```
        ctl-opt dftactgrp(*no) debug(*xmlsax) option(*nodebugio);

        // To compile this program
        // CRTSQLRPGI OBJ(XMLLIB/SAXPARSES)
        // SRCFILE(XMLLIB/QRPGLESRC)
        // COMMIT(*NONE) DBGVIEW(*SOURCE)

        dcl-c  num_elements   const(500);
        dcl-s  parsingstatus  char(1);
        dcl-s  parseddata     char(256);
        dcl-s  xmldoc         char(50);
        dcl-s  options        char(20)  inz('doc=file');
        dcl-s  currentdate    date inz(*sys);

        dcl-ds saxctlds  extname('SAXCTL') qualified end-ds;


         // Communications area definition
         dcl-ds  MyCommArea;
           namecount        int(10);
           elementdata      dim(num_elements);
             name             char(64)  overlay(elementdata);
             startcount       int(10)   overlay(elementdata:*next);
             endcount         int(10)   overlay(elementdata:*next);
         end-ds;

         dcl-pr  EventHandler int(10);
           commArea     likeds(MyCommArea);
           event        int(10)   value;
           pstring      pointer   value;
           stringlen    int(20)   value;
           exceptionID  int(10)   value;
        end-pr;


         // Read unprocessed records only (field prcflag = *blank)

         exec sql declare C1 Cursor For
           Select * from saxctl where prcflag = ' ';

          exec sql close C1;
          exec sql open C1;

          exec sql fetch C1 into :saxctlds;


          dow sqlcode = *zero;

         // Delete any records in output file with same path and docum
ent name
```

```
          // If any are found, they are from a previous run

           exec sql
           delete from saxdata
            where (xmldocpath = :saxctlds.prcdocpath) and
                  (xmldocname = :saxctlds.prcdocname);


          // Build the document path into field "xmldoc"
          // The document folder will always be in the root directory
         clear xmldoc;
          xmldoc = '/' + %trim(saxctlds.prcdocpath) + '/'
                       + %trim(saxctlds.prcdocname);


          // Begin the parsing, processing is similar to a subroutine.
          // Control will be passed to the "handler" for each event enc
ountered.
          // If an error occurs, mark the process flag with an 'E'
        monitor;

            parsingstatus = *blank;

        xml-sax %handler(EventHandler : myCommArea) %xml(%trim(xmldoc):
options);

          // Mark parsingstatus field for control file record as proces
sed.
          // Processed values are 'E' for Error, 'X' for no errors foun
d.

          on-error *all;
            parsingstatus = 'E';
          endmon;

          if parsingstatus = *blanks;
            parsingstatus = 'X';
          endif;

           exec sql
             update saxctl set processed_flag = :parsingstatus,
                   processed_dattim = current timestamp
               where (processed_path = :saxctlds.prcdocpath) and
                     (processed_doc = :saxctlds.prcdocname);

          // Process the next record in saxctl file
          exec sql fetch C1;

        enddo;
```

```
          // When here at eof, end job
           *inlr = *on;
          return;

         // *************************************************************
**************************
          // This handler will be called each time a new event is encou
ntered.
          // A long complex document can easily call it hundreds
          // or thousands of times.


       dcl-proc  EventHandler;

         dcl-pi *n int(10);
            commArea     likeds(MyCommArea);
            event        int(10)   value;
            pstring      pointer   value;
            stringlen    int(20)   value;
            exceptionID  int(10)   value;
         end-pi;

          dcl-s    string          char(65535)  based(pstring);
          dcl-s    returnCode      int(10)  inz(*zero);
          dcl-s    element         int(10);

           if stringlen > *zero;
            parseddata = %subst(string : 1 : stringlen);
           endif;

        select;

        // Communications area definition
       when event = *XML_START_DOCUMENT;
         clear commArea;

       when event = *XML_START_ELEMENT;
       exec sql
         insert into saxdata
           values('Start_Element', :parseddata,
                   :saxctlds.prcdocpath, :saxctlds.prcdocname);

       when event = *XML_END_ELEMENT;
       exec sql
         insert into saxdata
           values('End_Element', :parseddata,
                   :saxctlds.prcdocpath, :saxctlds.prcdocname);

       when event = *XML_ATTR_NAME;
       exec sql
```

```
         insert into saxdata
           values('Attr_Name', :parseddata,
                   :saxctlds.prcdocpath, :saxctlds.prcdocname);

       when event = *XML_ATTR_CHARS;
       exec sql
         insert into saxdata
           values('Attr_Chars', :parseddata,
                   :saxctlds.prcdocpath, :saxctlds.prcdocname);

       when event = *XML_CHARS;
        if parseddata > *blanks;
       exec sql
         insert into saxdata
           values('XML_Chars', :parseddata,
                   :saxctlds.prcdocpath, :saxctlds.prcdocname);
       endif;

        endsl;
        return returnCode;

       end-proc;
```