified the data you want, optionally using abbreviated element names, you specify what cell the data should go in. To do that, in the appropriate cell in the mapping sheet press the “=” key, click on the cell you want the data to go in (which would be in another sheet), and press the enter key. That’s it. ExcelRebel will examine the reference to put the right data in the right places. In the image below, three paths are specified using abbreviations (an attribute name is not abbreviated, showing that just because you define an abbreviation, you don’t have to use it.) In the _Spreadsheet Cell_ column, references are made to cells on another sheet. The reason values are present in that column is because the referenced cells contain sample content, which shows up when referencing the cell. (If this is hard to follow, don’t worry, it should come together when looking through the sample template file.) Note that a column is provided to specify default values. Default values will be used when a value cannot be found in the input file. (Note that no default values are specified below.)
Table Mapping

Table mapping is similar to cover-sheet mapping with the following differences:

- Instead of specifying the entire path for each value, you represent the path hierarchically down to an element that repeats (i.e., an element that effectively acts as a container for a record).
- Instead of mapping to a specific cell for the value, you map to specific column header (a cell in the first row) in a specific sheet. Values will be inserted beginning on the second row.

In the image below, you can see the hierarchical mapping. The first column specifies the path to the “parent record” (technically “parent element”), or if there isn’t one, the root. Put another way, this is the container for the repeating “records”. The second column specifies the path to the “record”—the element that repeats, which is a container for one or more values. The third column specifies the path to the value. Note that as with cell mappings, default values may be specified.
Referencing Ancestors

Sometimes when specifying a data-table map, you need to pull in a value from an ancestor. ExcelRebel provides a simple operator for this: “^”, which you can think of as “go up”. It is helpful to consider the sample content and template to grasp this concept.

Manager Features

Mapping is a tedious and generally non-creative process. It’s not the sort of task that developers look forward to. Consequently, often junior developers are assigned such tasks. Another characteristic of mapping is that it’s often difficult for managers to monitor progress. The combination of a difficult-to-monitor process and junior developers may put project timelines at risk.

ExcelRebel provides two features to help managers monitor progress:
1. The input content, mapping definition, and mapping output are conveniently assembled in a single Excel file.

2. The input color-coding to indicate unmapped content.

Excel is a convenient file format and user interface for viewing content. A junior programmer working on mapping can easily respond to a manager question, “How’s it going?” by running ExcelRebel and showing/emailing the resulting Excel file. The source content, mapping definition, and most important the output are all contained within the Excel file.

Example cell-mapping (like a cover sheet) output:

![Example cell-mapping output](image1.png)

Example table mapping output:

![Example table-mapping output](image2.png)
Furthermore, ExcelRebel color codes unmapped values in red, indicates the mapping for mapped values, and leaves structure blank and not color-coded. It’s important to understand that red cells may be red because the developer hasn’t reached that point in his/her development, or because the value is not important and won’t be mapped.

### Developer Features

With mapping/data transformations, there are three distinct activities: 1) Managers manage the integration project in which mapping is required; 2) Developers specify the mapping; 3) The mapping is plugged into an integration pipeline for operational data transformation. We’ve covered management features and operations, and we’ve covered how to map. This section covers additional features intended to enable the development process to move along more quickly. The image below shows three sections of settings, the first of two we covered in the Operational Features section.
The developer settings allow a developer to override operational behavior when transforming test input files. This helps save time examining results and setting up for their following development iteration. Any file with the characters .TEST. (case-insensitive) in the filename are considered test files (e.g., NFL.test.xml, My.TEST.File.xml). The setting are:

- Archive test files: When the value is set to yes, test files will not be archived, leaving the test files as they are in the input folder. This saves developer time in that they don’t have to copy input files back into the input folder nor delete excess files in the archive folder.
- Show all sheets for test files: When the value is set to yes, all sheets are shown in the output for input test files. A consequence of this is that the Sheets in Output settings are ignored. This allows for easy testing without adjusting Sheets in Output settings.
- Don’t zip exported test files: When the value is set to yes, output files are not zipped. A consequence of this is that the Zip exported files setting is ignored. This may save developer time in not having to unzip output files to examine the result.

One other developer feature is the information shown on the input worksheet. As the image below illustrates, a developer can look at the input file expressed hierarchically, see exactly what mapping matched the value and
Exactly where the value was stored, see what values have not been mapped (red cells), and see the XPath\(^3\) to the value.

**Frequently Asked Questions**

Q: Must Excel be installed on the computer doing the transformations?
A: No, ExcelRebel does not require Excel to perform transformations.

Q: What prerequisites are required to run ExcelRebel on a Microsoft Windows computer?
A: Microsoft .NET 4.5, which is freely available. (Search on *Microsoft .NET Framework 4.5 download*.)

Q: Will ExcelRebel run on a Mac? Linux?
A: Although it hasn’t been tested as of this writing, it should. Contact AgGateway for further information if your needs require Mac or Linux use.

Q: What’s up with the *ExcelRebel* name?
A: See the associated section in this document.

---

\(^3\) Experienced XPath users will note that the statement is not precisely correct in that the index to the value is not specified. The author views the precision of the discussion “close enough” for the intended audience.
Q: Shouldn’t my integration process be more sophisticated than to require Excel?
A: See the *A More Integrated Approach* section in this document.

Q: Can ExcelRebel for FertilizerTonnageReporting be used for other message types?
A: ExcelRebel can be licensed for transforming any XML content. It is perpetually royalty-free available for FertilizerTonnageReporting.