R.P.G. JSON & XML

David Andruchuk

Sr. Architect

Computer Systems Design Associates, Inc.

What can *i* do…..*i* can do JSON & XML

Read, Prepare and Generate

What are we covering today?

Brief overview of our target docments

- JSON (JavaScript Object Notation)
- XML (Extensible Markup Language)

Coding our CMD/PNLGRP/CLLE/SQLRPGLE

- RPGJSON – Coding our Command
- RPGJSONP – Coding our Panel Group (for Help)
- RPGJSONV – Coding our Command Validation CLLE
- RPGJSONC – Coding our Processing CLLE
- RPGJSONR – Coding our SQLRPGLE program

Read, Prepare and Generate

With the always growing need to interface with applications in our intranet or over the internet, we can create a generic solution to convert any DB2 table or view to a JSON or XML document using a command interface, CLLE driver and SQL RPG.

JSON and XML are lightweight, text-based, human-readable format, language-independent data interchange formats that have become increasingly popular that are currently supported by many different programming language APIs.

JSON

What is Json?

JSON is short for JavaScript Object Notation, is an open standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs. It is used primarily to transmit data between a server and web application, as an alternative to XML.

WHY DOES JSON MATTER?

With the rise of AJAX-powered sites, it's becoming more and more important for sites to be able to load data quickly and asynchronously, or in the background without delaying page rendering. It has been the preferred format because it is much more **lightweight.**

Sample:


```
{ "Root_Element": {
  "Table_Rows": [
   {"EMPNO": "000010",
    "FIRSTNME": "CHRISTINE",
    "MIDINIT": "I",
    "LASTNAME": "HAAS",
    "WORKDEPT": "A00",
    "PHONENO": "3978",
    "HIREDATE": "01/01/65",
    "JOB": "PRES",
    "EDLEVEL": "18",
    "SEX": "F",
    "BIRTHDATE": "01/01/40",
    "SALARY": "52750.00",
    "BONUS": "1000.00",
    "COMM": "4220.00"
   }
 ] }}
```

XML

What is XML?

XML is short for Extensible Markup Language, a markup language that defines a set of rules for encoding documents in a format which is both human and machine-readable.

WHY DOES XML MATTER?

One of the most time-consuming challenges for developers is to exchange data between incompatible systems over the Internet. Exchanging data as XML greatly reduces this complexity, since the data can be read by different incompatible applications.

Sample:

```
<?xml version="1.0" encoding="UTF-8"?>
<Root_Element>
 <Table_Rows>
  <EMPNO>000010</EMPNO>
  <FIRSTNME>CHRISTINE</FIRSTNME>
  <MIDINIT>I</MIDINIT>
  <LASTNAME>HAAS</LASTNAME>
  <WORKDEPT>A00</WORKDEPT>
  <PHONENO>3978</PHONENO>
  <HIREDATE>01/01/65</HIREDATE>
  <JOB>PRES</JOB>
  <EDLEVEL>18</EDLEVEL>
  <SEX>F</SEX>
  <BIRTHDATE>01/01/40</BIRTHDATE>
  <SALARY>52750.00</SALARY>
  <BONUS>1000.00</BONUS>
  <COMM>4220.00</COMM>
 </Table_Rows>
</Root_Element>
```

JSON vs XML

Side by Side comparison

JSON

Sample:

```
{ "Root_Element": {
  "Table_Rows": [
   {"EMPNO": "000010",
    "FIRSTNME": "CHRISTINE",
    "MIDINIT": "I",
    "LASTNAME": "HAAS",
    "WORKDEPT": "A00",
    "PHONENO": "3978",
    "HIREDATE": "01/01/65",
    "JOB": "PRES",
    "EDLEVEL": "18",
    "SEX": "F",
    "BIRTHDATE": "01/01/40",
    "SALARY": "52750.00",
    "BONUS": "1000.00",
    "COMM": "4220.00"
   }
 ] }}
```

XML

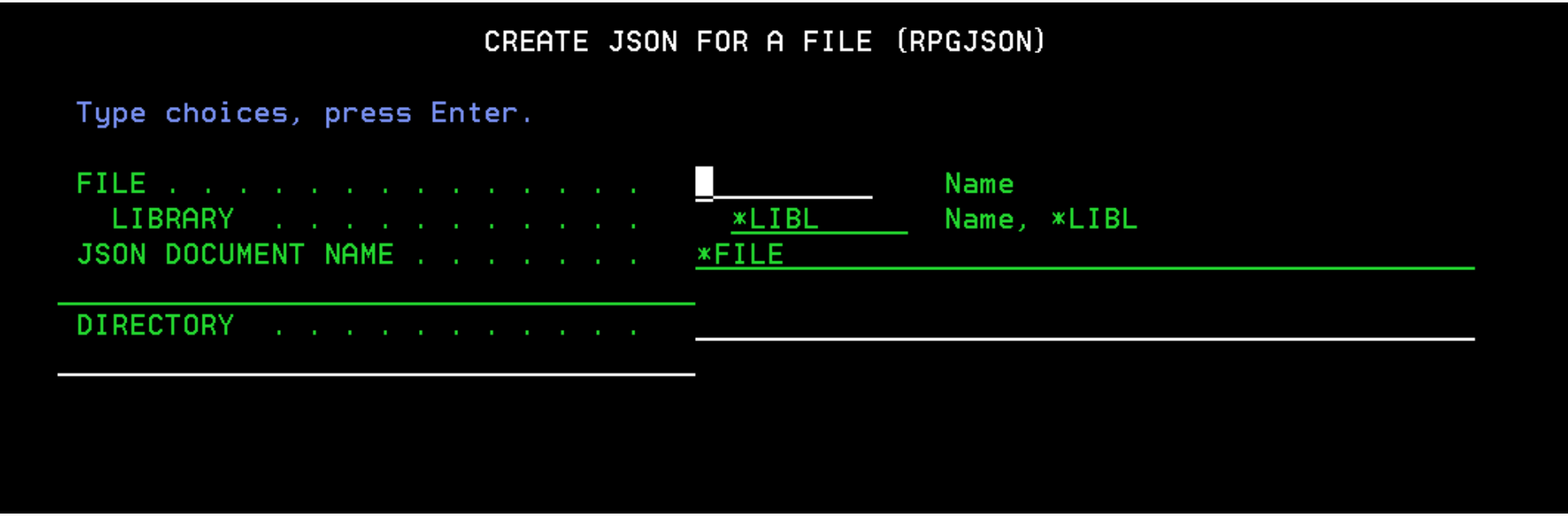
Sample:

```
<?xml version="1.0" encoding="UTF-8"?>
<Root_Element>
 <Table_Rows>
  <EMPNO>000010</EMPNO>
  <FIRSTNME>CHRISTINE</FIRSTNME>
  <MIDINIT>I</MIDINIT>
  <LASTNAME>HAAS</LASTNAME>
  <WORKDEPT>A00</WORKDEPT>
  <PHONENO>3978</PHONENO>
  <HIREDATE>01/01/65</HIREDATE>
  <JOB>PRES</JOB>
  <EDLEVEL>18</EDLEVEL>
  <SEX>F</SEX>
  <BIRTHDATE>01/01/40</BIRTHDATE>
  <SALARY>52750.00</SALARY>
  <BONUS>1000.00</BONUS>
  <COMM>4220.00</COMM>
 </Table_Rows>
</Root_Element>
```

Coding our Command

```
/*-------------------------------------------------------------------------------------------------*/
/* Create Instructions:                                                                            */
/*                                                                                                 */
/* CRTCMD CMD(RPGJSON) PGM(*LIBL/RPGJSONC) SRCFILE(QCMDSRC)  */
/*        VLDCKR(RPGJSONV) HLPPNLGRP(RPGJSONP) HLPID(*CMD)          */
/*-------------------------------------------------------------------------------------------------*/
          CMD        PROMPT('Create JSON for a File')

          PARM       KWD(FILE) TYPE(F1) MIN(1) PROMPT('File' 1)

 F1:      QUAL       TYPE(*NAME) LEN(10)
          QUAL       TYPE(*NAME) LEN(10) DFT(*LIBL) SPCVAL((*LIBL)) +
                       PROMPT('Library')

          PARM       KWD(DIRECTORY) TYPE(*CHAR) MIN(1) LEN(80) +
                       CASE(*MIXED) PROMPT('Directory' 3)

          PARM       KWD(DOCUMENT) TYPE(*CHAR) LEN(80) DFT(*FILE) +
                       CASE(*MIXED) PROMPT('JSON Document name' 2)
```

Coding our Command



Coding our Help Panel

```
.*-------------------------------------------------------------------------*
.* Create Instructions:                                                    *
.*                                                                         *
.* CRTPNLGRP PNLGRP(RPGJSONP) SRCFILE(QPNLSRC)  *
.*-------------------------------------------------------------------------*
:pnlgrp.

.* Command help text
:help name='RPGJSON'.
RPGJSON - Help
:P.
The RPGJSON utility creates a JSON document for the specified file. The
JSON document created uses column names of the table for the JSON Element
names.
:ehelp.

.* Command parameters help text
:help name='RPGJSON/FILE'.
File (FILE) - Help
:xh3.File (FILE)
:P.
Specifies the file you wish to generate a JSON document for.
:P.
The file parameter is a required parameter and must already exist
on disk.
```

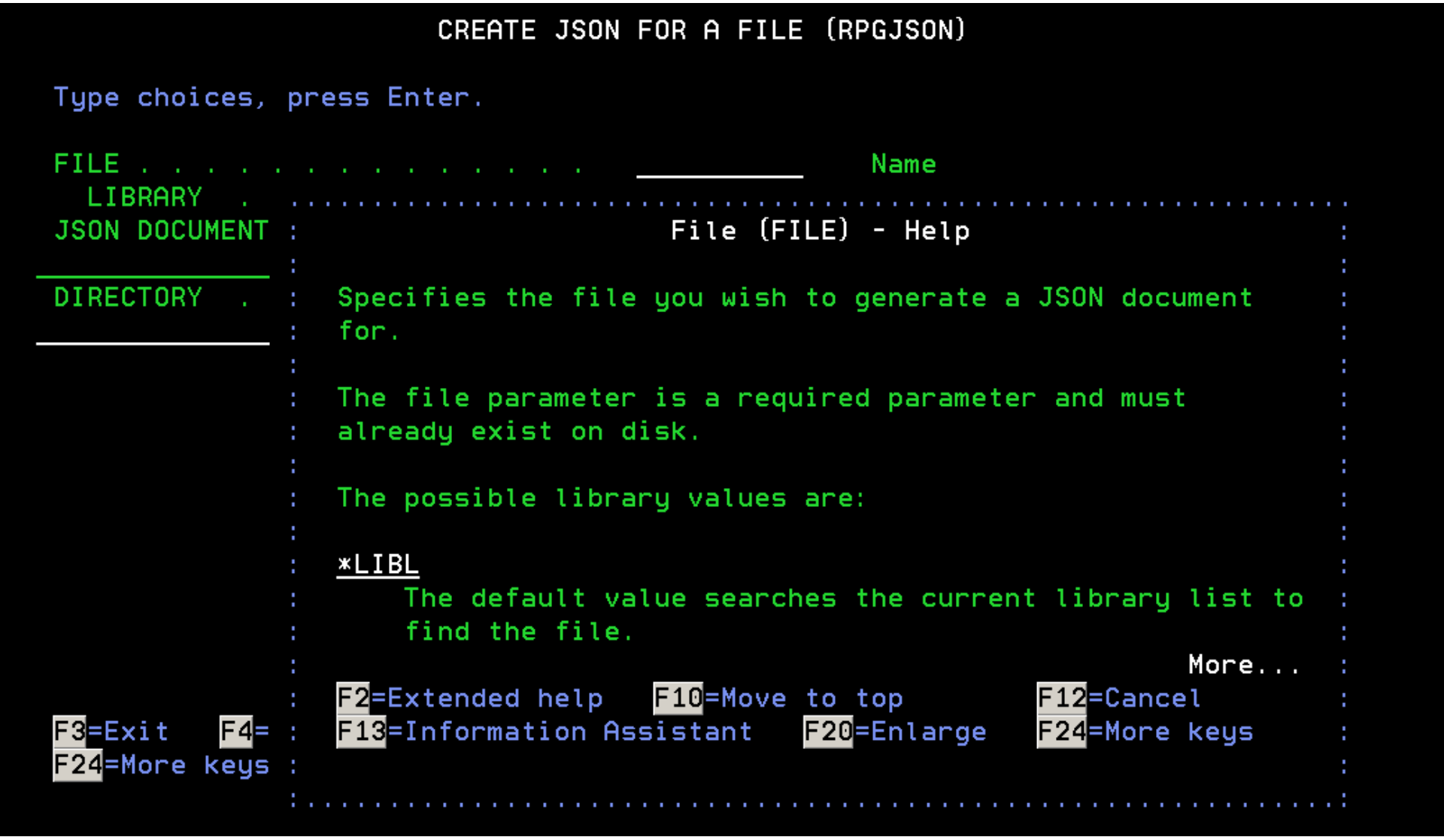
Coding our Help Panel-continued

```
:P.
The possible library values are:
:PARML.
:PT.:PK DEF.*LIBL:EPK.
:PD.
The default value searches the current library list to find the
file.
:PT.:PV.library-name:EPV.
:PD.
Key the library name that contains the file.
:eparml.
:ehelp.

:help name='RPGJSON/DOCUMENT'.
Document (DOCUMENT) - Help
:xh3.Document (DOCUMENT)
:P.
Specifies the JSON document name to be created.
:P.
The possible document values are:
:PARML.
:PT.:PK DEF.*FILE:EPK.
:PD.
The default value uses the name of the file specified as the JSON
document name with an .JSON appended to it.
```

Coding our Help Panel-continued

```
:PT.:PV.document-name:EPV.
:PD.
Key the document name you wish to create.
:eparml.
:ehelp.

:help name='RPGJSON/DIRECTORY'.
Directory (DIRECTORY) - Help
:xh3.Directory (DIRECTORY)
:P.
Specifies the directory in the IFS the JSON document will be
generated in.
:P.
The directory is a required parameter and must exist already exist.
:ehelp.
:epnlgrp.
```

Coding our Help Panel-continued



Coding our Help Panel-continued

```
                    CREATE JSON FOR A FILE (RPGJSON)
                                RPGJSON - Help
  The RPGJSON utility creates a JSON document for the specified file.
  The JSON document created uses column names of the table for the
  JSON Element names.

File (FILE)

  Specifies the file you wish to generate a JSON document for.

  The file parameter is a required parameter and must already exist on
  disk.

  The possible library values are:

  *LIBL
      The default value searches the current library list to find the
                                                                More...
F3=Exit help   F10=Move to top   F12=Cancel   F13=Information Assistant
F14=Print help
Help information is incomplete.
```

Coding our Validation CLLE

```
/*-------------------------------------------------------------------------------------*/
/* Create Instructions:                                                                */
/*                                                                                     */
/* CRTBNDCL PGM(RPGJSONV) SRCFILE(QCLSRC)                                              */
/*-------------------------------------------------------------------------------------*/
            PGM       PARM(&FILE &DIRECTORY &DOCUMENT)

/* program variables                                        */
            DCL       VAR(&FILE) TYPE(*CHAR) LEN(20)
            DCL       VAR(&DOCUMENT) TYPE(*CHAR) LEN(80)
            DCL       VAR(&DIRECTORY) TYPE(*CHAR) LEN(80)

            DCL       VAR(&BUF)      TYPE(*CHAR) LEN(4096)
            DCL       VAR(&EMSG)     TYPE(*CHAR) LEN(100)
            DCL       VAR(&EOSSTMF)  TYPE(*CHAR) LEN(256)
            DCL       VAR(&ERROR)    TYPE(*LGL) LEN(1) VALUE('0')
            DCL       VAR(&FILEN)    TYPE(*CHAR) LEN(10)
            DCL       VAR(&LIBRARY)  TYPE(*CHAR) LEN(10)
            DCL       VAR(&NULL)     TYPE(*CHAR) LEN(1)  VALUE(X'00')
            DCL       VAR(&RTNVALINT) TYPE(*CHAR) LEN(4)
            DCL       VAR(&RTNVAL)   TYPE(*CHAR) LEN(2)

/* set work variable(s) from command input                  */
            CHGVAR    VAR(&FILEN) VALUE(%SST(&FILE 1 10))
            CHGVAR    VAR(&LIBRARY) VALUE(%SST(&FILE 11 10))
            CHGVAR    VAR(&EOSSTMF) VALUE(&DIRECTORY |< &NULL)
```

Coding our Validation CLLE-continued

```
/* validate file name                                    */
          CHKOBJ     OBJ(&LIBRARY/&FILEN) OBJTYPE(*FILE)

          MONMSG     MSGID(CPF9801) EXEC(DO)
          CHGVAR     VAR(&EMSG) VALUE('File ' *CAT &FILEN *TCAT ' +
                       not found in Library ' *CAT &LIBRARY)

          SNDPGMMSG  MSGID(CPD0006) MSGF(QCPFMSG) +
                       MSGDTA('0000' *CAT &EMSG) MSGTYPE(*DIAG)

          CHGVAR     VAR(&ERROR) VALUE('1')
          ENDDO

/* validate input directory                              */
          CALLPRC    PRC('stat') PARM(&EOSSTMF &BUF) +
                       RTNVAL(%BIN(&RTNVALINT 1 4))

          CHGVAR     VAR(&RTNVAL) VALUE(%BIN(&RTNVALINT))
          CHGVAR     VAR(&EMSG) VALUE('Directory specified ' *cat   +
                       &directory *Tcat ' not found')

          IF         COND(&RTNVAL *NE '00') THEN(DO)
          SNDPGMMSG  MSGID(CPD0006) MSGF(QCPFMSG) +
                       MSGDTA('0000' *CAT &EMSG) MSGTYPE(*DIAG)

          CHGVAR     VAR(&ERROR) VALUE('1')
          ENDDO
```

Coding our Validation CLLE-continued

```
/* check if error occurred                                    */
            IF        COND(&ERROR) THEN( +
            SNDPGMMSG  MSGID(CPF0002) MSGF(QCPFMSG) MSGTYPE(*ESCAPE))

            ENDPGM
```

Coding our Processing CLLE

```
/*-------------------------------------------------------------------------------------*/
/* Create Instructions:                                                                */
/*                                                                                     */
/* CRTBNDCL PGM(RPGJSONC) SRCFILE(QCLSRC)                                              */
/*-------------------------------------------------------------------------------------*/
            PGM        PARM(&FILE &DIRECTORY &DOCUMENT)

/* program variables                                          */
            DCL        VAR(&FILE) TYPE(*CHAR) LEN(20)
            DCL        VAR(&DOCUMENT) TYPE(*CHAR) LEN(80)
            DCL        VAR(&DIRECTORY) TYPE(*CHAR) LEN(80)

            DCL        VAR(&EOSSTMF)   TYPE(*CHAR) LEN(256)
            DCL        VAR(&FILEN)     TYPE(*CHAR) LEN(10)
            DCL        VAR(&LIBRARY)   TYPE(*CHAR) LEN(10)
            DCL        VAR(&NULL)      TYPE(*CHAR) LEN(1)   VALUE(X'00')
            DCL        VAR(&PATLEN)    TYPE(*DEC)  LEN(3) VALUE(2)
            DCL        VAR(&PATTERN)   TYPE(*CHAR) LEN(999) VALUE('/ ')
            DCL        VAR(&RETURN)    TYPE(*DEC)  LEN(3)
            DCL        VAR(&STRLEN)    TYPE(*DEC)  LEN(3) VALUE(80)
            DCL        VAR(&STRPOS)    TYPE(*DEC)  LEN(3) VALUE(1)
            DCL        VAR(&TRANSLATE) TYPE(*LGL)  VALUE('0')
            DCL        VAR(&TRIM)      TYPE(*LGL)  VALUE('0')
            DCL        VAR(&WILDCARD)  TYPE(*CHAR) LEN(1)

/* set work variable(s) from command input                    */
            CHGVAR     VAR(&FILEN) VALUE(%SST(&FILE 1 10))
            CHGVAR     VAR(&LIBRARY) VALUE(%SST(&FILE 11 10))
```

Coding our Processing CLLE-continued

```
            CHGVAR     VAR(&EOSSTMF) VALUE(&DIRECTORY |< &NULL)

/* check for default of *LIBL in file name                          */
/* ·If *LIBL, retrieve file library location name                   */
            IF         COND(&library *eq '*LIBL') then( +
            RTVOBJD    OBJ(&filen) OBJTYPE(*FILE) RTNLIB(&library))

/* check for default of *FILE in document name                      */
/* ·If *FILE, set document to File Name specified                   */
            IF         COND(&document *eq '*FILE') then( +
            CHGVAR     VAR(&document) value(&FILEN *tcat '.JSON'))

/* check for trailing / in directory name                           */
/* ·slash (/) found, concatenate directory & document name          */
/* ··slash (/) not found, concatenate directory name, slash,        */
/*   document name                                                  */
            CALL       PGM(QCLSCAN) PARM(&DIRECTORY &STRLEN &STRPOS +
                       &PATTERN &PATLEN &TRANSLATE &TRIM +
                       &WILDCARD &RETURN)

            IF         COND(&return *ne 000) THEN( +
            CHGVAR     VAR(&EOSSTMF) VALUE(&directory |< +
                       &document |< &NULL))

            IF         COND(&return *eq 000) THEN( +
            CHGVAR     VAR(&EOSSTMF) VALUE(&directory |< '/' |< +
                       &document ))
```

Coding our Processing CLLE-continued

```
/* execute RPGJSONR program to generate JSON for specified table   */
            CALL       PGM(RPGJSONR) PARM(&FILEN &LIBRARY &EOSSTMF)

            RCLRSC     LVL(*)
            ENDPGM
```

Coding our SQL RPGLE

```
// *-------------------------------------------------------------------------------------------*
// * Create Instructions:                                                                      *
// *                                                                                           *
// * CRTSQLRPGI OBJ(RPGJSONR) SRCFILE(QRPGSRC)                                                 *
// *-------------------------------------------------------------------------------------------*
    dRPGJSONR        PR                EXTPGM('RPGJSONR')
    d  File                      10
    d  Library                   10
    d  JSONDocument             256

    DRPGJSONR        PI
    d  File                      10
    d  Library                   10
    d  JSONDocument             256

      // Execute system command prototype
    d qcmdexc            pr              extpgm('QCMDEXC')
    d  command            500a
    d  length              15p 5

      // Columns DS contains the columns names in the table
    D Columns        Ds                  Dim(100) Qualified
    D  Column                 10A

      // Values DS contains all the table contents for the columns in the table
    D Values          Ds                 Dim(100) Qualified
    D  Values001            128A
    D  Values002            128A
```

Coding our SQL RPGLE-continued

```
D  Values003             128A
D  Values004             128A
D  Values005             128A
D  Values006             128A
D  Values007             128A
D  Values008 ~~ 100  128A

D Values_n       ds                  dim(100) qualified
D  std_null_val                5i 0  dim(4)

  // Work field(s)
D apos              c                   const('')                   Apostrophe constant
D apossubst         c                   const('''')                 Apostrophe constant
D colon             c                   const(': ')                 Colon constant
D comma             c                   const(', ')                 Comma constant
d command           s            500
D elements          S              5I 0     Inz(%Elem(Columns))
D Index1            S              3U 0
D Index2            S              3U 0
d length            s             15p 5
D NbrOfColumns      S                   LIKE(SQLER3)
D NbrOfValues       S                   LIKE(SQLER3)
D pos               S              5U 0
D quote             c                   const('')                   Quote constant
d SqlStm            s            5000     varying
D start             S                5U 0
D statement         S            2000
d Value_wrk         s             128
```

Coding our SQL RPGLE-continued

```
/free

 // ************************************************************************************
 // * Retrieve the list of columns for the table and place in the Columns DS *
 // ************************************************************************************
 EXEC SQL Declare ColCsr Cursor for
  Select column_name from QSYS2/SYSCOLUMNS
   where table_schema = :Library and table_name = :File
    for read only;

 EXEC SQL Open ColCsr;
 EXEC SQL Fetch next from ColCsr for :elements rows into :Columns;
  NbrOfColumns = SqlEr3;
 EXEC SQL Close ColCsr;

 // ************************************************************************************
 // * For each of the column names, retrieve the values from the DB table   *
 // * and store in the Values DS                                             *
 // ************************************************************************************
 For Index1 = 1 to NbrOfColumns;
  statement = %trim(statement) + 'CHAR(' + %trim(Columns(Index1).Column) +
              ')';

  If Index1 <> NbrOfColumns;
    statement = %trim(statement) + ',';
  Endif;
 EndFor;
```

Coding our SQL RPGLE-continued

```
SqlStm = 'Select ' + %trim(statement) + 'from ' + %trim(library) +
            '/' + %trim(file) + ' For Read Only';

EXEC SQL Prepare Cursor1 from :SqlStm;
EXEC SQL Declare FldCsr cursor for Cursor1;

EXEC SQL Open FldCsr;
EXEC SQL Fetch next from FldCsr for :elements rows into :Values
                :Values_n;
 NbrOfValues = SqlEr3;
 EXEC SQL Close FldCsr;

// ***************************************************************************
// * Create the JSON data for specified table                                *
// ***************************************************************************
EXEC SQL
 create table qtemp/RPGJSON
  (char1 char(64));

// Insert JSON start of document curly bracket
EXEC SQL
 insert into qtemp/RPGJSON
   (char1)
   values('{')
 with nc;
```

Coding our SQL RPGLE-continued

```
// Insert ROOT tag
EXEC SQL
 insert into qtemp/RPGJSON
   (char1)
   values(' "Root_Element": {')
 with nc;

// Before we can generate our JSON document for our table, the column data must be
// cleansed of any values that will cause issues with our JSON string we build to
   // insert into our work table.
 For Index1 = 1 to NbrOfColumns;
  For Index2 = 1 to NbrOfValues;

   Pos = (Index1 * 128) - 127;
   Value_wrk = %Subst(Values(Index2):Pos:128);
   clear Statement;

// Scan for existence of a ' (apostrophe) in our data and replace with 4 apostrophes
   // so that our string build does not error out
   If %scan(apos: Value_wrk) >*zero;
    start = %scan(apos: Value_wrk) -1;
    Value_wrk = %subst(Value_wrk:1:start) + apossubst +
                %subst(Value_wrk:start+2);
   Endif;

   %Subst(Values(Index2):Pos:128)=Value_wrk;
  Endfor;
 Endfor;
```

Coding our SQL RPGLE-continued

```
// Insert Table Rows tag
EXEC SQL
   insert into qtemp/RPGJSON
    (char1)
    values('  "Table_Rows": [')
    with nc;

// Build JSON document data from column names and values
For Index1 = 1 to NbrOfValues;
 For Index2 = 1 to NbrOfColumns;

// Check if Index is 1st occurrence
// ·If first, insert curley bracket start tag
  If Index2 = 1;
   EXEC SQL
    insert into qtemp/RPGJSON
     (char1)
     values('   {')
     with nc;
  Endif;

  clear  sqlstm;

  Pos = (Index2 * 128) - 127;
  Value_wrk = %Subst(Values(Index1):Pos:128);
```

Coding our SQL RPGLE-continued

```
// Insert JSON table data by building an insert string of the column data
  If Index2 < NbrOfColumns;
   command= quote + %trim(Columns(Index2).Column) + quote +
        colon + quote + %trim(Value_wrk) + quote + comma ;
  Endif;
  If Index2 = NbrOfColumns;
   command= quote + %trim(Columns(Index2).Column) + quote +
        colon + quote + %trim(Value_wrk) + quote ;
  Endif;

  sqlstm = 'insert into qtemp/RPGJSON   (char1) +
     values(' + apos + '     ' + command + apos + ') with nc';

  EXEC SQL execute immediate  :SqlStm;
 EndFor;

// Check if all columns processed
   // ·If not all processed, insert end curley bracket tag w/comma
   //  ··If all processed, insert end curley bracket tag only
 If Index1 < NbrOfValues;
  EXEC SQL
   insert into qtemp/RPGJSON
   (char1)
   values('   },')
   with nc;
  Endif;
```

Coding our SQL RPGLE-continued

```
If Index1 = NbrOfValues;
   EXEC SQL
    insert into qtemp/RPGJSON
    (char1)
    values('   }')
    with nc;
   Endif;
 EndFor;

 // Insert end bracket tag
  EXEC SQL
   insert into qtemp/RPGJSON
    (char1)
    values('  ]')
    with nc;

 // Insert end curley bracket tag
  EXEC SQL
   insert into qtemp/RPGJSON
    (char1)
    values(' }')
    with nc;

 // Insert end ROOT tag
 EXEC SQL
  insert into qtemp/RPGJSON
    (char1)
    values('}')
    with nc;
```

Coding our SQL RPGLE-continued

```
// ***************************************************************************
// Execute CPYTOIMPF to create an JSON stream file                           *
// ***************************************************************************
command = 'CPYTOIMPF FROMFILE(QTEMP/RPGJSON) TOSTMF(' +
                apos + %trim(JSONDocument) + apos +
                ') MBROPT(*REPLACE) STMFCODPAG(819) RCDDLM(*CRLF) +
                DTAFMT(*FIXED) STRDLM(*NONE)';

length = %len(%trim(command));
callp  qcmdexc(command:length);

// ***************************************************************************
// * Drop the JSON work table                                                 *
// ***************************************************************************
EXEC SQL
 drop table qtemp/RPGJSON  ;

*Inlr = *on;
```

Coding our SQL RPGLE-continued

```
//*************************************************************************
//* Create the XML data for specified table                               *
//*************************************************************************
      EXEC SQL
       create table qtemp/rpggenxml
        (char1 char(2000));

 // Insert XML declaration to define the XML version and encoding
      EXEC SQL
       insert into qtemp/rpggenxml
         (char1)
         values('<?xml version="1.0" encoding="UTF-8"?>')
      with nc;

 // Insert ROOT element start tag
      EXEC SQL
       insert into qtemp/rpggenxml
         (char1)
         values('<Root_Element>')
      with nc;
```

Coding our SQL RPGLE-continued

```
// Before we can generate our XML document for our table, the column data must be cleansed
// of any XML Character Entity values that will cause issues with our XML document.
     For Index1 = 1 to NbrOfColumns;
        For Index2 = 1 to NbrOfValues;

        Pos = (Index1 * 128) - 127;
        Value_wrk = %Subst(Values(Index2):Pos:128);
        clear Statement;

// Scan for existence of a & in our data and replace with :&amp; which is allowed in XML
         If %scan('&': Value_wrk) >*zero;
          start = %scan('&': Value_wrk) -1;
          Value_wrk = %subst(Value_wrk:1:start) + xamp +  %subst(Value_wrk:start+2);
        Endif;

 // Scan for existence of a ' in our data and replace with :&apos; which is allowed in XML
         If %scan("": Value_wrk) >*zero;
          start = %scan("": Value_wrk) -1;
          Value_wrk = %subst(Value_wrk:1:start) + xapos + %subst(Value_wrk:start+2);
        Endif;

// Scan for existence of a > in our data and replace with :&gt; which is allowed in XML
         If %scan('>':  Value_wrk) >*zero;
          start = %scan('>': Value_wrk) -1;
          Value_wrk = %subst(Value_wrk:1:start) + xgt + %subst(Value_wrk:start+2);
        Endif;
```

Coding our SQL RPGLE-continued

```
// Scan for existence of a < in our data and replace with :&lt; which is allowed in XML
        If %scan('<':  Value_wrk) >*zero;
         start = %scan('<': Value_wrk) -1;
         Value_wrk = %subst(Value_wrk:1:start) + xlt +  %subst(Value_wrk:start+2);
        Endif;

// Scan for existence of a " in our data and replace with :&quot; which is allowed in XML
        If %scan('"':  Value_wrk) >*zero;
         start = %scan('"': Value_wrk) -1;
         Value_wrk = %subst(Value_wrk:1:start) + xquot + %subst(Value_wrk:start+2);
        Endif;

        %Subst(Values(Index2):Pos:128)=Value_wrk;
       Endfor;
     Endfor;

 // Build XML document data from column names and values
     For Index1 = 1 to NbrOfValues;
       For Index2 = 1 to NbrOfColumns;

// Check if Index is 1st occurrence
//  If first, insert Table Row Child element start tag
        If Index2 = 1;
         EXEC SQL
           insert into qtemp/rpggenxml (char1) values(' <Table_Rows>')
           with nc;
        Endif;
```

Coding our SQL RPGLE-continued

```
        clear  sqlstm;

         Pos = (Index2 * 128) - 127;
         Value_wrk = %Subst(Values(Index1):Pos:128);

// Select XML Element tag name format
            sqlstm = 'insert into qtemp/rpggenxml (char1) +
             values('+ apos +
                  Xbl + %trim(Columns(Index2).Column) + Xbr +
                  %trim(Value_wrk) +
                  Xblc + %trim(Columns(Index2).Column) + Xbr +
                   apos + ') +
              with nc';

          EXEC SQL execute immediate  :SqlStm;
         EndFor;

// Insert Table Row Child element end tag
         EXEC SQL
          insert into qtemp/rpggenxml   (char1) values(' </Table_Rows>')
           with nc;
       EndFor;

// Insert ROOT element end tag
        EXEC SQL
         insert into qtemp/rpggenxml
           (char1)
           values('</Root_Element>')
Ì        with nc;
```

Coding our SQL RPGLE-continued

```
// ************************************************************************
// Execute CPYTOIMPF to create an XML stream                              *
// ************************************************************************
      command = 'CPYTOIMPF FROMFILE(QTEMP/rpggenxml) TOSTMF(' +
            apos + %trim(XMLDocument) + apos +
            ') MBROPT(*REPLACE) STMFCODPAG(819) RCDDLM(*CRLF) +
            DTAFMT(*FIXED) STRDLM(*NONE)';

      length = %len(%trim(command));
      callp  qcmdexc(command:length);

 // ************************************************************************
 // * Drop the XML work table                                              *
 // ************************************************************************
      EXEC SQL
       drop table qtemp/rpggenxml;

      *Inlr = *on;
```

THiNK

An exploration into making the world work better