Minatek® Flat File Converter

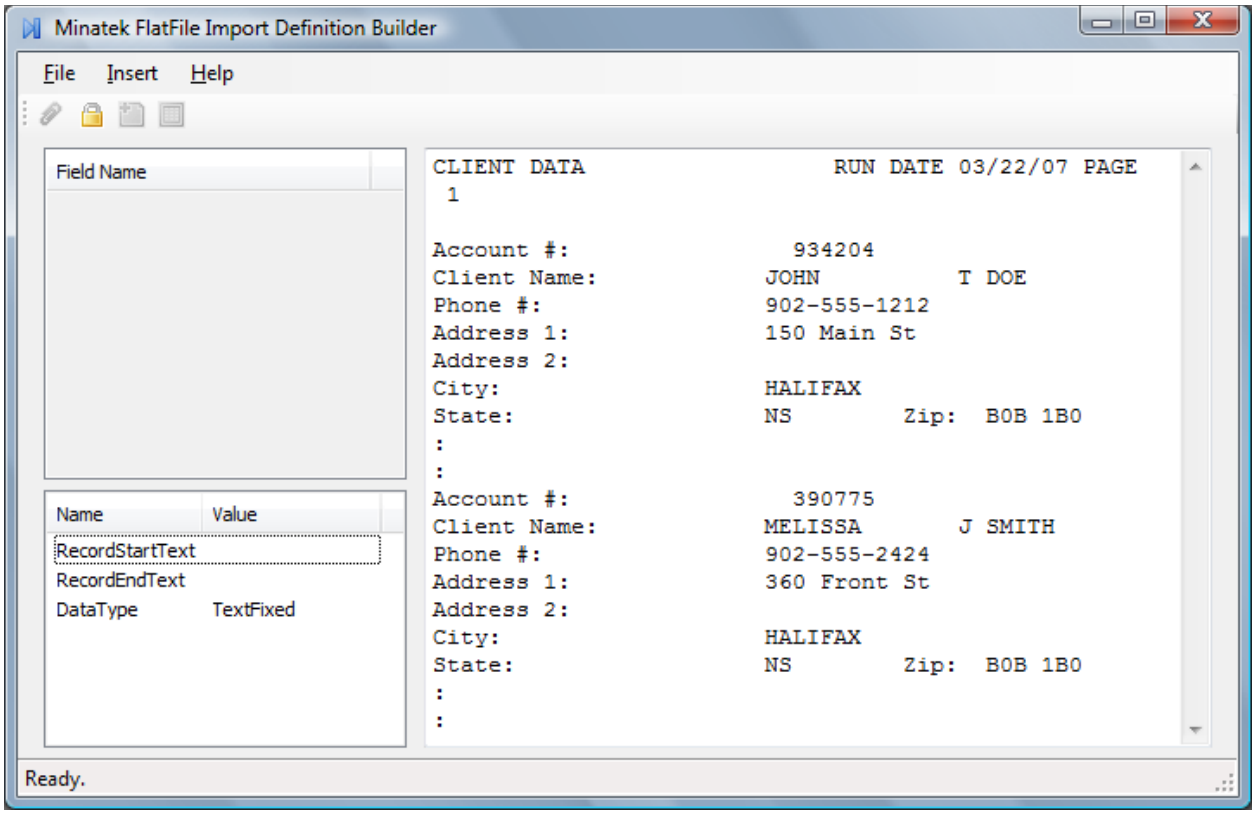
This .NET class offers a simple way to convert fixed-text flat files, such as report dumps from legacy systems, into useable data. By either manually creating a definition file or using the Definition Builder you can convert files to a variety of formats.

The following document will discuss how to use the definition builder and the class inside your application.

Definition Builder

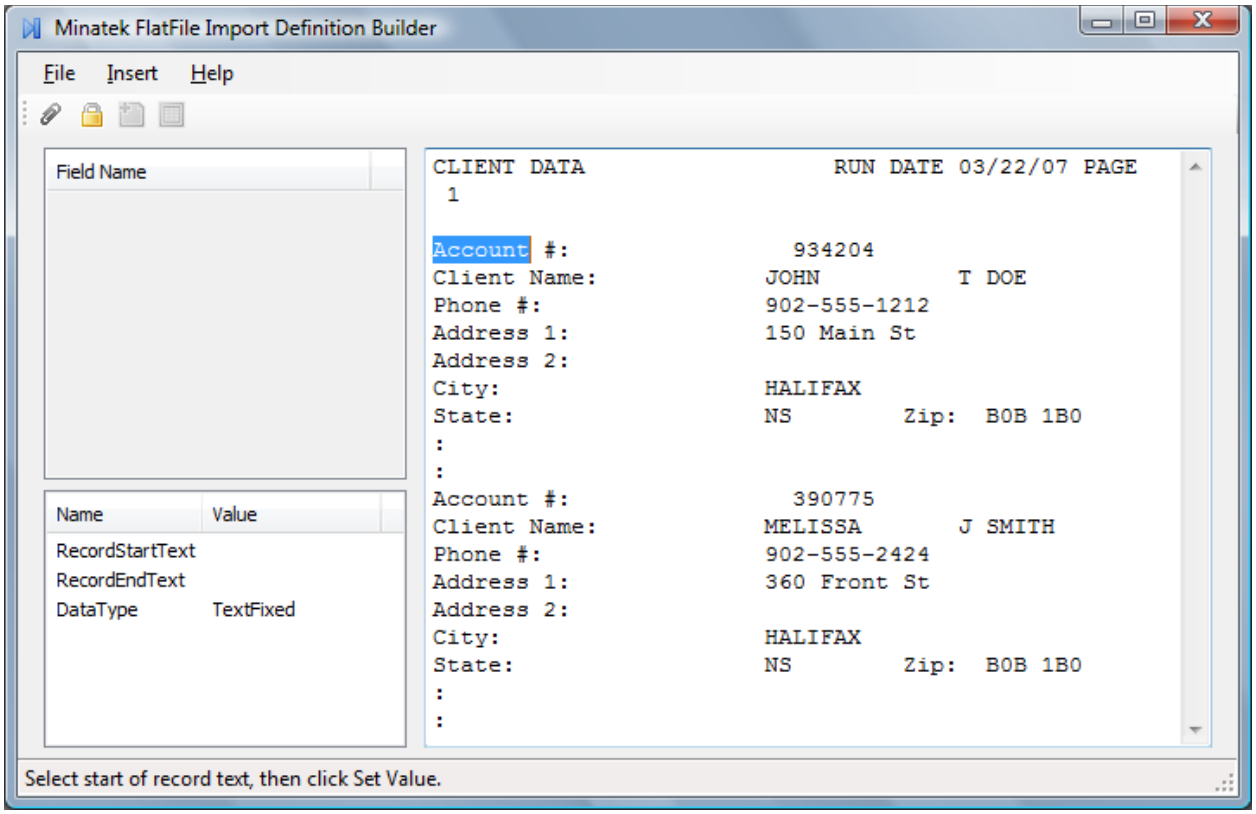
Use the Definition Builder to create the XML definition files for fixed-width data. These next few screenshots will show the process of creating such a file.

After starting the application, click File | New to create a new definition. First, you must select the flat file to work with. This will load the data in the design screen so it is easier to define it. Use the *SampleDataFixed* file found in package containing the application download for an example.



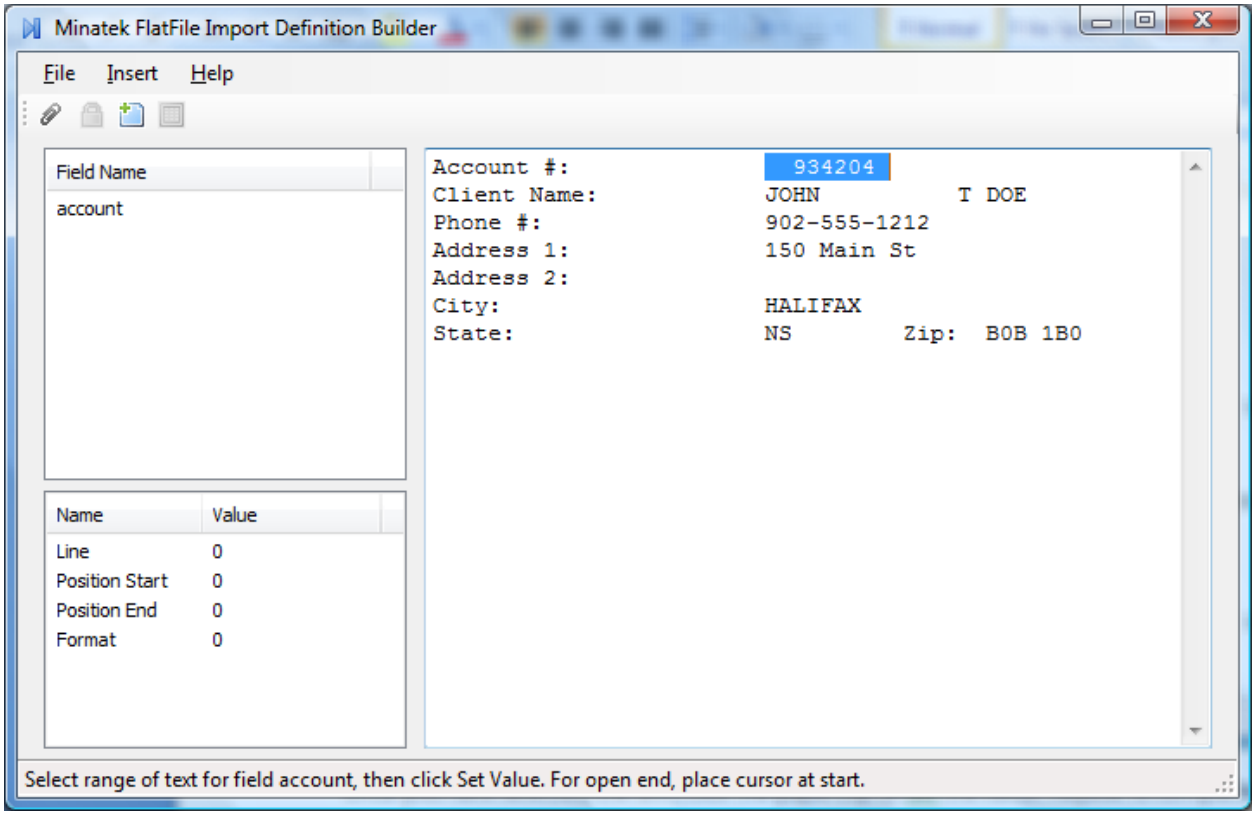
On the right, you will now see your raw data. At the bottom left you must now define the start and end record text so that the system will know what differentiates records. Double-click on the

RecordStartText, and then highlight the portion of text that signifies the start of a record. Ensure it is unique, such as *Account* in this sample.



Once you have selected the text, click the paperdip to *Set* the value. Repeat to complete the *RecordEndText*, which we'll use the colon in the line after *State*. Now, press the padlock icon on the toolbar to lock the set. This will define the data and display just one record of the data so that you can define the fields.

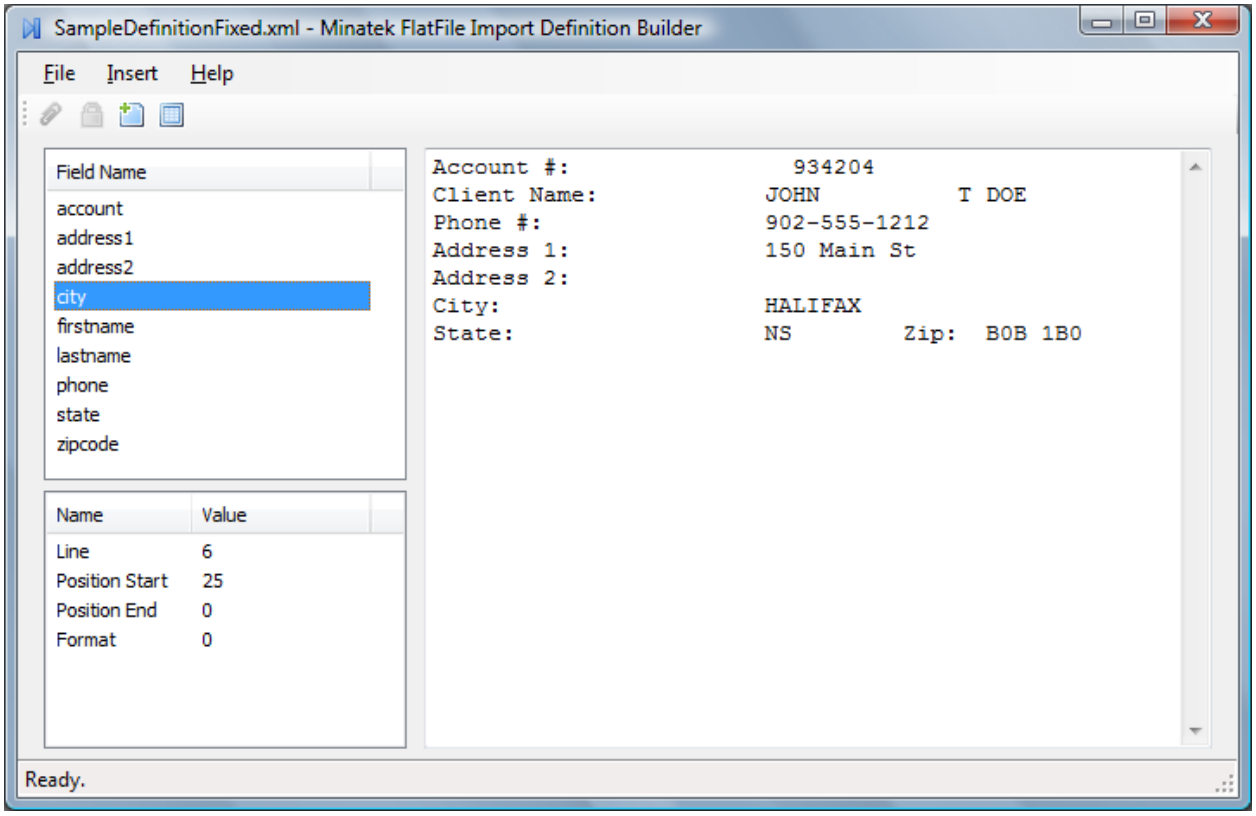
Now, click the third icon on the toolbar, the paper with a plus sign (*Add Field*). Specify the dataset name you wish to give this field. Keep in mind this is how it will be called in the dataset or resulting data from using this application.



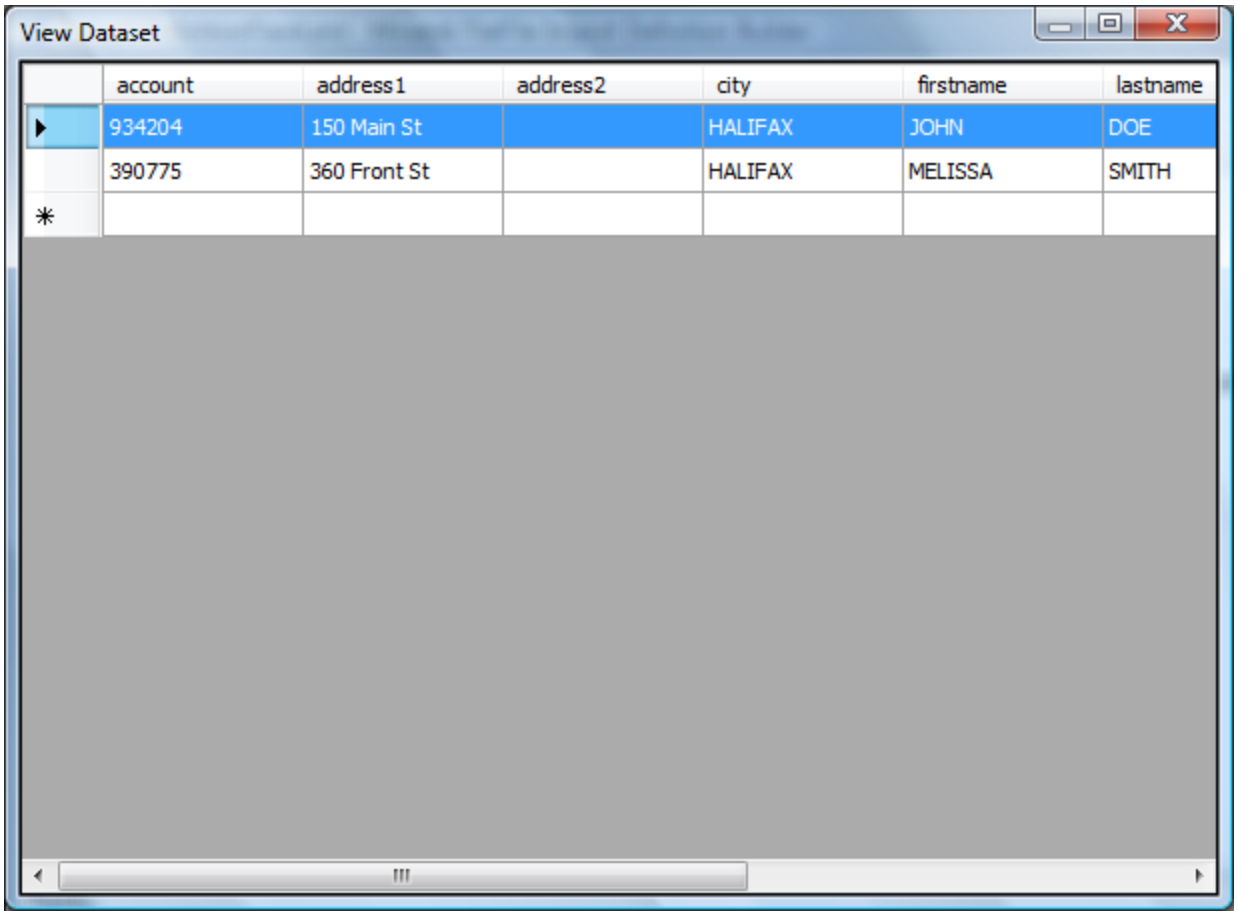
To specify what constitutes the account number, double-click one of the items in the bottom-left pane, highlight the region of text that has the account number, remembering to including leading spaces to where the record may grow, and then click the paperclip to set the *Line*, *Position Start*, and *Position End* automatically.

If a field runs from a certain start position to the end of the line, just place the cursor at the start of the data and click *Set Value*—this is handy for fields with open-ended values that terminate at end-of-line.

Just repeat for each field you wish to gather.



As you can see, all of the fields have been defined. The definition is ready to be saved and previewed. By clicking the last icon on the toolbar, *View Dataset*, you are able to view your data file against the definition in a database like manner.



Open the sample project in Visual Studio to see a Visual Basic .Net application use the sample definition file and data to return a dataset for the application's use.

Definition Files

A definition file stores details about the source data, such as type, and column name as location details. The functions available in the class allow referencing a definition file on the hard drive, or passing in the file contents so that another service or an embedded resource may be used.

Below are two examples: one for fixed-width, the other for delimited (CSV).

Fixed-Width Definition File Sample

As seen in the previous section, fixed-width definition files may be created by the Definition Builder. Typically a fixed-width data file is generated by a legacy system from a save-to-file report function, and while they are easy to read by a human they are more difficult to use in applications.

The following is a sample data file of containing client information:

```
CLIENT DATA                    RUN DATE 03/22/07 PAGE    1

Account #:                 934204
Client Name:               JOHN          T DOE
Phone #:                   902-555-1212
Address 1:                 150 Main St
Address 2:
City:                      HALIFAX
State:                     NS        Zip:  B0B 1B0
:
:
Account #:                 390775
Client Name:               MELISSA       J SMITH
Phone #:                   902-555-2424
Address 1:                 360 Front St
Address 2:
City:                      HALIFAX
State:                     NS        Zip:  B0B 1B0
```

Next is the definition file that contains the values needed to process these records:

```
<?xml version="1.0" encoding="utf-8" ?>

<definition version="1.0">
  <recordstarttext>Account</recordstarttext>
  <recordendtext>:</recordendtext>
  <datatype>TextFixed</datatype>
  <fields>
    <field name="account" line="1" posstart="25" posend="34" format="0"/>
    <field name="firstname" line="2" posstart="25" posend="39" format="0"/>
    <field name="lastname" line="2" posstart="41" posend="0" format="0"/>
    <field name="address1" line="4" posstart="25" posend="0" format="0"/>
    <field name="address2" line="5" posstart="25" posend="0" format="0"/>
    <field name="city" line="6" posstart="25" posend="0" format="0"/>
    <field name="state" line="7" posstart="25" posend="35" format="0"/>
    <field name="zipcode" line="7" posstart="41" posend="0" format="0"/>
    <field name="phone" line="3" posstart="27" posend="0" format="0"/>
  </fields>
</definition>
```

As you can see in the sample data, each record starts with the account number and ends with a colon; therefore, the *recordstarttext* and *recordendtext* contain "*Account*" and "*:*", respectively.

Fields are very straightforward. The name is how the resulting dataset or XML file should call this column. Line numbers are one-based, with one being the line the value found in the *recordstarttext* element is.

In this example, the account number is found on line 1, starting at character position 25 on that line and ending at character position 34 on that line.

A variation on that is when the end position may vary. Just as the address fields you may simply specify a zero for the *posend* attribute. This will start at *posstart* and read to the end of the line.

Delimited Definition File Sample

Delimited text files may sometimes be referred to as comma-separated or CSV files. Basically any file with one character separating fields or columns and each line being a new row may be processed with this method.

The following sample data contains a ZIP-code with its city and state:

```
"New York City","NY",10001
"Beverly Hills","CA",90210
```

To process this file, it is defined in the following manner:

```
<?xml version="1.0" encoding="utf-8" ?>

<definition version="1.0">
  <datatype>Delimited</datatype>
  <delimiter>,</delimiter>
  <textqualifier>"</textqualifier>
  <fields>
    <field name="zipcode" position="3" format="0"/>
    <field name="statecode" position="2" format="0"/>
    <field name="cityname" position="1" format="0"/>
  </fields>
</definition>
```

Since the *datatype* is "*Delimited*" the attributes needed are different. Since the sample data is a comma-separated values file, meaning a comma separates the columns, the *delimiter* element contains "*,*". The *textqualifier* element ensures that text inside the qualifier are taken as a whole, and not as a separate record.

An example of this could be a record containing a name in the last name comma first name format:

> "Doe, John", "Washington", "DC"

Without the quotation-mark text-qualifier, "*Doe*" and "*John*" would be considered separate columns, possibly mixing up the data.

minatek.flatfileconvert

There are several functions available to manipulate data. Some allow the path of the data and definition files to be passed in, others allow a text stream be passed in. One may output to a file, another return the data. This allows one to either compile in definitions and data into an application or open files, or any combination.

Ultimately, this is XML based; however, a data.dataset object may be returned in some cases as well.

ConvertToFile – Outputs a formatted XML file based on an XML stylesheet, or XSLT, file.

ConvertToExcelXMLSpreadsheet – Outputs to an Excel XML Spreadsheet based on an XSLT file.

> ProcessInstructions is a string array containing processing instructions found in the header of an XML file. Each instruction requires two sets of data: the name and the value. This makes this array have a bound of two for each instruction.

ConvertToXML – Returns an XML Document (System.Xml.XmlDocument)

ConvertToDataset – Returns a dataset containing records (System.Data.Dataset)

Support and Suggestions

This product has limited support. You may submit support requests and suggestions at the product website. See the terms-of-use and license agreement file included in the folder.

http://www.minateksolutions.com/tools/flatfileconvert