Algorithmica
Research AB

Creating XML position file from a CSV text file

using Quantlab 3.0

Last update: 2011-09-27 written by Robert Thoren.

XML file creation from a simple CSV text file

As we frequently get the question on how to produce XML files that are suitable for processing for example positions from one system to the next, here is a small example on how to do it. The example is based on Qlang language in order to be able to put the code directly into a production environment using the Quantlab Batch Server. Once in the production environment, it can be either event-triggered or scheduled.

Our example will take a CSV text file with a header row created using the "Save as csv file" in MS Excel. In order to make the code a bit more generic, we will code so that the column order does not matter, i.e. we will use a "look-up" function in order to find the specified data in the right column.

The output will, as it happens, be a valid XML position scheme of the ARMS risk management system for the position type equity.

In data specs

```
dealid;portf;pos_type;name;isin;id;id_type;nominal;eq_val;ccy;issuer;trader;mkt_cap;country;sector;cpty;style;systemcode
A01;Fond1;equity;AddtechB;SE0000808370;ADDTb.ST;RIC;13100;199;SEK;Addtech;pelle;SMALL;SE;IT;OMX;GROWTH;oldsystem:123
A02;Fond1;equity;AlfaLaval;SE0000695876;ALFA.ST;RIC;6400;133.5;SEK;AlfaLaval;pelle;MEDIUM;SE;INDRIAL;OMX;GROWTH;oldsystem:124
A003;Fondbolag1;equity;Autoliv;SE0000382335;ALIVsdb.ST;RIC;70;456.5;SEK;Autoliv;pelle;MEDIUM;SE;AUTO;OMX;GROWTH;oldsystem:125
```

Here we have a position file with the bare minimum of information that our XML will require. If any other information is needed it could either be hard coded or looked up from a database table or another file.

The procedure

The flow of things goes something like this:

1) First create a container of headers from which to look-up values.
2) Read the file using a streaming in-format. The streaming will process a row at a time and ensure that for very large files, the memory is not burst. In this example we could have read all at once using a file_read instead.
3) For each read row, parse (or tokenize as it is referred to) the row into sub-elements, i.e. the columns.
4) Create a XML structure to fit the desired output format.
5) Write the XML structure to file.
6) Encapsulate the actual procedure in a function that can be called and scheduled in the Quantlab Batch Server environment.

For all of the steps above there should be proper handling of errors since we aim at placing the code in a semi-supervised server environment. If we were to only do this once we could skip all error handling a drag-drop the function into a Quantlab workspace. Then the Quantlab runtime engine would handle all of our error handling for us, visually.

What we want is a file structure like the one below, having an outer data tag and then a repetition of positions tags <pos>. Within each pos tag we will have four sub-parent tags. A <pf> tag for portfolio info, an instrument specific tag, here <equity> positions, a classification tag used for grouping, and lastly a user defined tag in which the data is optional and user defined.

```
<Data>
        <pos>
                <pf>
                …
                </pf>
                <equity>
                …
                </equity>
                <classification>
                …
                </classification>
                <user_def>
                …
                </user_def>
        </pos>
        <pos>
                <pf>
                …
                </pf>
                <equity>
                …
                </equity>
                <classification>
                …
                </classification>
                <user_def>
                …
                </user_def>
        </pos>
</Data>
```

The Qlang Code

When reading the code, the main function is found at the bottom of the code. In Qlang, within an expression window, all helper functions (sub-routines) must be available above.

The code starts at the bottom with a main function that can be directly placed on a Quantlab Batch Server and scheduled or triggered whenever a new file is placed in a certain directory. The input/output directory and file name are the only inputs to the main wrapper function.

```
module MY_XML{
// A map container function that returns the header names and their order in the file
map_istr_num create_htc(vector(string) vh)
{
   map_istr_num  m = map_istr_num();
   number        n = v_size(vh);

   for (number i = 0; i < n; i++)
        m.add(vh[i], i);
   return m;
}

// A sub-function that creates the portfolio “pf” tag
xml_node pf_xml   (       date            upload_date,
                          string          pf,
                          string          id,
                          string          type,
                          string          trader)
{
```

```
    xml_node xn = xml_node("pf");

    xn.add_tag("upload_date").add_text(string(upload_date));
    xn.add_tag("portfolio").add_text(pf);
    xn.add_tag("type").add_text(type);
    xn.add_tag("customer_id").add_text("DemoCust");          // SOME THINGS HARD CODED
    xn.add_tag("system").add_text("SYSTEM");                 // MORE HARD CODING
    xn.add_tag("id").add_text(id);
    xn.add_tag("trader").add_text(trader);

    return xn;
}

// A sub-function that creates the classification tag
xml_node get_classification(vector(string) v, map_istr_num htc)
{
    xml_node xn = xml_node("classification");

    // Here we search for the data in a text vector using the index coming from the header map
    // the htc.find(<cpty>) will return the column index for the correct column
    xn.add_tag("counterparty").add_text(v[htc.find("cpty")]);
    xn.add_tag("issuer").add_text(v[htc.find("issuer")]);
    xn.add_tag("country").add_text(v[htc.find("country")]);
    xn.add_tag("sector").add_text(v[htc.find("sector")]);
    xn.add_tag("style").add_text(v[htc.find("style")]);
    xn.add_tag("market_cap").add_text(v[htc.find("mkt_cap")]);

    return xn;
}

// A sub-function that creates the “user_def” tag
xml_node get_userdef(vector(string) v, map_istr_num htc)
{
    xml_node xn = xml_node("userdef");

    // Here we search for the data in a text vector using the index coming from the header map
    // the htc.find(“systemcode”) will return the column index for the correct column
    xn.add_tag("systemcode").add_text(v[htc.find("systemcode")]);
    xn.add_tag("isin").add_text(v[htc.find("isin")]);

    return xn;
}

// A sub-function that creates the actual position tag
// Here we use the header lookup map to find the correct data in each row vector
xml_node get_equity(vector(string) v, map_istr_num htc)
{
    xml_node xn = xml_node("equity");          // OUTER PARENT TAG <equity>
    xml_node id = xml_node("id");              // AN INNER PARENT TAG <id>

    xn.add_tag("name").add_text(v[htc.find("name")]); //ADDING INFO TO TAG
    xn.add_tag("currency").add_text(v[htc.find("ccy")]);
    xn.add_tag("nominal").add_text(v[htc.find("nominal")]);

    xn.add_tag("eq_quote").add_text(v[htc.find("eq_val")]);

    id.add_tag("id").add_text(v[htc.find("id")]);
    id.add_tag("type").add_text(v[htc.find("id_type")]);
```

```
   xn.add_child(id);              //ADDING THE INNER PARENT TAG TO PARENT TAG

   return xn;
}

// Main helper function where the work is done. Called by main wrapper that can be scheduled
void import_file      (in_stream              is,
                       out_stream             os,
                       date                   upload_date,
                       map_istr_num           htc)
{
   string func = "import_file";

   number ok                 = 0;
   number failed             = 0;
   //Reading the text file row by row in an infinite loop. Break when empty.
   for (;;) {
      string s = is.read_line(); //From the in-stream read first line

      if (null(s))
         break;

      xml_node pos =     xml_node("pos");  //Create outer <pos> tag
      vector(string) v =     str_tokenize(s, ";");  //break apart string into vector of strings

      try {

// Call the four sub-functions above to create the sub-xml nodes
         xml_node pf = pf_xml(upload_date,
                              v[htc.find("portf")],
                              v[htc.find("dealid")],
                              "equity",
                              v[htc.find("trader")]);
         xml_node cl       = get_classification(v,htc);
         xml_node eq       = get_equity(v, htc);
         xml_node ud       = get_userdef(v, htc);

      //Add the sub-xml nodes to the parent <pos> tag
         pos.add_child(pf);
         pos.add_child(eq);
         pos.add_child(cl);
         pos.add_child(ud);

         if (!null(pos)) {
            pos.print(os);     // Write a complete row as a <pos> with all its children to out-stream
file
            ok++;
         }
      }
      catch {
         ahs_log_report(AHS_MESSAGE_ERROR, func, err.message());
         failed++;
      }
   } // END FOR LOOP
}  // END MAIN HELPER FUNC
} // END MODULE MY_XML (keep functions in custom namespace)

//Create a production style function wrapper that can be run with parameters from a batch server
//It is created so that it will read a path and file name in and create a file on a drive out.
// The function will also archive the read file to an archive folder once it is finished
```

```
out void import_single_file(    string          in_d,
                                string          in_f,
                                string          out_d,
                                string          out_f,
                                string          archive_dir)
{
    string func = "import_single_file";

    string in_path     = strcat([in_d,"\\", in_f]);
    string out_path    = strcat([out_d,"\\",out_f]);
    string archive_path = strcat([archive_dir, "\\", in_f]);

    ahs_log_report(AHS_MESSAGE_INFO, func, strcat("Started import from file: ", in_path));

    //Upload_date is today
    date   upload_date       = today();

    //create an instream and outstream file in memory
    in_stream is       = in_stream_file(in_path);
    out_stream  os   = out_stream_file(out_path);

    //Start by writing the XML version in the header of the XML file.
    os.write("<?xml version=\"1.0\" encoding=\"iso-8859-1\" ?>");
    os.write("<Data>");

    //Call the map function to place headers and corresponding column id into a lookup map
    map_istr_num htc_map      = MY_XML.create_htc(str_tokenize(is.read_line(), ";"));

    //Call the main helper function to both read and write our files
   MY_XML.import_file(is, os, upload_date, htc_map);

    is.close();
    os.write("</Data>");
    os.close();

    file_rename(in_path, archive_path);         //SEND FILE TO ARCHIVE AND WE ARE DONE

    ahs_log_report(AHS_MESSAGE_INFO, func, strcat("Finished import from file: ", in_path));

}
```

The resulting XML file

Now we will see the resulting file in the directory that we specified in the main function. It looks something like this:

```
<?xml version="1.0" encoding="iso-8859-1" ?><Data><pos>
    <pf>
      <upload_date>
            2011-09-27
      </upload_date>
      <portfolio>
            Fondbolag1
      </portfolio>
      <type>
            equity
      </type>
      <customer_id>
            DemoCust
      </customer_id>
      <system>
            SYSTEM
      </system>
      <id>
            A001
      </id>
      <trader>
            pelle
      </trader>
    </pf>
    <equity>
      <name>
            Addtech B
      </name>
      <currency>
            SEK
      </currency>
      <nominal>
            1310000
      </nominal>
      <eq_quote>
            199
      </eq_quote>
      <id>
            <id>
                  ADDTb.ST
            </id>
            <type>
                  RIC
            </type>
      </id>
    </equity>
    <classification>
      <counterparty>
            OMXNASDAQ
      </counterparty>
      <issuer>
            Addtech
      </issuer>
      <country>
            SE
      </country>
      <sector>
            IT
      </sector>
      <style>
```

```
            GROWTH
        </style>
        <market_cap>
            SMALL
        </market_cap>
    </classification>
    <userdef>
        <systemcode>
            oldsystem:123
        </systemcode>
        <isin>
            SE0000808370
        </isin>
    </userdef>
</pos>
<pos>
    <pf>
        <upload_date>
            2011-09-27
        </upload_date>
        <portfolio>
            Fondbolag1
        </portfolio>
        <type>
            equity
        </type>
        <customer_id>
            DemoCust
        </customer_id>
        <system>
            SYSTEM
        </system>
        <id>
            A002
        </id>
        <trader>
            pelle
        </trader>
    </pf>
    <equity>
        <name>
            Alfa Laval
        </name>
        <currency>
            SEK
        </currency>
        <nominal>
            6400000
        </nominal>
        <eq_quote>
            133.5
        </eq_quote>
        <id>
            <id>
                ALFA.ST
            </id>
            <type>
                RIC
            </type>
        </id>
    </equity>
    <classification>
        <counterparty>
            OMXNASDAQ
        </counterparty>
        <issuer>
            Alfa Laval
```

```
      </issuer>
      <country>
            SE
      </country>
      <sector>
            INDUSTRIAL
      </sector>
      <style>
            GROWTH
      </style>
      <market_cap>
            MEDIUM
      </market_cap>
    </classification>
    <userdef>
      <systemcode>
            oldsystem:124
      </systemcode>
      <isin>
            SE0000695876
      </isin>
    </userdef>
</pos>
```