XLPARSE2
Use RPG to read an Excel XLS/XLSX spreadsheet

by *Giovanni B. Perotti* (Italy)

XLPARSE2 is based on the March 2010 Scott Klement's XLPARSER4 service program, POI 3.6 and other Java supports, which are all bundled in the package.
XLPARSE2 supports
- *XLS* spreasheets generated from MS Excel 2003 and subsequent releases, including O365
- *XLSX* spreasheets generated from MS Excel 2007 and subsequent releases, including O365

1- About it

XLPARSER4 -a great utility developed by Scott Klement- is a set of procedures for reading an Excel XLS/XLSX spreadsheet with RPG and JAVA. In his package, Scott included some Java classes which refer to some other Java classes called *Jakarta POI* that the user should install by himself.

- The **Jakarta Project** creates and maintains open source software for the Java platform. It operates as an umbrella project under the auspices of the Apache Software Foundation.
- **POI** (a pure Java port of Microsoft's popular file formats) is one of the projects formerly part of Jakarta, but now an independent project within the Apache Software Foundation.

Scott also provided a few sample RPG programs to print *specific* Excel XLS/XLSX spreadsheets.

I wanted something more.
I wanted a tool to convert to a standard database file *any* Excel XLS/XLSX spreadsheet, so that any application program could then process the data collected from a spreadsheet.

The outcome was a library, named XLPARSE2, containing
- An installation procedure that installs the .jar Java classes (both from POI 3.6, from Scott Klement and others), needed by the Scott's XLPARSER4 service program, his sample programs, plus some code of mine.
  This installation procedure runs under the covers and requires no decisions from the installer.
- A command, XLSCONVERT, that converts any Excel XLS/XLSX spreadsheet to a physical file in library QTEMP, that you can process using our XLSGETCELL procedure.
- A command, XLSTABLE, that converts any Excel XLS/XLSX spreadsheet to a physical database file that you can very easily process with your programs.

This page explains how to install and how to run this utility.

Use **this link** to read some articles from Scott that I have saved into my pages.

XLPARSER4 restriction- This Scott Klement's service program can parse at maximum the first 256 columns (cells) of Excel spreadsheet rows. Subsequent cells are ignored.

Maintenance- As any other utility from the **Easy400.net** site, *XLPARSE2* is maintained as needed (fixes, new features, etc.). In order to know whether a new release is available and what is new there, please **refer to the maintenance page**.

2-Prerequisites

- OS/400 release V5R3 or subsequent
- Library QSYSINC, product 57xxWDS, opt. 13
- Library QHTTPSVR, product 57xx-DG1
- Compiler ILE RPG IV, product 57xxWDS, opt. 31
  This is just needed to create objects duting the installation of the utility.
- (*optional*) Library **CGIDEV2** (service program CGIDEV2/CGISRVPGM2) downloaded from site www.easy400.net .
  If this is available, you may run **some examples of XLPARSE2 CGI programs**.
- Developer Kit for Java, product 57xxJV1, *base
- If OS/400 release V5R3:
  - Product 57xxJV1, option 6 -
    Java Developer Kit Classic 4.0 (java version 1.4) .
  - On OS/400 release V5R3 only .XLS worksheets can be processed, .XLSX worksheets require at least OS/400 release V5R4.
- If OS/400 release V5R4 or a subsequent one:
  - Product 57xxJV1, opt. 7 -
    Java Developer Kit Classic 5.0 (java version 1.5) .
    This component is available from V5R4M0 on.
    On subsequent OS/400 releases, further JDK (Java Developer Kit) versions (options 8, 9, etc.) are available.
    For more information on IBM i 57xxJV1 product options vs OS/400 releases, see this page.
- **Warning on 57xxJv1** - *Make sure to have installed the last PTF cumulative for 57xxJV1* ***!!!***

3-Installation

1. Download file *xlparse2.zip* from the **Easy400 download page** and unzip it.
2. Follow the *xlparse2.txt* instructions to upload and to restore library XLPARSE2
3. On the IBM System **i**, logon with a user profile having special authority ***JOBCTL** and run the following procedure:
   ```
   STRREXPRC SRCMBR(INSTALL) SRCFILE(XLPARSE2/QREXSRC)
   ```
   It does the following:
   - creates service program GPPARSER4 (some procedures needed to support command XLSCONVERT)
   - creates Scott's service program XLPARSER4 (parsing Excel spreadsheet procedures)
   - creates Scott's sample programs
   - creates utilities XLSCONVERT and XLSTABLE
   - creates library XLPARSE2DT and populates it with some objects that will contain local data
   - restores IFS directory /xlparse2 .
     Note that subdirectory /xlparse2/java contains all the Java classes (from POI, from Scott Klement and others) needed by this utility.
   - if library CGIDEV2 (service program CGIDEV2/CGISRVPGM2 is available, displays the directives of HTTP instance XLSPARSE2 (initial comments tell how you can create such HTTP instance), otherwise
     displays the HTTP directives that you may add to an HTTP instance of yours to make these WEB pages available on your IBM System **i**.
4. **Remove** any POI-related and any XLPARSE-related objects from **/QIBM/UserData/Java400/ext**. Adding such Java objects to this

directory is a bad practice and should be discontinued. **Read Scott Klement's recommendation!**
Do the following:

i. After installing XLPARSE2, run command `WRKLNK '/xlparse2/java/*'`
and take a note of all the subdirectories there.

ii. Then run command `WRKLNK '/QIBM/UserData/Java400/ext'`
and make sure that no such subdirectories exist in the */QIBM/UserData/Java400/ext* directory.

In other words, you must make sure that none of the following IFS objects are left in directory **/QIBM/userdata/Java400/ext** or in any of its subdirectories:

/dom4j-1.6.1.jar	/poi-contrib-3.2-final-20081019.jar	/poi-ooxml-schemas-3.6-20091214.jar
/jsr173_1.0_api.jar	/poi-scratchpad-3.2-final-20081019.jar	/poi-ooxml-3.6-20091214.jar
/xlparse.jar	/poi-3.2-final-20081019.jar	/poi-3.6-20091214.jar
xbean.jar		

For more information about positioning Java objects, see Appendix **JVMSTARTUP**.

5. Last, to validate the installation run the following command:
`XLPARSE2/XLSCONVERT`
The first time this is done, a screen similar to **this** shows up. **Do not do anything, ... just wait!**
The next time you run command XLSCONVERT, it will be much faster.
Note- If command *XLPARSE2/XLSCONVERT* fails with Java error message
*Error occurred while parsing spreadsheet after cell (0,0) in *Unknown** , the reason could be one of the following:
   - Directory */QIBM/UserData/Java400/ext* still contains some POI-related and / or some XLPARSE-related objects
   - You are missing some PTF(s) for product 57xxJV1. Install the latest PTF CUM for product 57xxJV1.
6. **Important note** - If this utility is installed on a V5R3 box, when the box is updated to a subsequent OS/400 release, it is necessary to run command `xlparse2/compile` .

- **Re-installing XLPARSE2 on another box**
To re-install XLPARSE2 from one box ("*source system*") to another box ("*target system*"), proceed as follow:
   - If the target system has an OS/400 release *VxRyMz* lower than the source system one, and the source system supports such previous release, on the source system you must re-compile the XLPARSE2 programs by running command
`STRREXPRC SRCMBR(INSTALL) SRCFILE(XLPARSE2/QREXSRC) PARM(VxRyMz)`
   - On the *source system* save library XLPARSE2 (specify parameter TGTRLS if needed)
   - On the *target system*, **remove** any POI-related and any XLPARSE-related objects from **/QIBM/UserData/Java400/ext**
   - On the *target system*, restore library XLPARSE2
   - On the *target system*, run command **xlparse2/install**.

4-A major tip

For a correct operation of this tool, it is mandatory that library XLPARSE2 is in the job library list *before the Java Virtual Machine (JVM) is started*. Therefore, make sure to run command `ADDLIBLE XLPARSE2` as soon as the job is started.

09/06/2022 20:31:59

XLSCONVERT utility

Convert an Excel XLS/XLSX spreadsheet to a db file

1- Why

Use of Excel is very widely spread in small companies and in large companies departments for local processes. Allowing to receive inputs from Excel XLS/XLSX spreadsheets may increase the integration role of the iSeries. Excel inputs could be sent by e-mail and received by the iSeries: MMAIL utility provides a way to receive e-mail messages and to detach Excel attachments as individual IFS stream files.

On the iSeries, what is needed is a process able to convert *any* Excel XLS/XLSX spreadsheet to a standard database file, that can subsequently be processed by iSeries applications.

Command XLPARSE2/XLSCONVERT is exactly that tool.

2- Command XLSCONVERT

Command XLPARSE2/XLSCONVERT converts a given spreadsheet of an Excel XLS or XLSX workbook - residing on the IFS as a stream file - to a database file in library QTEMP. Once the conversion is finished, the QTEMP database file can be read by a local application.

- The Java job execution environment must not have been already set (through the command ADDENVVAR ENVVAR(CLASSPATH) ...). If a classpath other than the expected one was set, you will receive a Java escape message saying that a class was not found.
- The **user profile** running command XLSCONVERT **must have special authority *JOBCTL**. This is required by command ADDENVVAR which is executed by the program.
- If the command XLSCONVERT is executed from a program, the program is recommended be created with ACTGRP(XLPARSE2).
- The IFS stream file must be an Excel XLS or XLSX ASCII workbook containing at least one spreadsheet.
- The database file containing the converted data is physical file QTEMP/XLSOUTF.

```
                  Convert an Excel spreadsheet (XLSCONVERT)

 Type choices, press Enter.

 .xls/xlsx stream file  . . . . . XLS         ____________________________________

 Sheet sequence number  . . . . . SHEETNBR     1            1-999, *ALL
 'Sheet name' . . . . . . . . . . SHEETNAME    *SHEETNBR
 Display the database file  . . . DSPDBF       *NO          *YES, *NO
 Print results  . . . . . . . . . PRINT        *NO          *YES, *NO
 Decimal precision  . . . . . . . DEC          2            0-6
 Columns forced as numeric  . . . FRCNUMCOLS   *NONE        1-500, *NONE
                + for more values
 Date columns . . . . . . . . . . DATECOL      *AUTO        Number, *AUTO, *NONE
                + for more values
 Date format  . . . . . . . . . . DATEFMT      *ISO         *ISO
 Time columns . . . . . . . . . . TIMECOL      *NONE        Number, *NONE
                + for more values
 Time separator . . . . . . . . . TIMESEP      *            *, :, :, ., ,, -
 Run in a sync submitted job  . . SBMJOB       *YES         *YES, *NO
 Job queue  . . . . . . . . . . . JOBQ         QSYSNOMAX    Name
   Library  . . . . . . . . . . .                *LIBL      Name, *LIBL
 Type of failure message  . . . . FAILMSG      *ESCAPE      *ESCAPE, *DIAG

                          Additional Parameters

Set output record ID . . . . . . SETOUTID     *NO          *YES, *NO
```

- **.xls/xlsx stream file (XLS)** - This is the qualified name of the stream file Excel Workbook.
  IFS directory '/xlparse2/samples' contains a number of Excel workbooks that can be used for testing command XLSCONVERT.
- **Sheet sequence number (SHEETNBR)** - The sequence number of the spreadsheet you want to process. Specify **ALL* if you want to convert all the spreadsheets together.
  Instead of specifying the spreadsheet sequence number in parameter SHEETNBR, you may specify the spreadsheet name in parameter SHEETNAME. This is allowed only if:
  - the IBM i OS release is at least V76R1,
  - utility HSSFCGI is installed, and
  - service program HSSFCGI/XLPARSERJP is available.

  Note. When you specify a sheet name in parameter SHEETNAME, the value specified in parameter SHEETNBR is ignored.
- **Sheet name (SHEETNAME)** - The name of the sheet to be processed. This value is case sensitive and must be specified within quotes (').
  When you specify the spreadsheet name in parameter SHEETNAME, the sheet sequence number in parameter SHEETNBR is ignored.
  Parameter SHEETNAME requires
  - at least IBM i OS release V6R1,
  - utility HSSFCGI must be installed, and
  - service program HSSFCGI/XLPARSERJP must be available.
- **Display the database file (DSPDBF)** - Select one of the following:
  - *YES to display file QTEMP/XLSOUTF once the conversion is complete. This option can be used for testing.
  - *NO to avoid displaying file QTEMP/XLSOUTF once the conversion is complete.
- **Print results (PRINT)** - Whether a printout of the converted spreadsheet is desired.
- **Decimal precision (DEC)** - Number of decimal digits to be shown after the decimal point. When a spreadsheet cell contains a numeric value (example 142.27), this value is retrieved as a floating point number (example 1.422700000000E+002). That does not tell how many decimal digits should be displayed after conversion.
  This is why this piece of information must be supplied through this parameter.
  This parameter applies to *all* cells containing numeric values.
- **Columns forced as numeric (FRCNUMCOLS)** - Columns containing number-format cells and text-format cells do result into a character-data fields in the generated database file record format.
  To force these columns to generate numeric-data fields in the generated database file record format, you must mention their numbers (1 for column A, etc.) in the entries of this parameter.
  Up to 50 such columns may be specified.
  Of course, if a specified column happens to contain just number-format cells, a numeric-data record format field is generated.
  A text-format cell, to be converted to a number must containg a string where
  1. the minus ("-") sign, if needed, is the first character

2. thousand separators are not used
3. the "decimal point", if needed, is either character (".") or character (",") according to the national language of the Excel spreadsheet.

Valid examples:
- English language spreadsheet: -8927.353 or 8927.36
- Italian language spreadsheet: -8927,353 or 8927,36

- **Date columns (DATECOL)** - In Excel spreadsheets, a date is stored as a floating point number. This number represents the number of days elapsed since January 1, 1900. In order to have numeric values converted back to dates, do one of the following:
  - Type *NONE to avoid converting any numeric value to a date value.
    This is the recommended choice unless you know that the spreadsheet contains at least one date column.
  - Enter up to 50 column numbers to identify the columns where numeric values must be converted to date values.
    This is the recommended choice when you know that the spreadsheet contains at least one date column.
  - Type *AUTO to let the program establish which columns contain numeric values to be converted to date values.
    This choice may provide unexpected results on numeric columns and is suggested only for test purposes, in order to find out if any date columns exist in the s readsheet.

- **Date format (DATEFMT)** - Only *ISO date format (yyyy-mm-dd) is supported.

- **Time columns (TIMECOL)** - In Excel spreadsheets, the internal representation of a time value (example: 16:48:56) is a number (example: 700648). This number measures the time in millionths of a day (example of computation: there are 84,600 seconds in 24 hours; time 16:48:56 is 69,536 seconds; therefore the Excel internal representation of time 16:48:56 is (69,536/84,600)*1,000,000=700648 .
  In order to have Excel time numeric values converted back to a conventional time format (example: hh.mm.ss), do one of the following:
  - Type *NONE to avoid converting any numeric value to a time value. This is the recommended choice unless you know that the spreadsheet contains at least one time column.
  - Enter up to 50 column numbers to identify the columns where numeric values must be converted to time values. This is the recommended choice when you know that the spreadsheet contains at least one time column.
- **Time separator (TIMESEP)** - This parameter is not made available when *NONE is specified for parameter TIMECOL.
  Select one of the following:
  - `*` to use the time separator defined in system value QTIMSEP.
  - `:` to use a colon (:) as a time separator (example of result: 16:48:56)
  - `.` to use a period (.) as a time separator (example of result: 16.48.56)
  - `,` to use a comma (,) as a time separator (example of result: 16,48,56)
  - `-` to use a blank ( ) as a time separator (example of result: 16 48 56) .
- **Run in a sync submitted job (SBMJOB)** - If you specify SBMJOB(*YES), the XLSCONVERT command is executed in a separate submitted job. The current job waits until the submitted job completes, then it resumes execution.
  This can be useful in two cases:
  - you do not want XLSCONVERT to start a Java Virtual Machine in the current job, OR
  - you do not want XLSCONVERT to run in the current job because it may conflict with a Java Virtual Machine already active in this job.

  Should you absolutely need to run this command in the current job, though the JVM has already been started by a previous Java application, read about command `JVMSTARTUP` in Appendix JVMSTARTUP.
  Important note for SBMJOB(*YES)
  When SBMJOB(*YES) specified, the QTEMP files (example:XLSOUTF) created in the QTEMP library of the submitted job are saved and restored to the QTEMP library of the submitting job.
- **Job queue** - If you specify SBMJOB(*YES), this is the job queue where the synchronized job is submitted.
- **Type of failure message (FAILMSG)** - When you specify SBMJOB(*YES), the XLSCONVERT command is executed in a synchronized submitted job.
  It may however happen that the XLSCONVERT command fails during its execution. In such a case a message is sent to the current program in the current job. You may choose whether this message would be an *ESCAPE or a *DIAG (diagnostic) one.
- **Set output record ID (SETOUTID)** - The records of the output file QTEMP/XLSOUTF have a field named OUTID. The value of this field is intended to represent the data type of a record:
  - B means "sheet title"
  - H means "column headers"
  - D means "data columns"

  If you specify SETOUTID(*NO), no attempt is made to differentiate the record data type, and all records are assigned value D in field OUTID.
  If you specify SETOUTID(*YES), an attempt is made to identify the record data types according to the three values previously listed.

3- Spreadsheets containing formulas

XLSCONVERT is able to parse three types of cells: text, number, formula.
Formulas are Excel expressions that are computed at Excel execution time.
When you look at an Excel spreadsheet, you cannot tell whether the cell values that you read come from some formulas. To find it out, on the Excel tool bar you have to:

1. press the *Formula tab*
2. press the *Show formulas* button

XLSCONVERT can tell whether the value resulting from a formula is a number or a character string, but - with the exception of a few cases - cannot compute the value of a formula. In such cases, the conversion result is blank.

When you have a spreadsheet containing formulas, the best thing is to copy all the spreasheet cells to the clipboard and paste "the cell values" to an empty sheet of another workbook.
Details of this operation are documented in Microsoft Office Support page Copy cell values, not formulas.
Commands XLSCONVERT, XLSTABLE and XLSTABLE2 will have no problems in converting the spreadsheet resulting from this copy operation.

4- An XLSCONVERT failure

In some cases, XLSCONVERT may send the escape message *"No significant cells detected in this sheet"*.
This may happen when a spreadsheet was generated from some software tool and some properties are missing.
You can manually fix the Excel workbook on a PC in this way:

- Open the spreadsheet with Excel (2003 or subsequent)
- Save the spreadsheet; Excel will add the missing properties.

Should you need that be done in an IBM i procedure of yours, you may bypass this problem by specifying in parameter `SHEETNAME` the name of the spreadsheet you want to process.
Of course, utility HSSFCGI with service program XLPARSERJP is required.

5- The output file QTEMP/XLSOUTF

This database file contains the data converted from an Excel XLS or XLSX spreadsheet via command XLSCONVERT. There is a record for each row.
A record contains information about the first 256 cells of a row (subsequent cells are ignored).
The record format is as follow:

1. Field name OUTSHEET (50A) - The sheet name this row belongs to.
2. Field name OUTSEQ (5S 0) - The number of this row within the sheet.
3. Field name OUTID (1A) - The content-type of this row. Possible values are:
   - B - Sheet title
   - H - column headers
   - D - data columns
4. Field name OUTNBRCOL (3S 0) - Number of columns in this row.
5. Field name OUTDTA (5000A) - The contents of all the columns of this row.
   A given column has always the same size across all rows in the same sheet. The data-type and the length of each column are documented in the next two fields.

6. Field name OUTCOLLEN (256 subfields, each 4B 0) - Each subfield contains the size (number of bytes) of a column.
7. Field name OUTCOLTYPE (256 subfields, each 1A) - Each subfield contains the data-type of a column. Possible values are:
   - C - Character
   - N - Numeric value in a character string;
     Please note that the decimal point used is always a dot, character "." .
   - S - Null value

6- Printing an XLS/XLSX spreadsheet on the iSeries

Command XLSCONVERT does that for you, whatever the layout of the XLS or XLSX spreadsheet is.
Just enter command `XLSCONVERT XLS(...) PRINT(*YES)` and you are done!
Check out **our example**.

7- Reading an XLS/XLSX spreadsheet on the iSeries

To read an XLS or XLSX spreadsheet (residing on the IFS as a stream file), you have the following options:

1. Write your own program and use directly the Scott Klement's utilities provided in service program XLPARSER4 (see **this article**). This is what I did to write program XLSCONVERT. Sample exercises are provided by Scott with programs XLPDEMO and XLPDEMOF (included in library XLPARSE2).
2. Use my command XLSCONVERT to convert the spreadsheet to database file QTEMP/XLSOUTF.
   Your ILE-RPG program supposed to read the spreadsheet, must
   a. Be created with **ACTGRP(XLPARSE2)**
   b. Invoke a simple subprocedure (named ***XlsGetCell***) to receive the spreadsheet cells.
      Use
      i. *XlsGetCell('STR')* to position before the first cell of the converted spreadsheet available in file QTEMP/XLSOUTF
      ii. *XlsGetCell('GET')* to retrieve all the cells, one at a time
      iii. *XlsGetCell('END')* to reset positioning.

   In order to demonstrate how this is easy, I wrote a small sample program that does exactly that. It is named GETCELLS and here is its source:

```
*===========================================================================
* Create this program as follow:
* CRTBNDRPG PGM(XLPARSE2/GETCELLS) SRCFILE(XLPARSE2/QRPGLESRC)
*           DFTACTGRP(*NO) ACTGRP(XLPARSE2) DBGVIEW(*SOURCE)
* NOTE that activation group MUST be XLPARSE2
*===========================================================================
H BNDDIR('XLPARSE2/XLPARSE2')
H optimize(*NONE)
H decedit(*JOBRUN)
H truncnbr(*NO)
H option(*srcstmt : *nodebugio)
 *Prototype of XLSGETCELL subprocedure
D XLSGETCELL      PR           565
D  Action                       3    value  options(*nopass)
 *===========================================================================
 * Procedure XLSGETCELL, any time it is called,
 * returns a data structure containing information
 * about the next spreadsheet cell.
 *  This information is retrieved from physical file QTEMP/XLSOUTF.
 * This file contains the database version of the last spreadsheet
 * converted by command XLSCONVERT.
 * The following is the layout of the information data structure
 * returned from this subprocedure:
D InfoDS          ds
 * Return code: 0=cell found, -1=No more  cells.
D  rc                           10i 0
 * Sheet name
D  xsheet                       50
 * Row number of this cell
D  xrownbr                      11s 0
 * Type of row: B=Sheet title, H=Column headers, D=Data columns
D  xID                          1
 * Number of columns in this row
D  xnbrcol                      9s 0
 * Column number of this cell
D  xcolnbr                      9s 0
 * Type of the data in this cell: C=character, N=numeric (edited), S=null value
D  xcoldtatyp                   1
 * Estimated column length of this cell
D  xcollen                      10i 0
 * Data in this cell
D  xcoldta                      500
 *===========================================================================
 * Main line
 *===========================================================================
 /free
   rc=0;

   InfoDS=xlsgetcell('STR');      //Start process

   //loop getting spreadsheet cells, until no more cells (rc=-1)
   dow  rc=0;
        eval InfoDS=xlsgetcell('GET'); //get the info about the next cell
   enddo;

   InfoDS=xlsgetcell('END');      //END process

   *inlr=*on;
   return;
 /end-free
```

09/06/2022 20:32:03

XLSTABLE utility

Convert an Excel XLS/XLSX spreadsheet to a db file

1- What

After talking to some users about the XLSCONVERT utility, I realized that they needed something more.
It was not enough to provide a way to access an image of an XLS/XLSX spreadsheet (as the one created in library QTEMP from the XLSCONVERT utility) with their traditional ILE-RPG programming, via subprocedure *xlsgetcell*.
They simply wanted to have the an Excel XLS/XLSX spreadsheet transformed into a table: an obvious OS/400 physical file that can be read by any programming language for the System ***i***.
This is what commands XLSTABLE is about.

2- Command XLSTABLE

Command XLPARSE2/XLSTABLE generates and fills a physical file (target file) with the data from an Excel XLS/XLSX spreadsheet.

- The Java job execution environment must not have been already set (through the command ADDENVVAR ENVVAR(CLASSPATH) ...). If a classpath other than the expected one was set, you receive a Java escape message saying that a class was not found.
- The **user profile** running command XLSTABLE **must have special authority *JOBCTL**. This is required by command ADDENVVAR which is executed by the program.
- The Excel XLS/XLSX spreadsheet must be on the IFS and must be coded in ASCII characters (CCSID 819-ASCII ISO Latin 1, 1212-PC USA, or 1252-IBM PC).
- Though the spreadsheet may contain several spreadsheets, just one spreadsheet is processed.
- Only the first 256 cells of a row are retrieved, subsequent cells are ignored.

```
                    Generate PF from XLS/XLSX (XLSTABLE)

Type choices, press Enter.

.xls/xlsx stream file  . . . . . XLS         _____________________________________
_____________________________________________________________________________
Spreadsheet sequence number  . . SHEET       1__              1-999
'Sheet name' . . . . . . . . . . SHEETNAME   *SHEET_____________________________
Target file  . . . . . . . . . . FILE        __________       Name
  Library  . . . . . . . . . . .             __________       Name
Target member  . . . . . . . . . FILEMBR     *FIRST___        Name, *FIRST
Replace or add records . . . . . MBROPT      *REPLACE         *ADD, *REPLACE
(Re)create target file . . . . . CRTFILE     *NO              *YES, *NO
Source file  . . . . . . . . . . SRCFILE     QDDSSRC__        Name
  Library  . . . . . . . . . . .               *FILELIB__     Name, *FILELIB
Source member  . . . . . . . . . SRCMBR      *FILE____        Name, *FILE
Record format name . . . . . . . RCDFMT      *AUTO____        Name, *AUTO
Record format field names  . . . RCDFLDNMS   __________       Name, *NONE
              + for more values              __________
Header lines . . . . . . . . . . HLINES      0___             0-99
Columns forced as numeric  . . . FRCNUMCOLS  *NONE____        1-500, *NONE
              + for more values              _____
Field names  . . . . . . . . . . FLDNAMES    *DFT_____        *DFT, *COLHDG
Date columns . . . . . . . . . . DATECOL     *AUTO____        Number, *AUTO, *NONE
              + for more values              _____
Date format  . . . . . . . . . . DATEFMT     *YMD_____        *YMD, *MDY, *DMY, *ISO...
Time columns . . . . . . . . . . TIMECOL     *NONE____        Number, *NONE
              + for more values              _____
Time format  . . . . . . . . . . TIMEFMT     *HMS_____        *HMS, *ISO, *USA, *EUR, *JIS
Run in a sync submitted job  . . SBMJOB      *NO______        *YES, *NO
Job queue  . . . . . . . . . . . JOBQ        QSYSNOMAX__      Name
  Library  . . . . . . . . . . .               *LIBL____      Name, *LIBL
Type of failure message  . . . . FAILMSG     *ESCAPE__        *ESCAPE, *DIAG
Display target file  . . . . . . DSPFILE     *YES             *YES, *NO
```

- **.xls/xlsx stream file (XLS)** - This is the qualified name of the stream file (CCSID 819-ASCII ISO Latin 1, 1212-PC USA, or 1252-IBM PC) containing the Excel XLS/XLSX spreadsheet to be transformed into the physical file specified in parameter FILE.
  IFS directory '/xlparse2/samples' contains a number of spreadsheet stream files that can be used for testing command XLSTABLE.
- **Spreadsheet sequence number (SHEET)** - The sequence number (1 to 99) of the spreadsheet to be processed.
  Instead of specifying the spreadsheet sequence number in parameter SHEET, you may specify the spreadsheet name in parameter SHEETNAME. This is allowed only if:
  - the IBM i OS release is at least V6R1,
  - utility HSSFCGI is installed, and
  - service program HSSFCGI/XLPARSERJP is available.

  Note. When you specify a sheet name in parameter SHEETNAME, the value specified in parameter SHEET is ignored.
- **Sheet name (SHEETNAME)** - The name of the sheet to be processed. This value is case sensitive and must be specified within quotes (').
  When you specify the spreadsheet name in parameter SHEETNAME, the sheet sequence number in parameter SHEET is ignored.
  Parameter SHEETNAME requires
  - at least IBM i OS release V6R1,
  - utility HSSFCGI must be installed, and
  - service program HSSFCGI/XLPARSERJP must be available.
- **Target file (FILE)** - Qualified name of the physical file output from the process.
  If the physical files does not exist, and CRTFILE(*YES) specified, a DDS source member is generated and is used to create the physical file. The physical file is created with MAXMBRS(*NOMAX).
- **Target member (FILEMBR)** - Physical file member output from the process. If the member specified does not exist,
  - if the file is defined as MAXMBRS(*NOMAX), the member is added to the file
  - otherwise a program exception is generated.
- **Replace or add records (MBROPT)** - Specifies whether the new records replace or are added to the existing records. Select one of the following:
  - *REPLACE - The program clears the existing member and adds the new records.
  - *ADD - The program adds the new records to the end of the existing records.
- **(Re)create target file (CRTFILE)** - Select one of the following:
  1. *YES
     - if the target file does not yet exist, or

- if the target file exists already, but you want its record format layout be re-computed from the XLS/XLSX spreadsheet.

2. *NO
   if the target file already exists and you want to keep its record format layout that way it currently is. If you select *NO, you must be aware that possible format changes to the Excel XLS/XLSX spreadsheet may impair the ability to correctly load the data into the existing target file.

- **Source file (SRCFILE)** (only for CRTFILE(*YES)) - The source file to contain the DDS generated for the target file. If not yet existing, the source file is automatically generated.
- **Source member (SRCMBR)** (only for CRTFILE(*YES)) - The name of the source member to contain the DDS generated for the target file. You may use *FILE to mean the same name as the target file name in parameter FILE.
- **Record format name (RCDFMT)** - Record format name of the physical file to be created. This parameter may be used only when CRTFILE(*YES).
  Two options are available:
  a. Leave *AUTO* - The record format name is automatically generated.
  b. Enter your own record format name.
- **Record format field names (RCDFLDNMS)** - When CRTFILE(*YES), you may choose among three different ways to assign record format file names:
  a. *Define up to 100 field names in parameter RCDFLDNMS*
  b. *Leave RCDFLDNMS(*NONE) and pick up field names from spreadsheet column headings*, see parameters HLINES and FLDNAMES(*COLHDG).
  c. *Leave RCDFLDNMS(*NONE) and FLDNAMES(*DFT)*. In this case field names FLD1, FLD2, ... FLDn are automatically generated.
- **Header lines (HLINES)** - Number of header lines that should be ignored.
  Sometimes one or more initial rows of a sheet contain headers. Header columns must not be processed as they do not contain valid record data. This parameter allows to tell how many initial rows should be skipped.
- **Columns forced as numeric (FRCNUMCOLS)** - Columns containing number-format cells and text-format cells do result into a character-data fields in the generated database file record format.
  To force these columns to generate numeric-data fields in the generated database file record format, you must mention their numbers (1 for column A, etc.) in the entries of this parameter.
  Up to 50 such columns may be specified.
  Of course, if a specified column happens to contain just number-format cells, a numeric-data record format field is generated.
  A text-format cell, to be converted to a number must containg a string where
  1. the minus ("-") sign, if needed, is the first character
  2. thousand separators are not used
  3. the "decimal point", if needed, is either character (".") or character (",") according to the national language of the Excel spreadsheet.

  Valid examples:
  - English language spreadsheet: -8927.353 or 8927.36
  - Italian language spreadsheet: -8927,353 or 8927,36
- **Field names (FLDNAMES)** - This parameter is displayed only when CRTFILE(*YES) and the value of parameter HLINES (Header lines) is greater than zero.
  It establishes the naming rule for the record format fields of the physical file (defined in parameter FILE) to be created.
  Select one of the following naming rules:
  - *DFT* - Field names are named FLD followed by a sequence number
  - *COLHDG* - Field names are taken from the columns of the last hreading row, provided that they are valid field names.
- **Date columns (DATECOL)** - In Excel spreadsheets, a date is stored as a floating point number. This number represents the number of days elapsed since January 1, 1900. In order to have numeric values converted back to dates, do one of the following:
  - Type *NONE to avoid converting any numeric value to a date value.
    This is the recommended choice unless you know that the spreadsheet contains at least one date column.
  - Enter up to 50 column numbers to identify the columns where numeric values must be converted to date values.
    This is the recommended choice when you know that the spreadsheet contains at least one date column.
  - Type *AUTO to let the program establish which columns contain numeric values to be converted to date values.
    This choice may provide unexpected results on numeric columns and is suggested only for test purposes, in order to find out if any date columns exist in the spreadsheet.
- **Date format (DATEFMT)** - Date format (DATEFMT) - This parameter is made available when CRTFILE(*YES) and parameter DATECOL does not specify *NONE.
  It is used to establish the format of date fields (data type L) when creating the target database file (specified in parameter FILE). Select one of the available formats:

format	representation	
*ISO	yyyy-mm-dd	
*USA	mm/dd/yyyy	
*EUR	dd.mm.yyyy	
*JIS	yyyy-mm-dd	
*YMD	yy?mm?dd	where character ? is the date separator used at job level
*MDY	mm?dd?yy	where character ? is the date separator used at job level
*DMY	dd?mm?yy	where character ? is the date separator used at job level

- **Time columns (TIMECOL)** - In Excel spreadsheets, the internal representation of a time value (example: 16:48:56) is a number (example: 700648). This number measures the time in millionths of a day (example of computation: there are 84,600 seconds in 24 hours; time 16:48:56 is 69,536 seconds; therefore the Excel internal representation of time 16:48:56 is (69,536/84,600)*1,000,000=700648 .
  In order to have Excel time numeric values converted back to a conventional time format (example: hh.mm.ss), do one of the following:
  - Type *NONE to avoid converting any numeric value to a time value. This is the recommended choice unless you know that the spreadsheet contains at least one time column.
  - Enter up to 50 column numbers to identify the columns where numeric values must be converted to time values. This is the recommended choice when you know that the spreadsheet contains at least one time column.
- **Time format (TIMEFMT)** - This parameter is made available when CRTFILE(*YES) and parameter TTMECOL does not specify *NONE.
  It is used to establish the format of time fields (data type T) when ctreating the target database file (specified in parameter FILE). Select one of the available formats:
  - *HMS - Time representation is hh:mm:ss
  - *ISO - Time representation is hh.mm.ss
  - *USA - Time representation is hh:mm AM or hh:mm PM
  - *EUR - Time representation is hh.mm.ss
  - *JIS - Time representation is hh:mm:ss
- **Run in a sync submitted job (SBMJOB)** - If you specify SBMJOB(*YES), the XLSTABLE command is executed in a separate submitted job. The current job waits until the submitted job completes, then it resumes execution.
  This can be useful in two cases:
  - you do not want XLSTABLE to start a Java Virtual Machine in the current job, OR
  - you do not want XLSTABLE to run in the current job because it may conflict with a Java Virtual Machine already active in this job.

  Should you absolutely need to run this command in the current job, though the JVM has already been started by a previous Java application, read about command **JVMSTARTUP** in Appendix **JVMSTARTUP**.
- **Job queue** - If you specify SBMJOB(*YES), this is the job queue where the synchronized job is submitted.
- **Type of failure message (FAILMSG)** - When you specify SBMJOB(*YES), the XLSTABLE command is executed in a synchronized submitted job.
  It may however happen that the XLSTABLE command fails during its execution. In such a case a message is sent to the current program in the current job. You may choose whether this message would be an *ESCAPE or a *DIAG (diagnostic) one.
- **Display target file (DSPFILE)** - whether the target file should be displayed at the end of the process.

3- An XLSTABLE failure

In some cases, XLSTABLE command may send the escape message *"No significant cells detected in this sheet"*.
This may happen when a spreadsheet was generated from some software tool and some properties are missing.
You can manually fix the Excel workbook on a PC in this way:

- Open the spreadsheet with Excel (2003 or subsequent)
- Save the spreadsheet; Excel will add the missing properties.

Should you need that be done in an IBM i procedure of yours, you may bypass this problem by specifying in parameter SHEETNAME the name of the spreadsheet you want to process.
Of course, utility HSSFCGI with service program XLPARSERJP is required.

4- The process of command XLSTABLE

It may be helpful to understand what goes on under the covers:

a. Some initial checks are done.
b. Command XLPARSE2/XLSCONVERT is executed, thus creating file QTEMP/XLSOUTF from the Excel XLS/XLSX spreadsheet.
   Command XLPARSE2/XLSCONVERT really reads all the spreadsheet cells, using the Scott Klements XLPARSER4 utility (Scott's *xlparse* Java class and *POI* Java classes).
   As a result from this process, file QTEMP/XLSOUTF contains a database representation of all the spreadsheets in the XLS/XLSX workbook.
c. If CRTFILE(*YES) specified
   i. Subprocedure *xlsgetcell* is used to receive from file QTEMP/XLSOUTF all the cells from the requested sheet (see parameter SHEET).
      At the end of this process, the data type and the size of each column in the sheet are known, a DDS member is generated (*any previous version of the target file member is replaced by the new one (**!!!**)*) and the target file is created.
d. If CRTFILE(*NO) specified
   i. DDS are not regenerated, and the target file is not rebuilt.
e. Cells data are used to build, in a temporary stream file, an "imported file" script. The "imported file" script is used by command CPYFRMIPMF to upload the target file.
f. Next, the load of records to the target file takes place:
   - The "imported file" script is used by command CPYFRMIPMF to upload the target file.
g. Last, if DSPFILE(*YES), the contents of the target file are displayed.
   A far better insight of the target file can be displayed by another open-source *Easy400.net* utility: **CGI_WRKDBF**.

5- Spreadsheets containing formulas

Values resulting from formulas may have conversion problems. A simple way to bypass these problems is documented in **this topic**.

6- About the target file

In the record format of the target file, each field corresponds to a column in the XLS/XLSX sheet.
There are four type of fields:

- **Numeric fields** - A spreadsheet column containing only numeric data (formulas are considered be numeric) generates a numeric field.
  A numeric field is always generated as zoned, 30 digits, of which 6 are decimal digits.
- **Character fields** - A spreadsheet column not containing text data generates a character field.
  Character fields have a critical factor, their sizes. The size of a character field is computed from the largest cell data in a sheet column.
  This means that, if you process via command XLSTABLE an XLS/XLSX sheet similar (same type and number of columns) the resulting re-created target file record format may result different simply because the largest character cell in a given column has a size different from the largest character cell in the same column of the previous similar sheet.
- **Date fields** - These fields are generated from the spreadsheet columns listed in parameter DATECOL. All generated date fields share the same format, the one specified in parameter DATEFMT.
- **Time fields** - These fields are generated from the spreadsheet columns listed in parameter TIMECOL. All generated time fields share the same format, the one specified in parameter TIMEFMT.
- **Special conversion cases**
  - Case 1 - Generic format cell containing a positive number, example *1234,56*.
    This cell is converted to a decimal packed(30 6) field.
  - Case 2 - Generic format cell containing a negative number number with sign minus as first character, example *-1234,56*.
    This cell is converted to a decimal packed(30 6) field.
  - Case 3 - Generic format cell containing a negative number number with sign minus as last character, example *1234,56-*.
    This cell is converted to a text field.
  - Case 4 - Columns containing both numeric format and text format cells. All cells are converted to text fields. Numeric data are right aligned, text data are left aligned.
    However, if you mention these columns in XLSTABLE command parameter FRCNUMCOLS, all cells are converted to a numeric field.
  - Case 5 - Columns containing both numeric format and generic format cells. All cells are converted to text fields. Generic data and text data are left aligned.
    However, if you mention these columns in XLSTABLE command parameter FRCNUMCOLS, cells may be converted to a numeric field.

7- (Re)Creating the target file

Use parameter CRTFILE(*YES) to re-create the target file.
When re-creating the target file, you have some options regarding

A. The record format name.
   Two choices are available:
   1. Leaving the default value RCDFMT(*AUTO) creates a record format name made of the first (max 7) characters of the file name, followed by "RCD".
   2. You may specify your record format name in parameter RCDFMT.
B. The record format field names.
   Three choices are available:
   1. Up to 100 field names can be specified in parameter RCDFLDNMS.
   2. If the spreadsheet has column headings that can be used as field labels, you may obtain such field names by
      - Leaving RCDFLDNMS(*NONE), *and*
      - specifying for parameter HLINES a number of heading rows higher than zero, *and*
      - specifying FLDNAMES(*COLHDG)

      In such a case, field names are taken from the columng headings of the last heading rows specified in parameter HLINES.
   3. If you leave RCDFLDNMS(*NONE) and FLDNAMES(*DFT), default field names FLD1, FLD2, ... FLDn are automatically generated.

Note 1. The following field data types can be generated when CRTFILE(*YES):

- *Character* (data type A), any length.
- *Zoned numeric* (data type Z), always 30 digits, 6 decimal positions.
- *Date* (data type L), with the format specified in parameter DATEFMT.
- *Time* (data type T), with the format specified in parameter TIMEFMT.

Note 2. When CRTFILE(*NO) character, numeric, date and time fields are mapped to the format of the corresponding fields in the receiving target file.
Note 3. Assigning your own field names may be of great help in copying a generated target physical file to an existing database file with identical field names, with the same data type, but different field length and/or precision. That is easily done using command CPYF ... FMTOPT(*MAP).

8- Suggestions for a recursive use of the same XLS/XLSX spreadsheet model

In most cases you will be planning for a recurrent use of command XLSTABLE for some XLS/XLSX spreadsheet models.
What you must expect is that a worksheet of a given model is uploaded via XLSTABLE to its appropriate target file member, so that the target file can then be processed by a given program of yours, without the need of re-compile it to match the record format of the target file.
In order for this to happen without troubles, you must make sure that

1. A given worksheet has always the same number of columns
2. All columns in the spreadsheet maintain their data type (Character/Numeric; however, see NOTE 1 below.)
3. The character fields in the record format of the target file are large enough to account for the largest case. This can simply be done - once for ever - in the following way:
   i. review the DDS:
      1. You CAN assign all fields the names that best fit your process.
      2. You CAN add field level keywords.
      3. You CAN add keys.
      4. You are strongly recommended to increase the lengths of the character fields in order to fit the largest case.
      5. You MUST NOT change data type, digits and decimal positions of the numeric fields. They must always specify 30S 6 . See however NOTE 1 below.
      6. You MUST NOT resequence fields.
   ii. recreate the target file
   iii. run command *clrlib qtemp*
   iv. run command *xlstable ... crtfile(*no) ...*
4. **Make sure that CRTFILE(*NO) is specified in command XLSTABLE.** *If you specify* CRTFILE(*YES) *your DDS member is cleared and new DDS are generated from the program **!!!***

NOTE 1- There might be cases where some columns of a spreadsheet may some times show up as numeric and some other time show up as character. In such cases, the best thing to do is to

1. redefine via DDS the corresponding target database fields as 30 char fields
2. run command *clrlib qtemp*
3. run command *xlstable ... crtfile(*no) ...*

12/13/2019 14:58:58

Some XLPARSE2 CGI programs

If you have installed

- library CGIDEV2 (ILE-RPG CGI Development Toolkit, the version available from www.easy400.net)
- the **XLPARSE2 HTTP instance**

then you may run the following sample CGI programs:

1- Upload a PC XLS/XLSX file and run command XLSTABLE or XLSTABLE2 from it.

This is program XLPARSE2/UPLANDRUN. It can be executed by entering the following URL in the location bar of your browser:
`http://yourSystemItcpAddress:8050/xlparse2p/uplandrun.pgm`
where

- *yourSystemItcpAddress* is the TCP address of your IBM System **i**

The initial screen requires the following input:

1. **Path and name** of your PC file to be uploaded (a browse button is available for that)
2. **Some XLSTABLE / XLSTABLE2 command parameters:**
   i. the *number of the sheet to be processed*
   ii. the *number of heading lines to be skipped*
   iii. the *date format*
   iv. the *numbers of the columns containing date values*
   v. the *time format*
   vi. the *numbers of the columns containing time values*
   vii. the name of the target database *file*, the name of its *library* and the name of its *member*
   viii. whether the target database file should be *(re)created*
   ix. whether records should be *added* to or *replaced* in the target database file member.

A batch job is submitted to execute the **XLSTABLE / XLSTABLE2** command.
You are then enabled to inquiry about the status of the submitted batch job and to display its joblog if something went wrong.

12/17/2019 21:30:47

JVMSTARTUP

Programmers used to work with library lists may have some problems in understanding how the JVM (Java Virtual Machine) works in retrieving Java "programs".
The following may help in understanding what goes on and how a developer can master it.

A. **About JAR files**

1. Java code resides in Java classes. Stream files with extension JAR are collections of Java classes. In other words, a JAR file can be understood as a library of programs.
2. IFS directory `/QIBM/userdata/java400/ext` may contain several Java classes. These Java classes make up something like a Default Library List for Java classes.
3. The `CLASSPATH` environment variable is a list of JAR files. It can therefore be understood as a Library List for Java classes.
4. The environment variable CLASSPATH is mandatory when some of the Java code to be executed is in a directory other than `/QIBM/userdata/java400/ext` and must be set before that the JVM is started. Once started, the JVM cannot be restarted.

B. **JAR files in HSSFCGI and XLPARSE2**

- HSSFCGI jar files are in subdirectories of IFS directory /HSSFCGI/java
- XLPARSE2 jar files are in subdirectories of IFS directory /XLPARSE2/java
- HSSFCGI2 and XLPARSE2 jar files are the same
- HSSFCGI and XLSPARSE2 programs calling Java classes use each a separate CLASSPATH string, mentioning their jar files. However, as the jar files are the same for both the utilities, there two CLASSPATHs are interchangeable.
- Both HSSFCGI and XLPARSE2 feature a command, `SETDFTJARS`, that copies their jar files (identical for both the tools) to IFS directory `/QIBM/userdata/java400/ext`. This could be of some help when CLASSPATHs look like having problems. Both Both HSSFCGI and XLPARSE2 feature another command, `RMVDFTJARS`, that removes thier jar files from IFS directory `/QIBM/userdata/java400/ext`.

C. **Jobs running multiple Java products**

1. Jobs running both HSSFCGI programs and XLPARSE2 programs have no JVM problems.
2. Jobs running other Java products plus HSSFCGI and/or XLPARSE2 may have JVM troubles.
   Such troubles may by solved by running command `HSSFCGI/JVMSTARTUP` or `XLPARSE2/JVMSTARTUP` at the beginning of the job.
   By doing this, JVM properties are set in way that enables execution of intermixed HSSFCGI, XLPARSE2 and other Java products.

D. **In case of unsolved JVM problems**

If a mix of HSSFCGI/XLPARSE2 and other Java products fails to run within a single job, the only way out is to submit to a batch job the execution of HSSFCGI or XLPARSE2 commands.

- XLPARSE2 commands feature a SBMJOB parameter. That parameter allows to submit a command for execution in a separate batch job. The submitted job is synchronized with the submitting job: that is, the submittiong job resumes execution after completion of the submitted job.
- Also HSSFCGI commands feature a SBMJOB parameter. However in this case the submitted job is not synchronized with the submitting one.

12/31/2020 17:35:02

Questions and Answers

Answers to questions raised by some XLPARSE2 users.

1. **Q:** *When running command XLPARSE2/XLSTABLE or XLPARSE2/XLSCONVERT, the following error is displayed:*
   a. *Escape program message:*
      Program XLSCONVERT failed, see previous messages.
   b. *Previous diagnostic message:*
      Message . . . . : Error occurred while parsing spreadsheet after cell (0,0) in *Unknown*.
      Last cell processed was row 0, column 0 in sheet *Unknown*. Joblog message Java exception received when calling Java method.
      Error occurred while parsing spreadsheet after cell (0,0) in *Unknown*. Message ID . . . . . : RNX0301 Severity . . . . . : 50
      Message type . . . . . : Escape Date sent . . . . . : xx/xx/xx Time sent . . . . . : xx:xx:xx
   c. *Additional message information (via F1):*
      Message . . . . : Java exception received when calling Java method.
      Cause . . . . . : RPG procedure XLPARSE_WO in program XLPARSE2/XLPARSER4 received Java exception
      "java.lang.NoClassDefFoundError:
      com.iseriesnetwork.clubtech.xlparse.ParseSheet" when calling method
      "parse201003" with signature "([B)V" in class
      "com.iseriesnetwork.clubtech.xlparse.ParseSheet".
      Recovery . . . : Contact the person responsible for program maintenance to
      determine the cause of the problem.

   What can I do About it?
   A: This is a very common case. The error is that a Java class, needed by service program XLPARSE2/XLPARSER4, cannot be found.
   The way a Java class is retrieved by the Java Virtual Machine (JVM) is as follow:
      I. A search is made in IFS directory */QIBM/UserData/Java400/ext*. This directory works like a system library list.
      II. If the search is unsuccessful, then the search goes through the IFS directories mentioned in the environment variable CLASSPATH, which works like a job library list.

   One major problem is that once the JVM has been started in a job, the current CLASSPATH is stored in its data, and cannot be changed even by changing the job CLASSPATH. This restriction has the following consequences:
      i. If the job has to process two or more applications using different classes, and each application sets up a different CLASSPATH, only the first application will work, the subsequent ones will fail in retrieving their Java classes.
      ii. To overcome this problem, Java application developers often do install their classes in the IFS directory */QIBM/UserData/Java400/ext*. In this way, other problems may raise when different applications have classes with the same name, and class methods are different. This may happen when two applications, built over a given Java package (example: POI), require a different release of that package. Because of this, Java classes used by XLPARSE2 are not in IFS directory */QIBM/UserData/Java400/ext*, they are in IFS directory */hssfcgi/java*, and XLPARSE2 sets up the appropriate CLASSPATH environment variable before calling the JVM.

   Once the above is understood, the ways to solve the NoClassDefFoundError are quite obvious:
      A. Make sure that IFS directory */QIBM/UserData/Java400/ext* does not contain any POI-related and any XLPARSE-related objects.
      B. Run command WRKENVVAR(*SYS) and, if system level CLASSPATH environment variable is listed, remove it. Java applications should instead create their own CLASSPATH env.var's.
      C. If your job must execute some other Java applications beside running XLPARSE2 commands XLSTABLE and/or XLSCONVERT, in these XLPARSE2 commands specify the parameter SBMJOB(*YES). This parameter submits the execution of the command in a separate batch job and the submitting job waits for the completion of the submitted one.

   See also page **JVMSTARTUP**.

2. **Q:** *In processing some Excel spreadsheets with command XLSCONVERT or XLSTABLE, error message* ***"No significant cells detected in this sheet"*** *is displayed. How can I fix that?*
   A: See topic **An XLSCONVERT failure**.

3. **Q:** *Do commands XLSCONVERT and XLSTABLE work OK with cells containing data from a selection list?*
   A: Yes, the selected value is correctly received, provided it is a number, a test, or a formula.

4. **Q:** *How to retrieve the name of a given cell?*
   A: Cell names are made of a number representing its row and a letter representing its column. Example: a cell in row 10 and column 5 has name 10E.
   In XLPARSE2 service program XLPSARSER4, procedure xlparse_workbook(), while returning the row number and the column number of a cell, does not make up its name.
   However, service program XLPARSER4 procedure *GetCellName()* (see HSSFCGI/QRPGLESRC member XLPROC) returns the name of a cell in a given row and in a given column. Program HSSFCGI/DOCELLNAME (see HSSFCGI/QRPGLESRC member DOCELLNAME) provides an example of using such a procedure.

5. **Q:** *Does command XLSTABLE (command XLSCONVERT) provide the cell value resulting from a formula?*
   A: No. That is computed by MS Excel, not by XLPARSE2. See **Spreadsheets containing formulas**.

6. **Q:** *How to print from IBM i an Excel Spreadsheet residing in a IFS directory?*
   A: Write a program getting a cell at a time by using XLPARSE2/XLPARSER4 service program procedure *XlsGetCell()*.

7. **Q:** *Is XLPARSE2 able to process OpenOffice spreadsheets?*
   A: No.

8. **Q:** *Are XLPARSE2 commands XLSCONVERT and XLSTABLE able to retrieve the value (true/false) of a logical cell?*
   A: No.

Java product 5722JV1 / 5770JV1

Last modified on 09/06/2022 20:00:44

Option	Feature	Description	Java version	Library	IFS directory	Java Classic	IBM Technology for Java	V5R3	V5R4	V6R1	V7R1	V7R2	V7R3	V7R4	V7R5
*BASE	5050+lng	IBM Developer Kit for Java		QJAVA				x	x	x	x	x	x	x	x
5	5105	Java Developer Kit 1.3	1.3	QJAVA	/QIBM/ProdData/Java400/jdk13			x							
6	5106	Java Developer Kit 1.4	1.4	QJAVA	/QIBM/ProdData/Java400/jdk14	x		x	x	(x)					
7	5107	Java Developer Kit 5.0	1.5	QJAVA	/QIBM/ProdData/Java400/jdk15	x		x	x	(x)					
8	5108	J2SE 5.0 32 bit	1.5	QJVM50	/QOpenSys/QIBM/ProdData /JavaVM/jdk50/32bit		x		x	x	x				
9	5109	J2SE 5.0 64 bit	1.5	QJVM5064	/QOpenSys/QIBM/ProdData /JavaVM/jdk50/64bit		x			x	x				
10	5110	Java SE Development Kit 6	1.6	QJAVA	/QIBM/ProdData/Java400/jdk6	x			x	x					
11	5111	Java SE 6 32 bit	1.6	QJVM6032	/QOpenSys/QIBM/ProdData /JavaVM/jdk60/32bit		x		x	x	x	x			
12	5112	Java SE 6 64 bit	1.6	QJVM6064	/QOpenSys/QIBM/ProdData /JavaVM/jdk60/64bit		x			x	x	x			
13	5113	Java Developer Kit (J2SE 1.4.2 64-bit)	1.4	QJVM1464	/QOpenSys/QIBM/ProdData /JavaVM/jdk14/64bit		x			x	x				
14	5114	Java SE 7 32 bit	1.7	QJVM7032	/QOpenSys/QIBM/ProdData /JavaVM/jdk70/32bit		x				x	x	x		
15	5115	Java SE 7 64 bit	1.7	QJVM7064	/QOpenSys/QIBM/ProdData /JavaVM/jdk70/64bit		x				x	x	x		
16	5116	Java SE 8 32 bit	1.8	QJVM8032	/QOpenSys/QIBM/ProdData /JavaVM/jdk80/32bit		x					(x)	x	x	x
17	5117	Java SE 8 64 bit	1.8	QJVM8064	/QOpenSys/QIBM/ProdData /JavaVM/jdk80/64bit		x					(x)	x	x	x

Reference: **Support Java Versions by Operating System Release**.

(x): not compatible with HSSFCGI and XLPARSE2