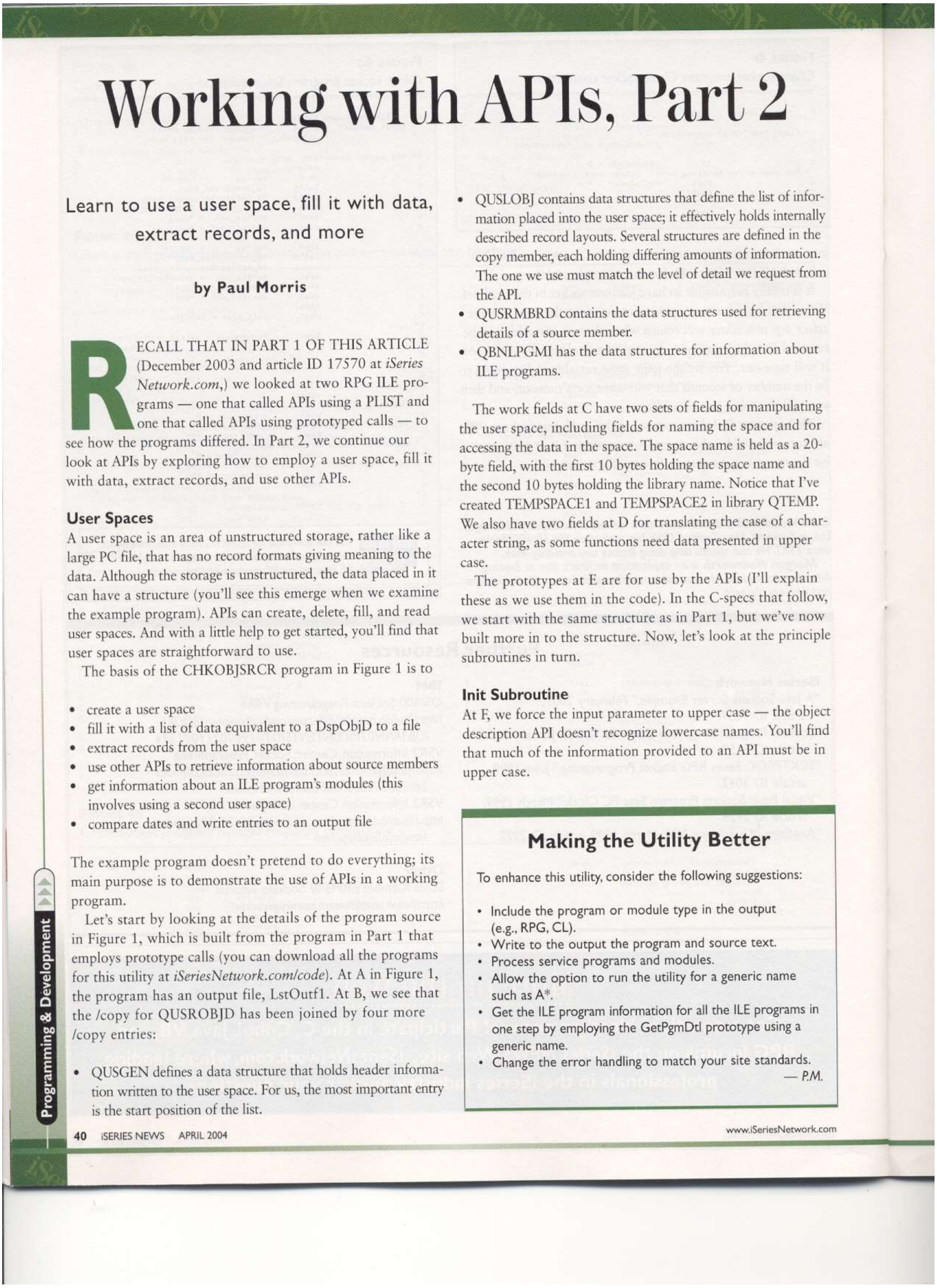
Working with APIs Part 2



Working with APIs, Part 2

Learn to use a user space, fill it with data, extract records, and more

by Paul Morris

RECALL THAT IN PART 1 OF THIS ARTICLE (December 2003 and article ID 17570 at *iSeries Network.com*,) we looked at two RPG ILE programs — one that called APIs using a PLIST and one that called APIs using prototyped calls — to see how the programs differed. In Part 2, we continue our look at APIs by exploring how to employ a user space, fill it with data, extract records, and use other APIs.

User Spaces

A user space is an area of unstructured storage, rather like a large PC file, that has no record formats giving meaning to the data. Although the storage is unstructured, the data placed in it can have a structure (you'll see this emerge when we examine the example program). APIs can create, delete, fill, and read user spaces. And with a little help to get started, you'll find that user spaces are straightforward to use.

The basis of the CHKOBJSRCR program in Figure 1 is to

- create a user space
- fill it with a list of data equivalent to a DspObjD to a file
- extract records from the user space
- use other APIs to retrieve information about source members
- get information about an ILE program's modules (this involves using a second user space)
- compare dates and write entries to an output file

The example program doesn't pretend to do everything; its main purpose is to demonstrate the use of APIs in a working program.

Let's start by looking at the details of the program source in Figure 1, which is built from the program in Part 1 that employs prototype calls (you can download all the programs for this utility at *iSeriesNetwork.com/code*). At A in Figure 1, the program has an output file, LstOutf1. At B, we see that the /copy for QUSROBJD has been joined by four more /copy entries:

- QUSGEN defines a data structure that holds header information written to the user space. For us, the most important entry is the start position of the list.
- QUSLOBJ contains data structures that define the list of information placed into the user space; it effectively holds internally described record layouts. Several structures are defined in the copy member, each holding differing amounts of information. The one we use must match the level of detail we request from the API.
- QUSRMBRD contains the data structures used for retrieving details of a source member.
- QBNLPGMI has the data structures for information about ILE programs.

The work fields at C have two sets of fields for manipulating the user space, including fields for naming the space and for accessing the data in the space. The space name is held as a 20-byte field, with the first 10 bytes holding the space name and the second 10 bytes holding the library name. Notice that I've created TEMPSPACE1 and TEMPSPACE2 in library QTEMP. We also have two fields at D for translating the case of a character string, as some functions need data presented in upper case.

The prototypes at E are for use by the APIs (I'll explain these as we use them in the code). In the C-specs that follow, we start with the same structure as in Part 1, but we've now built more in to the structure. Now, let's look at the principle subroutines in turn.

Init Subroutine

At F, we force the input parameter to upper case — the object description API doesn't recognize lowercase names. You'll find that much of the information provided to an API must be in upper case.

Making the Utility Better

To enhance this utility, consider the following suggestions:

- Include the program or module type in the output (e.g., RPG, CL).
- Write to the output the program and source text.
- Process service programs and modules.
- Allow the option to run the utility for a generic name such as A*.
- Get the ILE program information for all the ILE programs in one step by employing the GetPgmDtl prototype using a generic name.
- Change the error handling to match your site standards.

— P.M.

Figure 1
Program CHKOBJSRCR

```
     *====================================================================
     * Object Name   - CHKOBJSRCR  Version   1.0      Date  16.07.2003
     * Based on      -             Version   1.0      As at   .  .
     * Author        - Paul Morris, Welwyn Support Services Ltd.
     * Function      - API Example Utility build part 3
     *
     *
     *====================================================================
    H Copyright('Copyright (C) 2003 Welwyn Support Services Ltd.')
    H Debug Datedit(*DMY) Datfmt(*ISO)
     *====================================================================
(A) fLstOutf1  o    e             disk    infsr(*pssr)   Prefix(l_)
    *====================================================================
     * API Error data structure
    d Qusec           DS
    d Qusbprv                       10i 0 Inz(%size(Qusec))
    d Qusbavl                       10i 0 Inz(0)
    d Qusei                          7
    d Quserved                       1
    D QusMsgData                   240
     *====================================================================
     * API copy sources
     *--------------------------------------------------------------------
(B) * API structure for retrieving object description API
     /copy QSYSINC/QRPGLESRC,QUSROBJD
     *--------------------------------------------------------------------
     /copy QSYSINC/QRPGLESRC,QUSGEN
     * API structure for retrieving list API generic header
     *--------------------------------------------------------------------
     /copy QSYSINC/QRPGLESRC,QUSLOBJ
     * API structure for retrieving list of objects
     *--------------------------------------------------------------------
     /copy QSYSINC/QRPGLESRC,QUSRMBRD
     * API structure for retrieving member description API
     *--------------------------------------------------------------------
     /copy QSYSINC/QRPGLESRC,QBNLPGMI
     * API structure for retrieving list program information
    *====================================================================
     * API parameter fields and work areas
(C) d @InLib          s             10
    d SpaceName1      s             20    Inz('TEMPSPACE1QTEMP')
    d Indx            s             10i 0
    d SpcStr          s             10i 0
    d SpcLen          s             10i 0
    d HdrEntStr       s             10i 0
    d HdrEntLen       s             10i 0
    d HdrEntNbr       s             10i 0

    d SpaceName2      s             20    Inz('TEMPSPACE2QTEMP')
    d Indx2           s             10i 0
    d SpcStr2         s             10i 0
    d SpcLen2         s             10i 0
    d HdrEntStr2      s             10i 0
    d HdrEntLen2      s             10i 0
    d HdrEntNbr2      s             10i 0
(D) d Lo              c                   'abcdefghijklmnopqrstuvwxyz'
    d Up              c                   'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    *====================================================================
     * Prototypes
     *--------------------------------------------------------------------
(E) * Prototype for retrieving an object description
    d RtvObjD         pr                  extpgm('QUSROBJD')
    d  ReturnArea                 1024    options(*varsize)
    d  RtnLen                       10i 0 const
    d  Format                        8    const
    d  ObjFilLib                    20    const
    d  ObjType                      10    const

    d  ErrRtn                       16
     *--------------------------------------------------------------------
     *  Prototype for Create user space API
    d CrtSpace        Pr                  extpgm('QUSCRTUS')
    d  UsrSpcNam                    20a   const
    d  SpcAtr                       10a   const
    d  SpcSiz                       10i 0 const
    d  SpcInt                        1a   const
    d  SpcAut                       10a   const
    d  SpcTxt                       50a   const
    d  SpcRpl                       10a   const
    d  ErrRtn                             like(Qusec)
    d  SpcDmn                       10a   Const

     *  Prototype for Delete user space API
    d DltSpace        Pr                  extpgm('QUSDLTUS')
```

continued on page 42

We validate the library name as before but take a different approach with errors (at G). For this example program, we'll keep error handling to a minimum, so we write an error message to the file before aborting the program with a call to *Pssr.

Next, at H, we create the two user spaces using the parameters of

- the space name
- a space attribute that I've arbitrarily set to 'TEMPORARY'
- an initial size (this is the smallest size possible, but spaces can grow)
- a fill character of hexadecimal zero
- a public authority, which I've set to *ALL (but since it's in QTEMP, no one else can get to it anyway)
- some text
- *YES to replace any existing space of that name
- the error structure Qusec, which is common to all these API calls
- the domain in which the space is placed (*USER)

At I, we list the objects into the user space with the following parameters:

- the space name
- the format of the data required
- the name of the object we want listed and its library (here, we use the special value of *ALL for all object names, with the library name coming from the entry parameter)
- the object type (*PGM)
- the error structure

You can find details such as the formats and the data returned at IBM Information Center's API page at *http://publib.boulder.ibm.com/iseries/v5r2/ic2924/info/apis/api.html.*

Main1 Subroutine

In the Main1 subroutine at J, we get the header at the beginning of the user space — this tells us about the data loaded into the space. I've separated this into a different subroutine, GetSpcHdr, to make it easier to cut and paste code into other programs. I also provide a failsafe here for no entries returned (i.e., no program objects exist in the library). Notice, too,

FIGURE 1 *continued*

```
d  UsrSpcNam                   20a   const
d  ErrRtn                            like(Qusec)
 *-----------------------------------------------------------------
 *  Prototype for listing objects API
d LstObjs         Pr                 extpgm('QUSLOBJ')
d  UsrSpcNam                   20a   const
d  LstFormat                    8a   const
d  ObjAndLib                   20a   const
d  ObjType                     10a   const
d  ErrRtn                            like(Qusec)
 *-----------------------------------------------------------------
 *  Prototype for retrieving member details API
d GetMbrD         Pr                 extpgm('QUSRMBRD')

d  FormatDS                   135a
d  FmtLen                      10i 0 const
d  FmtNam                       8a   const
d  FilLibNam                   20a   const
d  Member                      10a   const
d  Override                     1a   const
d  ErrRtn                            like(Qusec)
 *-----------------------------------------------------------------
 *  Prototype for getting user space entry
d UsrSpcEnt       Pr                 extpgm('QUSRTVUS')
d  UsrSpcNam                   20a   const
d  SpcEntStr                   10i 0 const
d  SpcEntLen                   10i 0 const
d  SpcEntDS                   512a   options(*varsize)
d  ErrRtn                            like(Qusec)

 *-----------------------------------------------------------------
 *  Prototype for getting ILE program details into a space
d GetPgmDtl       Pr                 extpgm('QBNLPGMI')
d  UsrSpcNam                   20a   const
d  FmtNam                       8a   const
d  PgmLibNam                   20a   const
d  ErrRtn                            like(Qusec)
 *=================================================================
 * Definitions
 *-----------------------------------------------------------------
c     *Entry        Plist
c                   Parm                    @InLib
 *=================================================================

c                   Exsr     Init

c                   Exsr     Main1

c                   Exsr     Exit

 *=================================================================
 * 1-off initialization
 *-----------------------------------------------------------------
c     *Inzsr        Begsr


c                   Endsr
 *=================================================================
 * initialization
 *-----------------------------------------------------------------
(F) c     Init          Begsr

 * force library name to upper case
c     Lo:Up         Xlate(p)  @inlib        L_ObjLib
 * call API to validate library name
(G) c                   Reset                   qusec
c                   Callp    RtvObjd(Qusd0100 : %len(Qusd0100) :
c                            'OBJD0100' : L ObjLib + 'QSYS' : '*LIB' :
c                            Qusec)
c                   If       Qusbavl <> 0
c                   Eval     L Diagnostic = 'Invalid Library'
c                   Write    LstOutR1
c                   Exsr     *Pssr
c                   Endif
 * call API to create user space 1
(H) c                   Reset                   qusec
c                   Callp    CrtSpace(SpaceName1: 'TEMPORARY':
c                            8192: X'00':'*ALL':
c                            'Utility use':
c                            '*YES': Qusec : '*USER')
c                   If       Qusbavl <> 0
c                   Eval     L ObjLib = Spacename1
c                   Eval     L Diagnostic = 'Unable to create user space1'
c                   Write    LstOutR1
c                   Exsr     *Pssr
```

continued on page 44

that we have a do loop that indexes through the number of entries to retrieve the data records.

At K, the GetSpcHdr subroutine employs the QUSRTVUS API defined in the UsrSpcEnt prototype (in section E). Notice, the header is starting at position 1 for the length of the predefined header format Qush0100. This returns into the Qush0100 format (among other data) the start point of the list in Qusold (OLD = Offset to List Data), the length of the entry Qussee (this value should match the length of the data structure OBJL0400), and Qusnbrle (the number of list entries). Because we reuse this data structure, these three fields are saved to work fields. Now, let's look at the GetSpcDtl subroutine.

GetSpcDtl Subroutine

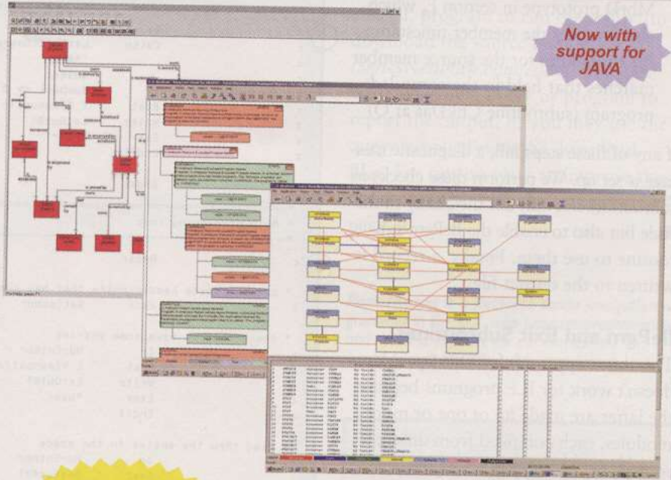
In this subroutine at L, we calculate the start position of each entry from the offset and the index counter multiplied by the record length. The "+1" takes us from an offset to a start position. We then call the API using the UsrSpcEnt prototype (in section E) to get the record from the space, but this time, we place it in the Qusl0400 data structure. Now we have data in this data structure resembling that obtained in a record created by DspObjD.

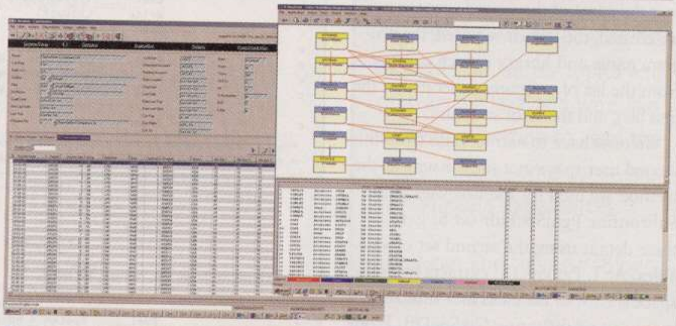
Why not use DspObjD? You can do so for this application (the original version, written some years ago, did this), but APIs are faster, more flexible, and provide scope for future enhancements.

Now that we have the record, we must decide what to do with it. For an ILE program, no source information is in this record, so we execute the IlePgm subroutine that needs to do more work. For an OPM program, we now have source details from the program, so we can execute a simple subroutine — OpmPgm — to process these records. We differentiate between OPM and ILE programs by identifying the compiler used to create the program.

OpmPgm Subroutine

With the OpmPgm subroutine (at M), we set up some basic details in the output file and then go through several checks.





Using the original RtvObjd prototyped call, we can now check that

- the source library exists (subroutine ChkLib at N)
- the source file exists (subroutine ChkFil at O)
- the source member exists (subroutine ChkMbr at P); to do this, we get the member description using the Get MbrD prototype in section E, which also provides the member timestamp
- the timestamp for the source member matches that held in the compiled program (subroutine ChkDat at Q)

If any of these steps fail, a diagnostic message is set up. We perform these checks as subroutines not only for the clarity of the code but also to enable the IlePgm subroutine to use them. Finally, a record is written to the output file.

IlePgm and Exit Subroutines

The above approach for OPM programs doesn't work for ILE programs because the latter are made up of one or more modules, each compiled from different sources. So we use the API equivalent of DspPgm — QBNLPGMI — that we defined in the GetPgmDtl prototype in section E. Again, as with many APIs, you can select the amount of data returned.

In the IlePgm subroutine at R, we use the PGML0100 format, which tells us about the modules used by the program. After we initialize the output fields, the API is called using the prototype, passing to it the name of the second user space we created, the format required, the program name and library (which came from the list of programs generated in the first list), and the error structure.

We now have to extract data from this second user space in a similar way to the method we used for the first user space. Subroutine PgmSpcHdr (at S) gets these details using the second set of variables. At T, PgmSpcDtl extracts from the space the list of modules in a similar manner to subroutine GetSpcDtl. From the PgmSpcDtl subroutine, subroutine ChkPgmModSrc (at U) is executed to check the source details for each module using the same subroutines that

FIGURE 1 *continued*

```
   c                 Endif

    * call API to create user space 2
   c                 Reset                      qusec
   c                 Callp     CrtSpace(SpaceName2: 'TEMPORARY':
   c                           8192: X'00':'*ALL':
   c                           'Utility use':
   c                           '*YES': Qusec : '*USER')
   c                 If        Qusbavl <> 0
   c                 Eval      l ObjLib = Spacename1
   c                 Eval      l Diagnostic = 'Unable to create user space2'
   c                 Write     LstOutR1
   c                 Exsr      *Pssr
   c                 Endif
    * call API to list objects into user space1
I  c                 Reset                      qusec
   c                 Callp     LstObjs(SpaceName1: 'OBJL0400':
   c                           '*ALL      ' + l ObjLib : '*PGM' :
   c                           Qusec)
   c                 If        Qusbavl <> 0
   c                 Eval      l Diagnostic = 'Unable to create object list'
   c                 Write     LstOutR1
   c                 Exsr      *Pssr
   c                 Endif

   c                 Endsr
    *==================================================================
    * Main processing loop
    *------------------------------------------------------------------
J  c     Main1       Begsr

    * get the space header entry that has control info
   c                 Exsr      GetSpcHdr

    * check that we have some entries
   c                 If        HdrEntNbr = 0
   c                 Eval      l Diagnostic = 'No program entries'
   c                 Write     LstOutR1
   c                 Exsr      *pssr
   c                 Endif

    * Read thru the enties in the space
   c                 Do        HdrEntNbr     Indx
   c                 Exsr      GetSpcDtl
   c                 Enddo

   c                 Endsr
    *==================================================================
    * Get the user space header entry
    *------------------------------------------------------------------
K  c     GetSpcHdr   Begsr

   c                 Reset                      qusec
   c                 Eval      SpcStr = 1
   c                 Eval      SpcLen = %len(Qush0100)
   c                 Callp     UsrSpcEnt(SpaceName1: SpcStr: SpcLen:
   c                           Qush0100: Qusec)
   c                 If        Qusbavl <> 0
   c                 Eval      l Diagnostic = 'Read of space1 header failed'
   c                 Write     LstOutR1
   c                 Exsr      *pssr
   c                 Endif
   c                 Eval      HdrEntStr = Qusold
   c                 Eval      HdrEntLen = Qussee
   c                 Eval      HdrEntNbr = Qusnbrle

   c                 Endsr
    *==================================================================
    * Get the list entries by stepping thru the space
    *------------------------------------------------------------------
L  c     GetSpcDtl   Begsr

   c                 Reset                      Qusec
   c                 Eval      l Diagnostic = ' '

    * set parms from the header format
   c                 Eval      SpcStr = HdrEntStr +
   c                                    ((Indx-1) * HdrEntLen) + 1
   c                 Eval      SpcLen = HdrEntLen
    * Get the data entry
   c                 Callp     UsrSpcEnt(SpaceName1: SpcStr: SpcLen:
   c                           Qusl0400: Qusec)
   c                 If        Qusbavl <> 0
   c                 Eval      l_Diagnostic = 'Read of space1 entry failed'
```

FIGURE 1 *continued*

```
c                    Write     LstOutR1
c                    Exsr      *pssr
c                    Endif

c                    If        %subst(Quspiler:1:6) = 'CRTPGM'
c                    Exsr      IlePgm
c                    Else
c                    Exsr      OpmPgm
c                    Endif

c                    Endsr
 *=====================================================================
 * Process OPM programs
 *---------------------------------------------------------------------
M c     OpmPgm        Begsr

c                    Eval      l ObjNam = QusObjNu02
c                    Eval      l ModNam = ' '
c                    Eval      l ModLib = ' '
c                   Eval      l ObjSrcLib = QussFln
c                   Eval      l ObjSrcFil = QussFiln
c                   Eval      l ObjSrcMbr = QussFmn
c                   Eval      l ObjSrcTim = QussFudt
c                   Eval      l Diagnostic = ' '
 * check on the source details
c                   Do
c                   Exsr      ChkLib
c                   If        l Diagnostic <> ' '
c                   Leave
c                   Endif
c                   Exsr      ChkFil
c                   If        l Diagnostic <> ' '
c                   Leave
c                   Endif
c                   Exsr      ChkMbr
c                   If        l Diagnostic <> ' '
c                   Leave
c                   Endif
c                   Exsr      ChkDat
c                   If        l Diagnostic <> ' '
c                   Leave
c                   Endif
c                   Enddo

c                   Write     LstOutR1

c                   Endsr
 *=====================================================================
 * Check the source library
 *---------------------------------------------------------------------
N c     ChkLib        Begsr

 * call API to validate library name
c                   Reset                 qusec
c                   Callp     RtvObjd(Qusd0100 : %len(Qusd0100) :
c                             'OBJD0100' : l ObjSrcLib + 'QSYS' : '*LIB' :
c                             Qusec)

c                   If        Qusbavl <> 0
c                   Eval      l Diagnostic = 'Source library not found'
c                   Endif

c                   Endsr
 *=====================================================================
 * Check the source file
 *---------------------------------------------------------------------
O c     ChkFil        Begsr

 * call API to validate source file
c                   Reset                 qusec
c                   Callp     RtvObjd(Qusd0100 : %len(Qusd0100) :
c                             'OBJD0100' : l ObjSrcFil + l ObjSrcLib :
c                             '*FILE' : Qusec)

c                   If        Qusbavl <> 0
c                   Eval      l Diagnostic = 'Source File not found'
c                   Endif

c                   Endsr
 *=====================================================================
 * Check the source file member
 *---------------------------------------------------------------------
```

continued on page 46

OpmPgm did. We wrap up with the Exit subroutine at V, which includes the API calls to delete the user spaces.

Give It a Try

We now have a program that can be called from the command line and will write records to an output file. Go ahead and try it with a small program library and use query or SQL to view the results. For completeness, I've included a simple command definition and CL program to run the utility (to download the source bundle, go to *iSeriesNetwork.com/code*). You can write your own query or program to report the output, or you may use the query supplied with the download files. For tips on ways to improve the utility, see "Making the Utility Better," page 40. ■

Paul Morris *is a freelance senior analyst/programmer in the U.K. who provides programming and systems support for the iSeries. You can e-mail him at Paul@wssltd.demon.co.uk.*