



APIs at Work

Bruce Vining
bvining@brucevining.com

*i want stress-free IT.
i want control.
i want an **i**.*

Copyright Bruce Vining Services LLC, 2007, 2012.
All Rights Reserved.

In This Session ...

- This session will go through the pros and cons, dos and don'ts of the primary types of System APIs
- System APIs can appear daunting at first, but when you dig a bit deeper you realize that there are really just a few API concepts that, once learned, allow you to easily access hundreds of APIs
- Examples used in the session will be based on RPG with the session handout providing the same examples in CL, COBOL, and C. By the end of this session, attendees will be able to use the vast majority of i5/OS APIs.

What We'll Cover ...

- **The Basics**
- Retrieve APIs
- List APIs
- Open List APIs
- Unix-type APIs
- Wrap-up

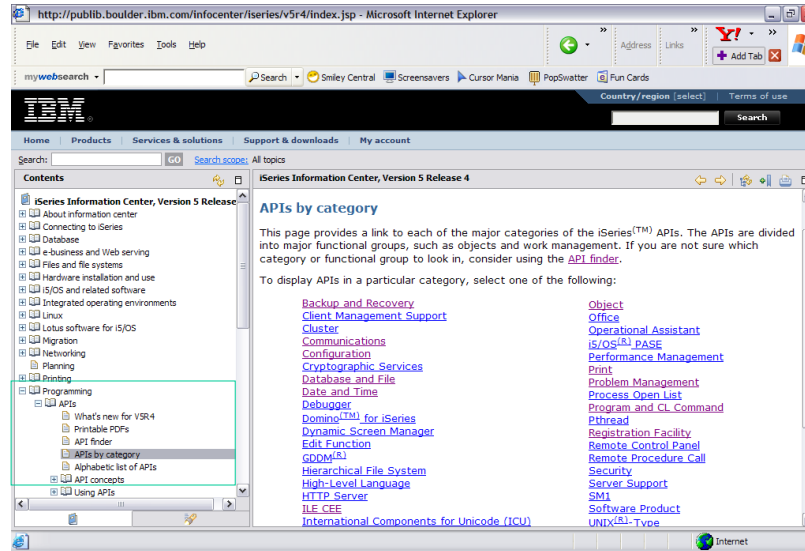
3

Main API Types

- **Retrieve**
These APIs return information to you as a variable when you call the API. The amount of data returned is generally less than 4K bytes.
- **List**
These APIs return information to an object known as a User Space (*USRSPC). You then use other APIs or based variables to retrieve the information. The amount of data returned can be up to 16M bytes.
- **Open List**
These APIs combine Retrieve and List characteristics. The data is returned to you as variables while a list of information is being built in the background. The amount of data returned can be up to 64G bytes.
- **Unix-type**
These APIs conform to industry standard interfaces that exist on other systems. Included are areas such as Integrated File System, sockets, and Lightweight Directory Access Protocol (LDAP).
- **Exit Points**
These are points in the system where you can be given control and/or notification. Included are areas such as Telnet and Command Analyzer where you can control how the system responds to requests.

4

The Prerequisite – System i Information Center



Retrieve Job Queue Information Parameters

■ QSPRJOBQ – A Sample Retrieve API

Send feedback | Rate this page

Retrieve Job Queue Information (QSPRJOBQ) API

Required Parameter Group:

Number	Parameter Name	Type	Length
1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Format name	Input	Char(8)
4	Qualified job queue name	Input	Char(20)
5	Error Code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: No

The Retrieve Job Queue Information (QSPRJOBQ) API retrieves information associated with a specified job queue.

Receiver variable is where data is returned by API
Length of receiver variable indicates how much storage you've allocated for Receiver variable
Format name indicates the form and content of the data you want returned

Note

- Char(*) indicates a variable length parameter. Often a data structure.
- Binary(4) indicates a 4-byte binary value (NOT 4B on D spec, rather use 9B or 10i)
- Char(8) indicates an 8-byte character value
- Char(20) indicates a 20-byte character value

Retrieve Job Queue Information Format

- Objective: How many jobs are on the *JOBQ?

JOBQ0100 Format

The following table shows the information returned for the JOBQ0100 format. For more details about the fields in the following table, see [Field Descriptions](#).

Offset	Dec	Hex	Type	Field
	0		BINARY(4)	Bytes returned
	4		BINARY(4)	Bytes available
	8		CHAR(10)	Job queue name
	18		CHAR(10)	Job queue library name
	28		CHAR(10)	Operator controlled
	38		CHAR(10)	Authority to check
	48		BINARY(4)	Number of jobs
	52		CHAR(10)	Job queue status
	62		CHAR(10)	Subsystem name
	72		CHAR(50)	Text description
	122		CHAR(10)	Subsystem library name
	132		BINARY(4)	Sequence number

Watch out: RPG Substring operations are base 1 while API documentation is base 0. Fine for pointer manipulation, but not for %SUBST or building your own include files. You need to add 1!

Caution

7

Retrieve Job Queue Information ILE RPG Include:

```

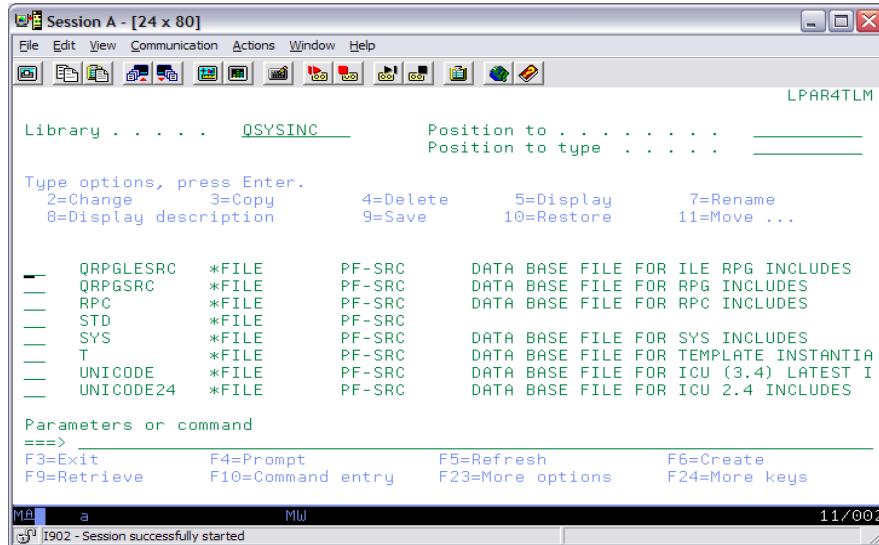
D*****
D*Type Definition for the JOBQ0100 format.
D*****
DQSPQ010000      DS
D*
D  QSPBRTN00      ①      4B 0      Qsp JOBQ0100
D*
D  QSPBAVL00      5      8B 0      Bytes Returned
D*
D  QSPJQN         9      18      Bytes Available
D*
D  QSPJQLN        19     28      Job Queue Name
D*
D  QSPQC01        29     38      Job Queue Lib Name
D*
D  QSPAC          39     48      Operator Controlled
D*
D  QSPAC          39     48      Authority Check
D  QSPNBRJ        49     52B 0      Number Jobs
D*

```

8

A Strong Recommendation

- QSYSINC System Openness Includes Installed as Option 13 of i5/OS



Retrieve Job Queue Information

- ILE RPG Program to Display the Number of Jobs on a JOBQ

```
d/copy qsysinc/qrpglesrc,qsprjobq
d/copy qsysinc/qrpglesrc,qusec

dRtvJobQ          pr                extpgm('QSPRJQBQ')
d Receiver                1          options(*varsize)
d LengthRcv             10i 0      const
d Format                 8          const
d JobQ                   20          const
d QUSEC                  likeds(QUSEC)

dJobQ
d JobQName              ds          10      inz('QBATCH')
d JobQLib                10          inz('*LIBL')

dWait               s              1

/free
QUSBPRV = 0;
RtvJobQ( QSPQ010000 (%size(QSPQ010000)) : 'JOBQ0100' :JobQ :QUSEC);
dsply QSPNBRJ ' ' Wait;
*inlr = *on;
/end-free
```

When using APIs that return data, always make sure the Length of the Receiver variable is set correctly and of the right type.

This is the first place to look when you get "bizarre" errors.

Use of the %size builtin is a best practice.



Tip

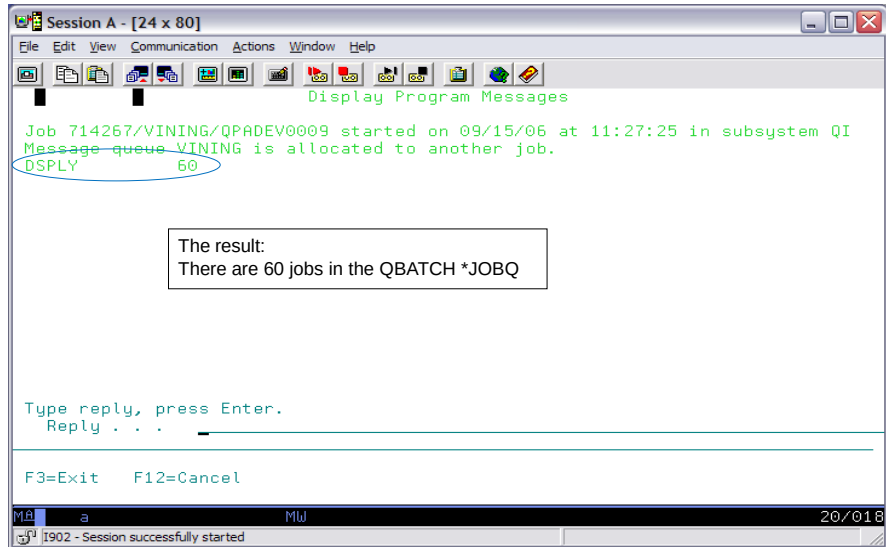


Best Practice

And you probably thought using APIs took lots of code!

Retrieve Job Queue Information (cont.)

- The Display of the Number of Jobs on QBATCH



The Error Code Structure

- If Bytes provided is >0 ALWAYS test Bytes available

Offset	Dec	Hex	Use	Type	Field
0	0		INPUT	BINARY(4)	Bytes provided
4	4		OUTPUT	BINARY(4)	Bytes available
8	8		OUTPUT	CHAR(7)	Exception ID
15	F		OUTPUT	CHAR(1)	Reserved
16	10		OUTPUT	CHAR(*)	Exception data

Offset	Dec	Hex	Use	Type	Field
0	0		INPUT	BINARY(4)	Key
4	4		INPUT	BINARY(4)	Bytes provided
8	8		OUTPUT	BINARY(4)	Bytes available
12	C		OUTPUT	CHAR(7)	Exception ID
19	13		OUTPUT	CHAR(1)	Reserved
20	14		OUTPUT	BINARY(4)	CCSID of the CCHAR data
24	18		OUTPUT	BINARY(4)	Offset to the exception data
28	1C		OUTPUT	BINARY(4)	Length of the exception data
			OUTPUT	CHAR(*)	Exception data

Bytes provided – how much error data you want returned to your program. 0 means send error messages.

Just like with Length of receiver variable, make sure this value reflects reality!

Bytes available – If Bytes provided is 8 or greater then a non-zero value indicates an error was found

Exception ID – If Bytes provided is 15 or greater AND Bytes available > 0 then this is the message ID of the error

Exception data – If Bytes provided is 17 or greater AND Bytes available > 0 then this is the replacement data for the error message

CCSID of the CCHAR data – If exception data was returned this is the CCSID of the CCHAR fields

The Error Code Structure (cont.)

■ ILE RPG Include:

```

D*****
D*Record structure for Error Code Parameter
D****
D*NOTE: The following type definition only defines the fixed
D* portion of the format. Varying length field Exception
D* Data will not be defined here.
D*****
DQUSEC          DS
D*
D* QUSBPRV          1          4B 0          Qus EC
D*
D* QUSBAVL          5          8B 0          Bytes Provided
D*
D* QUSEI           9          15          Bytes Available
D*
D* QUSERVED        16          16          Exception Id
D*
D* QUSED01         17          17          Reserved
D*
D*
D* Varying length

```

13

Retrieve Job Queue Information

■ ILE RPG Program to Display the Number of Jobs on a JOBQ

```

d/copy qsysinc/qrpglesrc,qsprjobq
d/copy qsysinc/qrpglesrc,qusec

dRtvJobQ          pr
d Receiver          1          options(*varsize)
d LengthRcv        10i 0 const
d Format            8          const
d JobQ             20          const
d QUSEC            likeds(QUSEC)

dJobQ             ds
d JobQName          10          inz('QBATCH')
d JobQLib           10          inz('*LIBL')

dWait             s          1

/free
QUSBPRV = 0;
RtvJobQ( QSPQ010000 :%size(QSPQ010000) :'JOBQ0100' :JobQ :QUSEC);
dsply QSPNBRJ ' ' Wait;
*inlr = *on;
/end-free

```

Have errors returned as Escape messages

14

Retrieve Job Queue Information (cont.)

- ILE RPG Program to Display the Number of Jobs on a JOBQ
 - Check for errors in the program

```
d/copy qsysinc/qrpglesrc,qsprjobq
d/copy qsysinc/qrpglesrc,qusec
```

```
dRtvJobQ          pr                  extpgm('QSPRJQBQ')
d Receiver                1          options(*varsize)
d LengthRcv              10i 0      const
d Format                  8          const
d JobQ                    20          const
d ErrCde                  likeds(ErrCde)
```

```
dErrCde           ds                  qualified
d EC               likeds(QUSEC)
d ErrorDta         100
```

Allocate an error code structure so that you can receive error replacement data

```
dJobQ             ds
d JobQName         10      inz('QBATCH')
d JobQLib           10      inz('*LIBL')

dWait             s         1
dMsg              s         40
```

15

Retrieve Job Queue Information – Check for Errors in the Program

```
/free
```

```
ErrCde.EC.QUSBPRV = %size(ErrCde);
```

Have errors returned as data

```
RtvJobQ( QSPQ010000 :%size(QSPQ010000) : 'JOBQ0100' :JobQ :ErrCde);
if (ErrCde.EC.QUSBAVL > 0);
```

```
  if (ErrCde.EC.QUSEI = 'CPF3307');
    Msg = 'Error ' + ErrCde.EC.QUSEI + ' on JobQ ' +
          %subst(ErrCde.ErrorDta :1 :10);
```

```
  else;
    Msg = 'Error ' + ErrCde.EC.QUSEI + ' returned';
  endif;
```

```
  dsply Msg ' ' Wait;
```

```
else;
  dsply QSPNBRJ ' ' Wait;
```

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

First check Bytes available (QUSBAVL)
If > 0 then check for the error message ID
If you handle the error then do so,
otherwise send it on
If = 0 then continue on as there's no error

16

Use of the Error Code Structure



Tip

- For initial development set Bytes provided to 0
- For production set Bytes provided to an appropriate non-zero value



Caution

- But make sure you test Bytes available
 - Would you assign an indicator for Record Not Found on a Chain and not test it?
 - Would you define a Global Monitor with an empty on-error?
 - Then don't ignore Bytes available

17

What We'll Cover ...

- The Basics
- Retrieve APIs
- List APIs
- Open List APIs
- Unix-type APIs
- Wrap-up

18

QDCRCTL D – Retrieve Controller Description

- Retrieve devices attached to a controller when you don't know how many devices there might be!

Retrieve Controller Description (QDCRCTL D) API

Required Parameter Group:

Number	Receiver variable	Output	Char(*)
1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Format name	Input	Char(8)
4	Controller name	Input	Char(10)
5	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: Yes

The Retrieve Controller Description (QDCRCTL D) API retrieves information about a controller description.

This Retrieve API has the same types of parameters as the last example. The change is that rather than passing a *JOBQ name we now pass a *CTL D name.

Building Blocks

19

QDCRCTL D – Retrieve Controller Description (cont.)

- Format CTLD0200 for attached devices

CTLD0200 Format

This format returns basic controller information, plus a list of attached devices. Some basic information is also included for each attached device.

Offset	Dec	Hex	Type	Field
0	0		BINARY(4)	Bytes returned
4	4		BINARY(4)	Bytes available
8	8		BINARY(4)	Number of attached devices
12	C		CHAR(7)	Date information retrieved
19	13		CHAR(6)	Time information retrieved
25	19		CHAR(10)	Controller name
35	23		CHAR(10)	Controller category
45	2D		CHAR(10)	Online at IPL
55	37		CHAR(50)	Text description
105	69		CHAR(3)	Reserved

The API will tell you how much data was returned and how much could have been returned

CTLD0200 Format

This format returns basic controller information, plus a list of attached devices. Some basic information is also included for each attached device.

Offset	Dec	Hex	Type	Field
0	0		BINARY(4)	Returns everything from format CTLD0100
108	6C		BINARY(4)	Offset to list of attached devices
112	70		BINARY(4)	Entry length for list of attached devices
			CHAR(10)	Attached device name
			CHAR(10)	Device category
			CHAR(10)	Device type
			CHAR(50)	Device text description

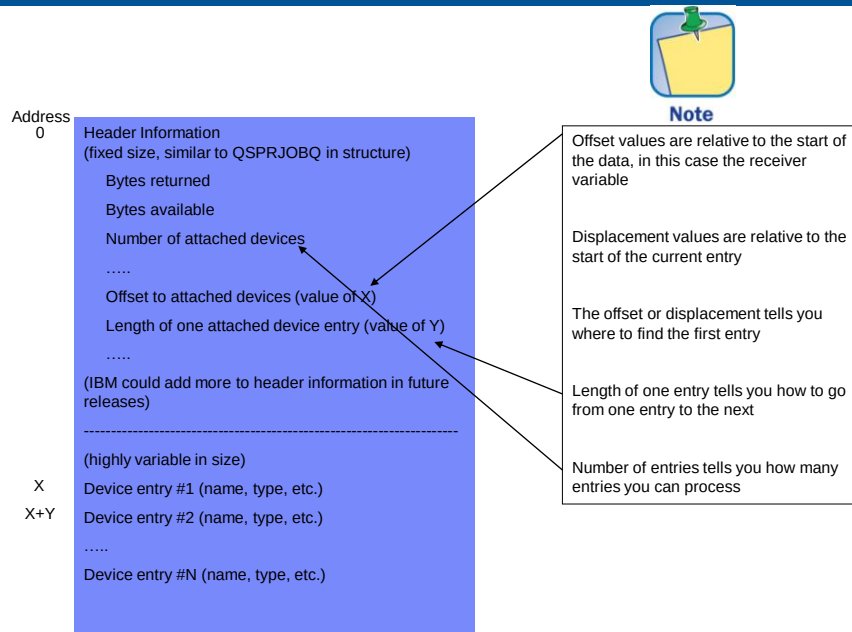
These fields repeat for each attached device

If the API returns an offset and length value then make sure you use it!

Building Blocks

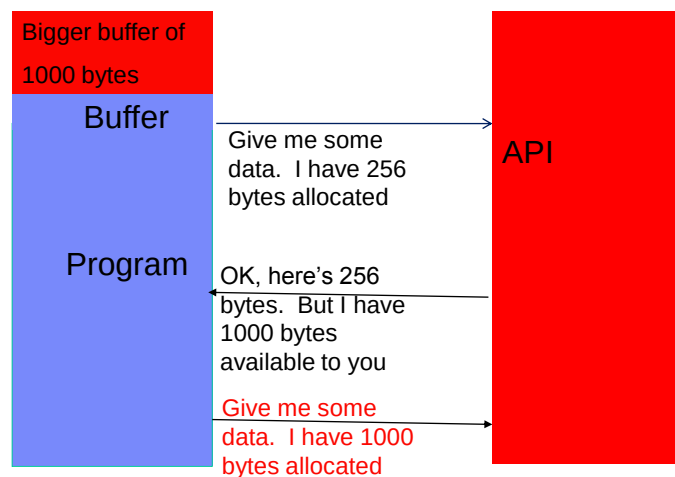
20

Format CTLD0200 for Attached Devices



21

Dynamically Allocating Storage to Adjust to System Changes



22

ILE RPG Include for Format CTLD0200

DQDCD020000 DS				Qdc CTLD0200
D*				
D*QDCD010001				108
D*				CTLD0100
D	QDCBRTN01	1	4B 0	Bytes Returned
D*	QDCBAVL01	5	8B 0	Bytes Available
D*	QDCNBRAD00	9	12B 0	Num Attached Device
D	QDCDIR01	13	19	Date Info Retrieved
D*	QDCTIR01	20	25	Time Info Retrieved
D	QDCCN02	26	35	Controller Name
D*	QDCCC00	36	45	Controller Category
D	QDCOAIPLO0	46	55	Online At IPL
D*	QDCTD01	56	105	Text Desc
D	QDCERVED03	106	108	Reserved
D*	QDCOAD	109	112B 0	Offset Attached Dev
D*	QDCLAD	113	116B 0	Length Attached Dev
D*				

23

ILE RPG Include for Format CTLD0200 Attached Devices

D*****				
D*Type Definition for the CTLD0200 format.				
D*****				
DQDCLAD00 DS				Qdc List Attached D
D*	QDCADN	1	10	Attached Device Nam
D*	QDCDC	11	20	Device Category
D	QDCDT	21	30	Device Type
D*	QDCDTD	31	80	Device Text Desc
D*				

We want to display the device names, so we're interested in QDCADN

24

QDCRCTL D – Retrieve Controller Description Retrieve the Devices

```
d/copy qsysinc/qrpglesrc,qdcrctld
d/copy qsysinc/qrpglesrc,qusec
```

```
dRtvCtld      pr      extpgm('QDCRCTL D')
d Receiver    pr      options(*varsize)
d LengthRcv   10i 0    const
d Format       8        const
d CtlName      10        const
d QUSEC        likeds(QUSEC)

dReceiver     ds      likeds(QDCD020000)
d              based(ReceivePtr)
dDevice        ds      likeds(QDCLAD00)
d              based(DevPtr)

dReceiverSize  s        10i 0 inz(8)
dCtld          s        10    inz('CTL01')
dWait          s        1
dCount         s        10i 0
```



Tip

Base the Receiver variable as it is variable length. The alternative is to guess a max size.

Also base the device information as it is accessed by offset and not at a fixed location. An alternative is to use substring operations.

25

QDCRCTL D – Retrieve Controller Description Retrieve the Devices (cont.)

```
/free
QUSBPRV = 0;
ReceivePtr = %alloc(ReceiverSize);
RtvCtld(Receiver :ReceiverSize : 'CTLD0200' : Ctld : QUSEC);
dow (Receiver.QDCBAVL01 > ReceiverSize); <
    ReceiverSize = Receiver.QDCBAVL01;
    ReceivePtr = %realloc(ReceivePtr :ReceiverSize);
    RtvCtld(Receiver :ReceiverSize : 'CTLD0200' : Ctld : QUSEC);
enddo;
if (Receiver.QDCNBRAD00 > 0); <
    DevPtr = ReceivePtr + Receiver.QDCOAD;
    for Count = 1 to Receiver.QDCNBRAD00;
        dsply Device.QDCADN;
        DevPtr = DevPtr + Receiver.QDCLAD;
    endfor;
endif;
dsply 'End of devices' ' ' Wait;
dealloc ReceivePtr;
*inlr = *on;
/end-free
```

Find out how much data is available by calling the API and looking at the Bytes available field QDCBAVL01. Then allocate that much storage and do it again (in case someone is adding devices concurrently).

When you have all the information returned (Bytes available is <= Bytes returned) then:

Check if there are any devices (QDCNBRAD00). If so use the offset to devices (QDCOAD) and Process QDCNBRAD00 devices displaying the device name (QDCADN).

If you don't use the provided offset information then you can be rudely surprised if the API adds more information to the CTLD0200 format.



Warning

Always de-allocate storage that you have allocated, otherwise you may have a "memory leak"

26

What We'll Cover ...

- The Basics
- Retrieve APIs
- List APIs
- Open List APIs
- Unix-type APIs
- Wrap-up

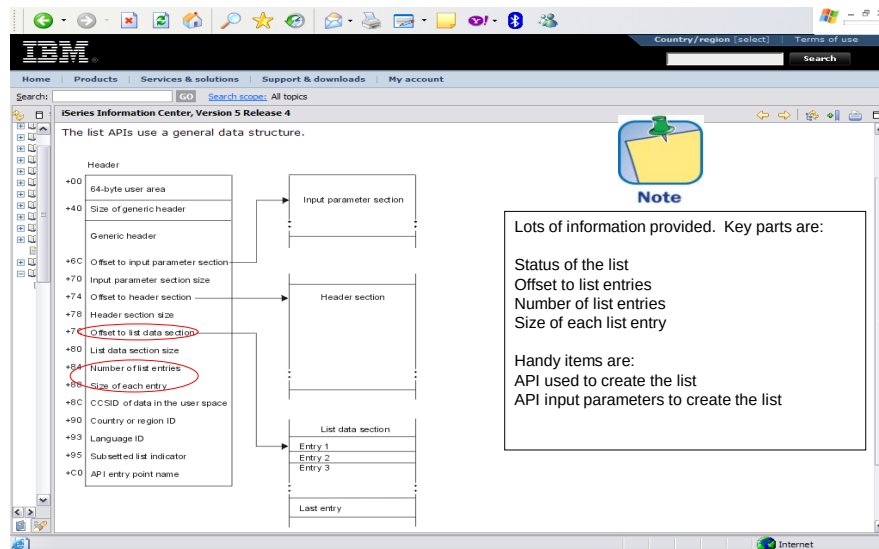
27

List APIs

- List APIs generate lists of information to a User Space (*USRSPC)
- *USRSPCs are permanent objects (unlike Receiver variables on Retrieve APIs)
- *USRSPCs can be up to 16M bytes in size
- List APIs will automatically extend the size of the *USRSPC to accommodate the available data
- APIs exist to:
 - Create a *USRSPC
 - Delete a *USRSPC
 - Change the contents of a *USRSPC
 - Retrieve the contents of a *USRSPC
 - Retrieve a pointer to a *USRSPC
 - Change the attributes of a *USRSPC

28

List APIs Have a Common Header Which Describes the Contents of the List/*USRSPC



29

The ILE RPG Include File – QUSGEN

- List APIs Have a Common Header Which Describes the Contents of the List/*USRSPC:

```

D*****
D*Type Definition for the User Space Generic Header.
D*****
DQUSH0100      DS
D*
D* Lots removed to fit
D QUSIS          104      104
D*
D* Lots removed to fit
D QUSOLD          125      128B 0
D*
D QUSSLD          129      132B 0
D*
D QUSNBRLE        133      136B 0
D*
D QUSSEE          137      140B 0
D*
D* Lots removed to fit

```

Qus Generic Header

Information Status

Offset List Data

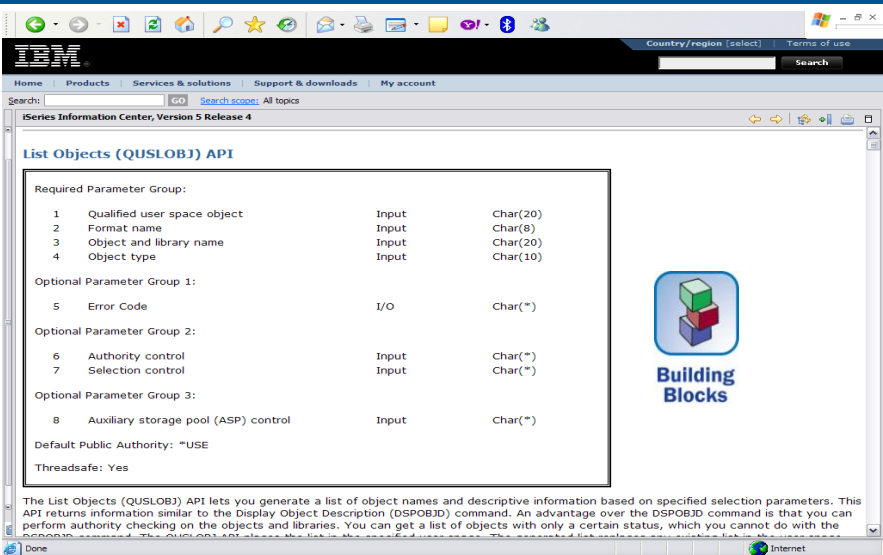
Size List Data

Number List Entries

Size Each Entry

30

List Objects API (QUSLOBJ)



31

List All Files in a Library with List Objects API (QUSLOBJ)

```
h dftactgrp(*no) bnddir('CRTLIB')

dListFile      pr
d LibName      10      const

dListFile      pi
d LibName      10      const

d/copy qsysinc/qrpglesrc,qusgen
d/copy qsysinc/qrpglesrc,quslobj
d/copy qsysinc/qrpglesrc,qusec

dCrtUsrSpc     pr
d SpcName      20      *      extproc('CrtUsrSpc')
d SpcName      20      const

dListObj       pr
d SpcName      20      const
d Format       8      const
d ObjLibName   20      const
d ObjTyp       10      const
d QUSEC        10      const
d AutCtl       65535   const options(*nopass :*varsize)
d SlctCtl      65535   const options(*nopass :*varsize)
d ASPCtl       65535   const options(*nopass :*varsize)
```

Parameter passed to our program – Library name

QSYSINC include files and prototypes for CRTUSRSPC sample and List Object API

32

List All Files in a Library with List Objects API (QUSLOBJ) (cont.)

```

d* list API generic header
dGenHdr          ds
d
                                likeds(QUSH0100)
                                based(GenHdrPtr)
d* List Object API (QUSLOBJ) format OBJL0100
dListEntry       ds
d
                                likeds(QUSL010003)
                                based(LstPtr)
                                Use QSYSINC
                                definitions and
                                based support
                                for easy access

dSpcName          ds
d SName           10    inz('OBJLIST')
d SLib            10    inz('QTEMP')

dObjLibName       ds
d                 10    inz('*ALL')
d ObjLib          10

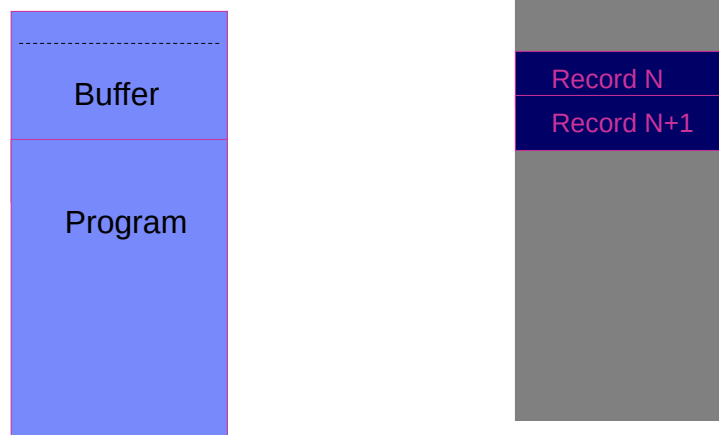
dCount            s      10i 0
dWait             s      1

```

33

Using Traditional Buffer Management

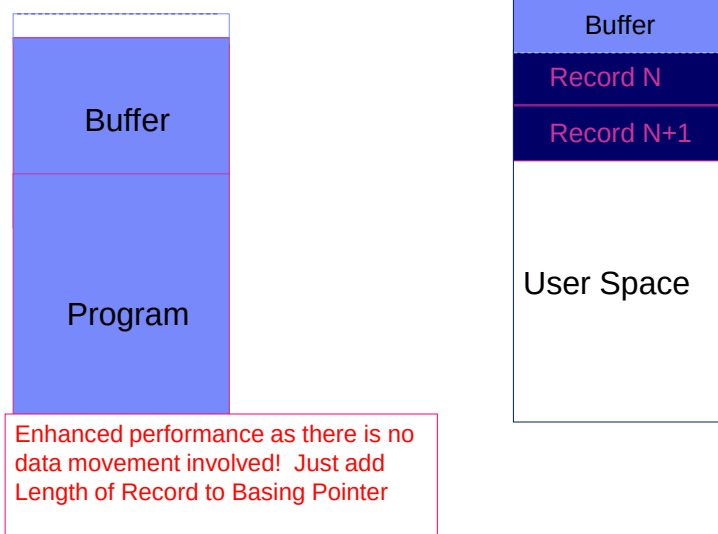
- The Data Moves to Your Program Buffer



34

Using Pointer Based Buffer Management

- The Buffer Addresses the Data Directly



35

List all files in a library with List Objects API (QUSLOBJ)

```

/free

// set ErrCde bytes provided to 0 to cause exceptions
QUSBPRV = 0;

// create the user space for the list of files
GenHdrPtr = CrtUsrSpc(SpcName);

// get the list of files
ObjLib = LibName;
ListObj( SpcName : 'OBJL0100' : ObjLibName : '*FILE' : QUSEC);

// check to see if the list is complete
if (GenHdr.QUSIS = 'C') or (GenHdr.QUSIS = 'P');
    // get to the first list entry and process the list
    LstPtr = GenHdrPtr + GenHdr.QUSOLD;
    for Count = 1 to GenHdr.QUSNBRLE;
        dsply ('File: ' + ListEntry.QUSOBJNU);
        LstPtr = LstPtr + GenHdr.QUSSEE;
    endfor;
endif;
dsply 'End of List' ' ' Wait;
*inlr = *on;
Return;

/end-free

```

'C' = The list is complete
'P' = The list is only partial. More data is available that couldn't fit in 16M bytes

36

Procedure to Create *USRSPCs and return a pointer to the *USRSPC – CRTUSRSPC

```

h nomain

dCrtUsrSpc      pr      *      extproc('CrtUsrSpc')
d SpcName              20      const

pCrtUsrSpc      b              export

dCrtUsrSpc      pi      *
d SpcName              20      const

dCrtSpc         pr              extpgm('QUSCRTUS')
d SpcName              20      const
d SpcAttr              10      const
d SpcSiz          10i 0      const
d SpcVal              1      const
d SpcAut              10      const
d SpcTxt              50      const
d SpcRpl              10      const options(*nopass)
d QUSEC              likeds(QUSEC) options(*nopass)
d SpcDmn              10      const options(*nopass)
d SpcTfrSiz        10i 0      const options(*nopass)
d SpcSpcAln          1      const options(*nopass)

```

37

Procedure to Create *USRSPCs and Return a Pointer to the *USRSPC – CRTUSRSPC (cont.)

```

dRtvSpcPtr      pr              extpgm('QUSPTRUS')
d SpcName              20      const
d UsrSpcPtr      *
d QUSEC              likeds(QUSEC) options(*nopass)

d/copy qsysinc/qrpglesrc,qusec

dUsrSpcPtr      s              *
dSpcLib          s              10

```

38

Procedure to Create *USRSPCs and return a pointer to the *USRSPC – CRTUSRSPC (cont.)

```

/free

// Set Error code bytes provided to 0
QUSBPRV = 0;

// Create the user space
CrtSpc( SpcName :' ' :1 :x'00' :'*CHANGE' :' ' :'*YES' :QUSEC);

// Get a pointer to the user space
RtvSpcPtr( SpcName :UsrSpcPtr :QUSEC);

Return UsrSpcPtr;
/end-free

pCrtUsrSpc      e

```

```

CRTRPGMOD MODULE(CRTSPC)
CRTSRVPGM SRVPGM(CRTSPC) EXPORT(*ALL)
CRTBNDDIR BNDDIR(CRTLIB)
ADDBNDDIRE BNDDIR(CRTLIB) OBJ((CRTSPC))

```

39

What We'll Cover ...

- The Basics
- Retrieve APIs
- List APIs
- Open List APIs
- Unix-type APIs
- Wrap-up

40

Open List APIs

- Open List APIs can generate lists of information in a background task
- Lists are temporary (unlike *USRSPCs on List APIs)
 - Removed when you close the list
 - Removed when the job ends
- Lists can be up to 64G bytes in size
- List APIs will automatically extend the size of the list to accommodate the available data
- APIs exist to:
 - Create a list (lots of these – open list of objects, open list of spooled files, open list of messages, etc.)
 - Get list entries
 - Close a list
 - Find list entries
 - Manage list servers

41

Open List APIs (cont.)

- Have a Common Parameter Which Describes the Contents of the List

The screenshot shows the IBM Series Information Center, Version 5 Release 4, with a table of list fields. The table has columns for Offset, Dec, Hex, Type, and Field. Annotations point to specific fields in the table:

Offset	Dec	Hex	Type	Field
0	0		BINARY(4)	Total records
4	4		BINARY(4)	Records returned
8	8		CHAR(4)	Request handle
12	C		BINARY(4)	Record length
16	10		CHAR(1)	Information complete indicator
17	11		CHAR(13)	Date and time created
30	1E		CHAR(1)	List status indicator
31	1F		CHAR(1)	Reserved
32	20		BINARY(4)	Length of information returned
36	24		BINARY(4)	First record in receiver variable
40	28		CHAR(40)	Reserved

Annotations and their corresponding fields:

- Total records currently available in list (points to Total records)
- Number records in current receiver variable (points to Records returned)
- Length of records in receiver variable (points to Record length)
- Status of list: Complete, In process, etc. (points to List status indicator)
- Relative record number of first record in receiver variable relative to entire list (points to First record in receiver variable)

Plus a Request handle which uniquely identifies the list. This is used to retrieve more list entries later and to close the list when you are done. You can have many open lists active at the same time.

Note

42

Open List APIs (cont.)

- Have a Common Header Which Describes the Contents of the List
 - The ILE RPG Include File – QGY

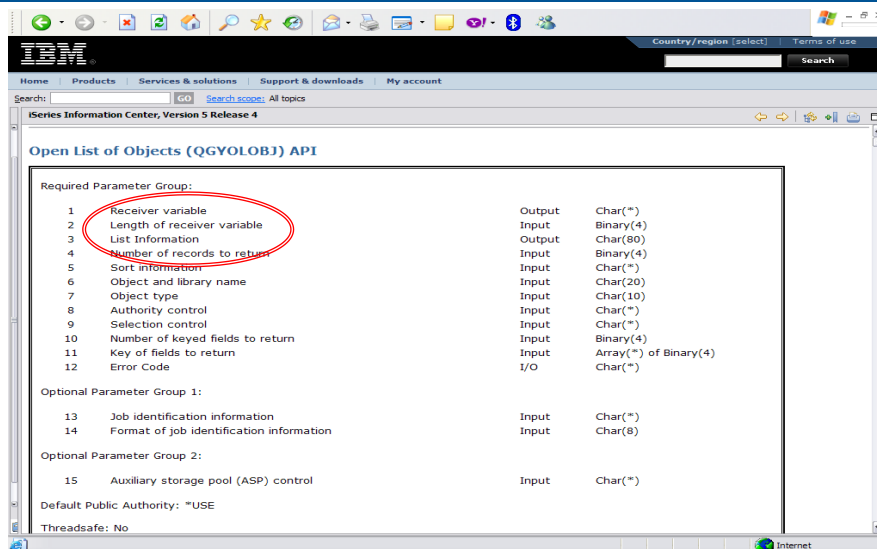
```

D*****
D*List information structure
D*****
DQGYLI          DS
D*
D QGYTR07              1          4B 0          Qgy List Info
D*
D QGYRRTN02            5          8B 0          Total Records
D*
D QGYRH07              9          12          Records Returned
D*
D QGYRL07             13         16B 0          Request Handle
D*
D QGYIC07             17          17          Record Length
D*
D* Date information removed for space reasons
D QGYLSI01            31          31          Info Complete
D*
D QGYERVED56           32          32          List Status Indicator
D*
D QGYRTNIL01           33         36B 0          Reserved
D*
D QGYFBR01             37         40B 0          Returned Info Length
D*
D QGYRSV214            41          80          First Buffer Record
D*
D QGYRSV214            41          80          Reserved2

```

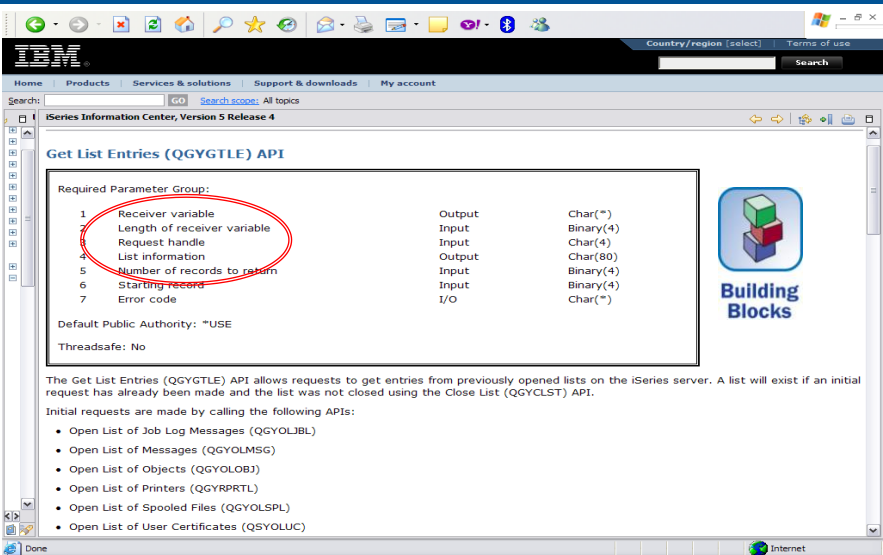
43

Open List of Objects (QGYOLOBJ)



44

Get List Entries (QGYGTLE)



45

List All Files in a Library: Open List Objects API (QGYOLOBJ)

```
h dftactgrp(*no)

dListFile2      pr
d LibName          10      const

dListFile2      pi
d LibName          10      const

d/copy qsysinc/qrpglesrc,qgyolobj
d/copy qsysinc/qrpglesrc,qgy
d/copy qsysinc/qrpglesrc,qusec

dOpnListObj      pr
d RcvVar          1      options(*varsize)
d LenRcvVar       10i 0  const
d ListInfo        80
d NbrRodRqs       10i 0  const
d SortInfo        const likeds (QGYOSI)
d QualObjNam      20      const
d ObjType         10      const
d AuthCtl         const likeds (QGYOAC)
d SelCtl          const likeds (SelCtl)
d NbrKeys         10i 0  const
d KeysToRtn       10i 0  const
d QUSEC           likeds (QUSEC)
d JobID           65535  const options(*nopass :*varsize)
d JobIDFmt        8      const options(*nopass)
d ASPCTL          65535  const options(*nopass :*varsize)
```

Parameter passed to our program – Library name

QSYSINC definitions and API function prototypes

46

List All Files in a Library: Open List Objects API (QGYOLOBJ) (cont.)

```

dGetNextEnt      pr          extpgm('QGYGTLE')
d RcvVar          1          options(*varsize)
d LenRcvVar       10i 0 const
d RqsHandle       4          const
d ListInfo       80         const
d NbrRodRqs       10i 0 const
d StrRod          10i 0 const
d QUSEC           likeds(QUSEC)

dCloseList       pr          extpgm('QGYCLST')
d RqsHandle       4          const
d QUSEC           likeds(QUSEC)

dRcvVar          ds          4096

dRcvVarEntry     ds          based(RcvVarEntPtr)
d               qualified
d Hdr             likeds(QGYORV01)

dSelCtl          ds          qualified
d Ctl             likeds(QGYOSC)
d Status         1

dObjLibName      ds          10 inz('*ALL')
d ObjLib         10

dCount           s          10i 0
dWait            s          1

```

More API prototypes

Based QSYSINC definitions and work fields

47

List All Files in a Library: Open List Objects API (QGYOLOBJ) (cont.)

```

/free

// set ErrCde bytes provided to 0 to cause exceptions
QUSBPRV = 0;

// get ready to call the QGYOLOBJ API
QGYNBRK = 0;
QGYOAC = *loval; // no need to sort API output
QGYFL04 = %size(QGYOAC); // initialize input structure
// no authority controls needed
SelCtl.Ctl = *loval; // initialize selection structure
SelCtl.Ctl.QGYFL05 = %size(SelCtl); // room for one status value
SelCtl.Ctl.QGYSOOS = 0; // select
SelCtl.Ctl.QGYSO01 = 20; // displacement to status
SelCtl.Ctl.QGYNBRS = 1; // number of supplied statuses
SelCtl.Status = ' '; // return entries with no errors
// get the list of files
ObjLib = LibName;
OpnListObj( RcvVar :%size(RcvVar) :QGYLI :50 :QGYOSI :ObjLibName
: '*FILE' :QGYOAC :SelCtl :0 :0 :QUSEC);

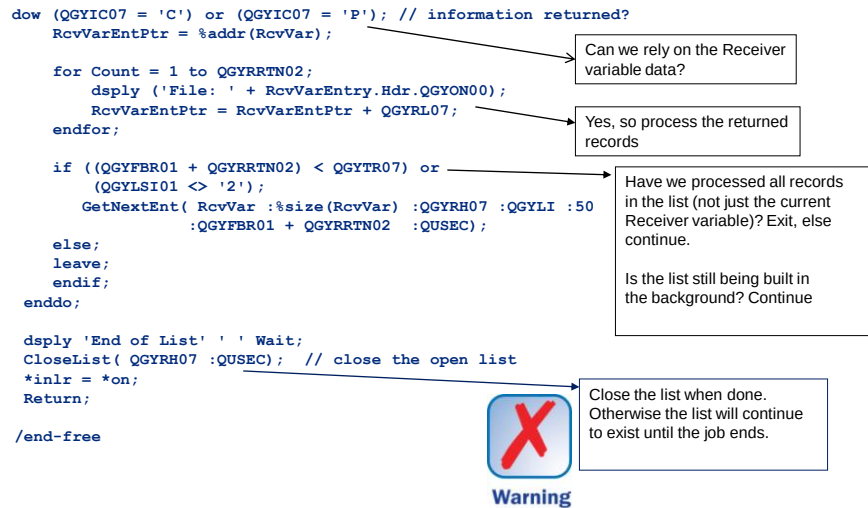
```


Heads-Up

ALWAYS initialize input structures in their entirety, NOT by reserved field names -- what's reserved today can be used in the future ...

48

List All Files in a Library: Open List Objects API (QGYOLOBJ) (cont.)



49

What We'll Cover ...

- The Basics
- Retrieve APIs
- List APIs
- Open List APIs
- Unix-type APIs
- Wrap-up

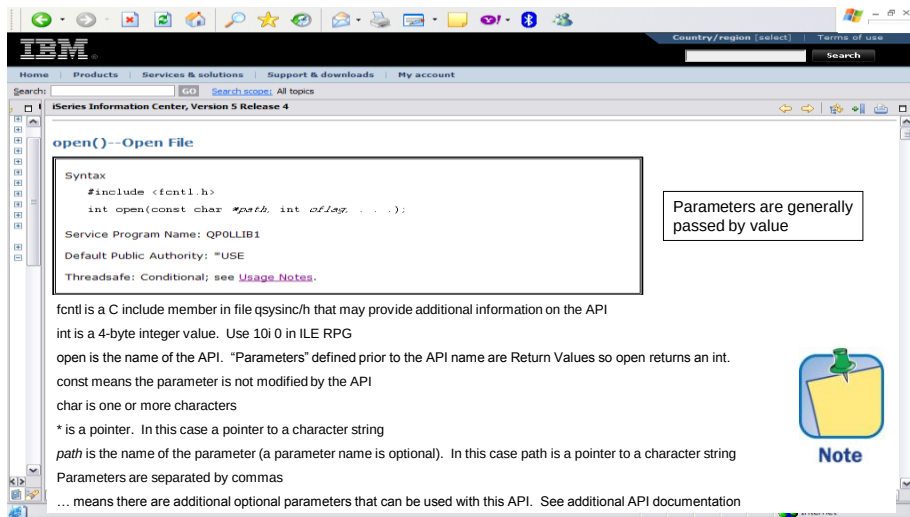
50

Unix-Type APIs

- Provide for cross platform portability
- Follow industry definitions
- Documented using the C programming language
- Generally return errors using an errno variable
- Many families of Unix-Type APIs:
 - Integrated File System
 - Environment variables
 - Sockets
 - Secure sockets
 - Threads
 - Remote Procedure Calls
 - And more ...

51

Reading Unix-Type Parameter Descriptions



The screenshot shows the IBM iSeries Information Center web page for the `open()` API. The page title is "open()--Open File". The syntax section shows the following code:

```
Syntax
#include <fcntl.h>
int open(const char *path, int oflag, ...);
```

Service Program Name: QPOLLIB1
Default Public Authority: *USE
Threadsafe: Conditional; see [Usage Notes](#).

fcntl is a C include member in file `qsysinc/h` that may provide additional information on the API
int is a 4-byte integer value. Use 10i 0 in ILE RPG
open is the name of the API. "Parameters" defined prior to the API name are Return Values so open returns an int.
const means the parameter is not modified by the API
char is one or more characters
* is a pointer. In this case a pointer to a character string
path is the name of the parameter (a parameter name is optional). In this case path is a pointer to a character string
Parameters are separated by commas
... means there are additional optional parameters that can be used with this API. See additional API documentation

A callout box on the right states: "Parameters are generally passed by value". A "Note" icon is located at the bottom right of the page.

52

ILE RPG Function Prototype for Open

```

dopen          pr          10i 0 extproc('open')
d path          *          value options(*string)
d oflag         10i 0 value
d mode          10u 0 value options(*nopass)
d convID        10u 0 value options(*nopass)
d textCrtConvID 10u 0 value options(*nopass)

```

The open API returns an integer value so 10i 0 is used. This is the file descriptor (fd) used by other APIs to reference the opened file. It also serves as an error indicator when -1 is returned

ILE API (and Unix-Type is ILE) names are case sensitive so extproc('open') is specified. If you get an error such as 'CPD5D02 – Definition not found for symbol OPEN' check for the extproc keyword and note that OPEN in the error message is in uppercase while the API name is supposed to be lowercase

path is a pointer to a character string. As Unix-Type APIs generally expect character strings to be terminated by a null byte (x'00') use options(*string)

oflag is a required parameter which defines how the file is to be opened (more on this later)

mode is an optional parameter and options(*nopass) must be specified if the open API is to "see" this parameter

The API documentation for open indicates that mode, conversion ID (convID), text file creation conversion ID (textCrtConvID) are defined as unsigned integer values so 10u 0 is used

All parameters use the value keyword as Unix-Type APIs generally expect pass by value. Note that when a * (pointer) is used as the parameter then passing the name of the variable addressed by the pointer (without value) is the same as passing the pointer by value.

53

Representative C to RPG Datatypes

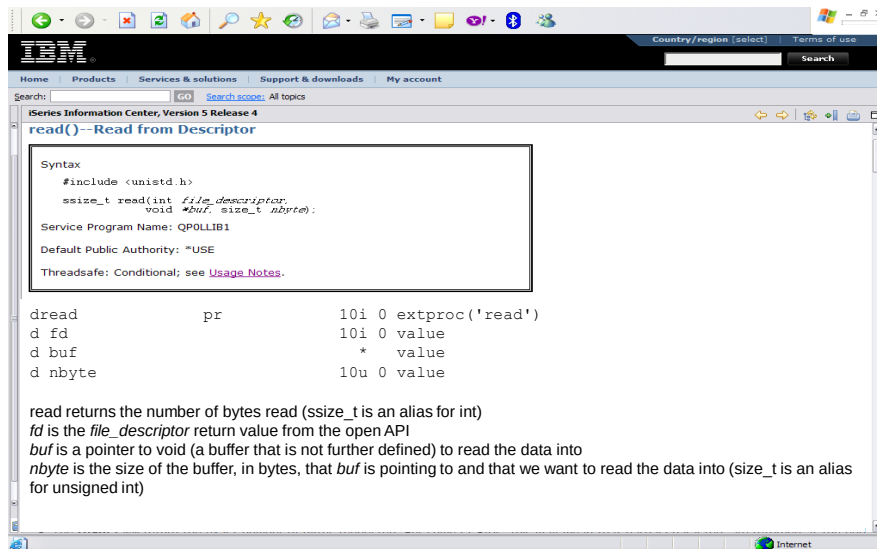


Heads-Up

C definition	RPG definition	Type
short int signed short int	5i 0	2-byte signed integer
unsigned short int	5u 0	2-byte unsigned integer
int long long int signed int	10i 0	4-byte signed integer
unsigned int unsigned long unsigned long int	10u 0	4-byte unsigned integer
float	4f	4-byte floating point
double	8f	8-byte floating point
void	Context sensitive	Nothing as in function(); or undefined
*	*	Pointer
char *	* value options(*string)	Pointer to a null terminated input character string
int *	* value	Pointer to a 4-byte integer

54

read – Read from Descriptor



The screenshot shows the IBM iSeries Information Center, Version 5 Release 4, displaying the documentation for the `read()` function. The page includes a search bar, navigation tabs (Home, Products, Services & solutions, Support & downloads, My account), and a search scope dropdown. The main content area is titled "read()--Read from Descriptor".

Syntax

```
#include <unistd.h>
ssize_t read(int file_descriptor,
             void *buf, size_t nbytes);
```

Service Program Name: QPOLLIB1
Default Public Authority: *USE
Threadsafe: Conditional; see [Usage Notes](#).

dread pr 10i 0 extproc('read')

d fd 10i 0 value

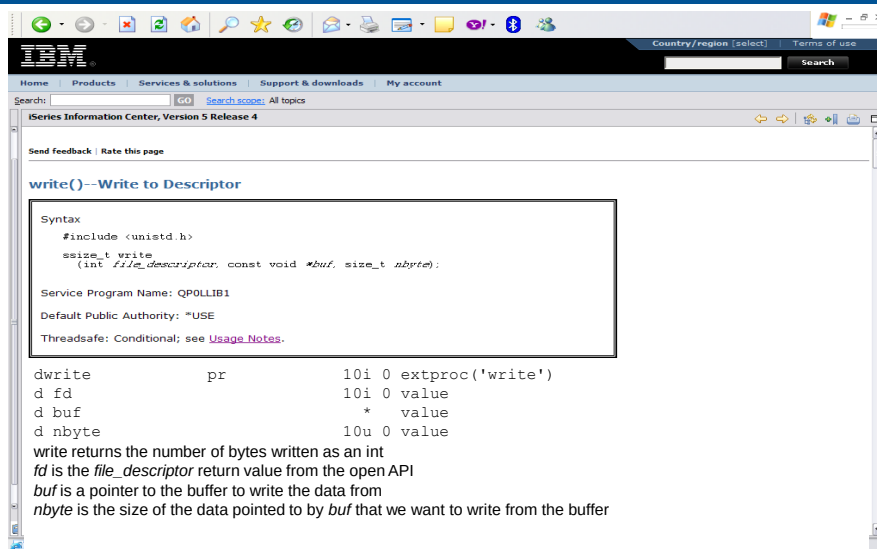
d buf * value

d nbytes 10u 0 value

read returns the number of bytes read (ssize_t is an alias for int)
fd is the file_descriptor return value from the open API
buf is a pointer to void (a buffer that is not further defined) to read the data into
nbyte is the size of the buffer, in bytes, that buf is pointing to and that we want to read the data into (size_t is an alias for unsigned int)

55

write – Write to Descriptor



The screenshot shows the IBM iSeries Information Center, Version 5 Release 4, displaying the documentation for the `write()` function. The page includes a search bar, navigation tabs (Home, Products, Services & solutions, Support & downloads, My account), and a search scope dropdown. The main content area is titled "write()--Write to Descriptor".

Syntax

```
#include <unistd.h>
ssize_t write(
    int file_descriptor, const void *buf, size_t nbytes);
```

Service Program Name: QPOLLIB1
Default Public Authority: *USE
Threadsafe: Conditional; see [Usage Notes](#).

dwrite pr 10i 0 extproc('write')

d fd 10i 0 value

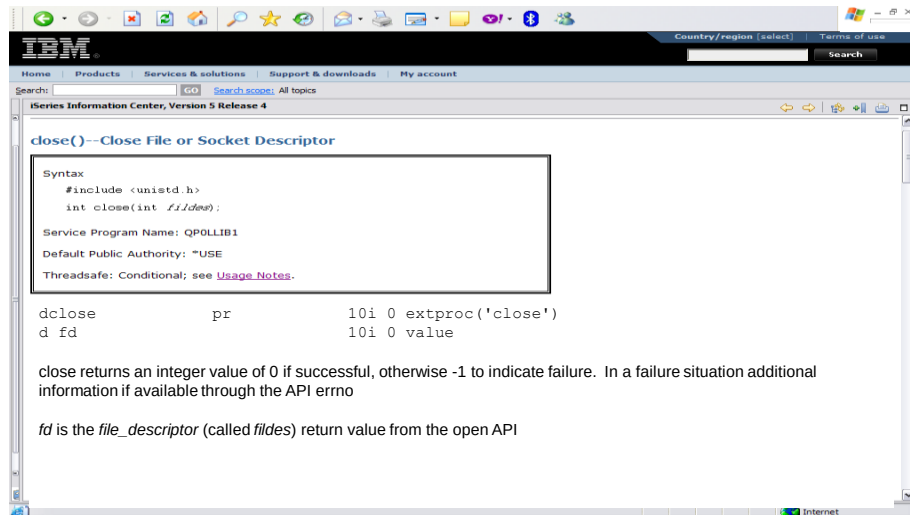
d buf * value

d nbytes 10u 0 value

write returns the number of bytes written as an int
fd is the file_descriptor return value from the open API
buf is a pointer to the buffer to write the data from
nbyte is the size of the data pointed to by buf that we want to write from the buffer

56

close – Close File or Socket Descriptor



57

What if an Error is Found?

■ errno provides some additional information

- errno is actually another API which returns an additional error information

```
derrno          pr          *   extproc('__errno')
```

- errno returns a pointer to an integer (10i 0) so you would also define:

```
dErrorNbr       s          10i 0 based(ErrorNPtr)
```

in order to access the ErrorNbr based off of the returned pointer. Each ErrorNbr that the Unix-Type API might return can be found in the API documentation

- The value of ErrorNbr, nnnn, maps to the message CPEnnnn in QCPFMSG so ErrorNbr 3021 is also CPE3021
- You can also retrieve the text for the message using the strerror API:

```
dstrerror        pr          *   extproc('strerror')
d ErrorNbr       10i 0 value
```

- strerror returns a pointer to a null terminated character string describing the error
- Note that the text for these errors tends to be very vague... CPE3021 for instance is 'The value specified for the argument is not correct.' You will learn to appreciate all the CPF messages i5/OS provides!



These two APIs (errno and strerror) are part of C runtime -- you need to specify BNDDIR('QC2LE') when compiling prior to V6R1



58

A Program to Use an Integrated File System File

■ The Prototypes:

```
h_dftactgrp(*no) bnddir('QC2LE') ← QC2LE needed for errno prior to V6R1

dopen          pr          10i 0 extproc('open')
d_path         *          value options(*string)
d_oflag        10i 0 value
d_mode         10u 0 value options(*nopass)
d_convID       10u 0 value options(*nopass)
d_textCrtConvID 10u 0 value options(*nopass)

dread          pr          10i 0 extproc('read')
d_fd           10i 0 value
d_buf          *          value
d_nbyte        10u 0 value

dwrite         pr          10i 0 extproc('write')
d_fd           10i 0 value
d_buf          *          value
d_nbyte        10u 0 value

dclose         pr          10i 0 extproc('close')
d_fd           10i 0 value

derrno         pr          *    extproc('__errno')
```

59

A Program to Use an Integrated File System File (cont.)

■ Working Variables and Constants:

```
dErrorNbr      s          10i 0 based(ErrnoPtr)
dfd            s          10i 0
dSomeText      s          50   inz('Hello World')
dReturnedData  s          30
dwait          s          1

dO_RDONLY      c          1
dO_RDWR       c          4
dO_CREAT       c          8
dO_EXCL        c          16
dO_CCSD        c          32
dO_TEXTDATA    c          16777216
dO_TEXT_CREAT  c          33554432

dS_IRGRP       c          32
dS_IWUSR       c          128
dS_IRUSR       c          256
```

These are constants that are defined in the `fcntl` C header file

They are used to set the *oflag* and *mode* options of how we want the file to be opened

60

Understanding C Constants

- In member fcntl of QSYSINC/H:

- #define O_RDONLY 00001
- #define O_TEXTDATA 0100000000

- In RPG program:

- dO_RDONLY c 1
- dO_TEXTDATA c 16777216



Heads-Up

- Why the difference?

- The C definition is in octal!
 - ▶ Values that start with a 0 (zero) use an octal base
 - ▶ Values that start with a non-0 numeric use a decimal base
 - ▶ Values that start with an x use a hexadecimal base

- How to go from octal to decimal?

61

Simple Octal to Decimal RPG Program

- There is an API strtol but this is an easier example

```
dOctal      s          15
dValue      s          10u 0
dX          s          3u 0
dChar       s          1
dWait       s          1
dError      s          n
```



Tool

```
/free
dsply 'Enter octal number' ' ' Octal;           // Get octal value
```

62

Simple Octal to Decimal RPG Program (cont.)

```

for X = 1 to %size(Octal);
    Char = %subst(Octal :X :1);
    if Char <> ' ';
        if ((Char < '0') or (Char > '7'));
            dsply ('Invalid digit ' + Char + ' found');
            Error = *on;
        else;
            if Error = *off;
                Monitor;
                Value = (Value * 8) + %int(Char); // raise 1 power
                On-Error 00103; // overflow error
                dsply 'Value too large';
                Error = *on;
            EndMon;
            endif;
        endif;
    endif;
endfor;
if Error = *off;
    dsply Value ' ' Wait;
else;
    dsply 'Errors found' ' ' Wait;
endif;
*inlr = *on;
return;
/end-free

```

63

A Program to Use an Integrated File System File

```

/free
fd = open( '/COMMON_File.txt'
           :O_RDWR + O_CREAT + O_EXCL + O_CSID + O_TEXTDATA +
           O_TEXT_CREAT
           :S_IRUSR + S_IWUSR + S_IRGRP
           :0:0);
if (fd = -1);
    ErrnoPtr = errno;
    dsply ('Failure ' + %char(ErrorNbr) + ' on open') ' ' wait;
    *inlr = *on;
    return;
endif;

if write(fd :%addr(SomeText) :%len(%trimr(SomeText)))
    < %len(%trimr(SomeText));
    dsply 'Did not write all data' ' ' wait;
else;
    if close(fd) = -1;
        ErrnoPtr = errno;
        dsply ('Failure ' + %char(ErrorNbr) + ' on close')
            ' ' wait;
        *inlr = *on;
        return;
    endif;
endif;

```

Create a new file with data to be tagged with the jobs default job CCSID

If an error display the errno.

If errno is 3457 (file exists) then remove the link or open with different options (not shown). There is an unlink API (along with many others).

Write SomeText to the file.

If the return value from write is < the size of data written then there was an error and we will report the error.

Otherwise close the file and check for an error on the close.

64

What is

:O_RDWR + O_CREAT + O_EXCL + O_CSID + O_TEXTDATA + O_TEXT_CREAT ???

Variable Name	Decimal Value	Bit Value
&O_RDWR	4	00000000 00000000 00000000 00000100
&O_CREAT	8	00000000 00000000 00000000 00001000
&O_EXCL	16	00000000 00000000 00000000 00010000
&O_CSID	32	00000000 00000000 00000000 00100000
&O_TEXTDATA	16777216	00000001 00000000 00000000 00000000
&O_TEXT_CREAT	33554432	00000010 00000000 00000000 00000000
Sum:		00000011 00000000 00000000 00111100

32 separate flags in one 4-byte integer field

The API then examines each bit (or indicator) for whether it's on or off to control processing

65

A Program to Use an Integrated File System File

```
fd = open( '/COMMON_File.txt'
           :O_RDONLY + O_TEXTDATA );
if (fd = -1);
    ErrnoPtr = errno;
    dsply ('Failure ' + %char(ErrorNbr) + ' on 2nd open') ' ' wait;
    *inlr = *on;
    return;
endif;

if read(fd :%addr(ReturnedData) :%size(ReturnedData))
    < %len(%trimr(SomeText));
    dsply 'Not all data returned' ' ' wait;
else;
    dsply ReturnedData ' ' wait;
endif;

fd = close(fd);
*inlr = *on;
/end-free
```

Reopen the file for read and report if there is an error

Read the file. The return value for the read API is the number of bytes read.

If we didn't read all of the data we had previously written, then report the error.

Otherwise display the data.

66

What We'll Cover ...

- The Basics
- Retrieve APIs
- List APIs
- Open List APIs
- Unix-type APIs
- Wrap-up

67

Resources

- Bruce Vining, Doug Pence, Ron Hawkins *APIs at Work, Second Edition* (MC Press, 2007).
 - All sample programs in free form RPG, fixed form RPG, and COBOL
- [MC RPG Developer](#)
 - Monthly column oriented toward RPG developers
- [MC Tips 'n Techniques](#)
 - Monthly column oriented toward CL developers
- [System i News](#)
 - 4-part series "Industry Standard API Basics for the CL Programmer"
 - Applicable also to RPG and COBOL developers (who know CL)
- http://publib.boulder.ibm.com/infocenter/systems/scope/i5os/topic/apiref/apiconcept.htm?tocNode=int_169218
 - API concepts in IBM Information Center

68

7 Key Points to Take Home

- APIs are different than language opcodes found in languages such as RPG, COBOL, and CL but have common building blocks
- Never assume, guess, or hardcode when using APIs, the information is there
 - Do not assume returned values on 'else' legs. Check them
 - Use provided offsets and lengths
 - Do not hardcode receiver variable lengths
- Always check for errors if exception messages are turned off
- APIs are not difficult but do have a learning curve. The curve is worth it if you want the most out of your system

69

Your Turn!



Questions?

How to contact me:
Bruce Vining
bvining@brucevining.com

70

Examples Using Other Languages

QSPRJOBQ – A Sample Retrieve API Retrieve Job Queue Information ILE COBOL Include

```
*****
*Type Definition for the JOBQ0100 format.
*****
01  QSP-JOBQ0100.
   05  BYTES-RETURNED                PIC S9(00009) BINARY.
   05  BYTES-AVAILABLE                PIC S9(00009) BINARY.
   05  JOB-QUEUE-NAME                 PIC X(00010).
   05  JOB-QUEUE-LIB-NAME              PIC X(00010).
   05  OPERATOR-CONTROLLED              PIC X(00010).
   05  AUTHORITY-CHECK                  PIC X(00010).
   05  NUMBER-JOBS                     PIC S9(00009) BINARY.
```

QSPRJOBQ – A Sample Retrieve API
Retrieve Job Queue Information ILE COBOL Program to
Display the Number of Jobs on a JOBQ

```
IDENTIFICATION DIVISION.
PROGRAM-ID. RJOBQCBL.

DATA DIVISION.
WORKING-STORAGE SECTION.
COPY QUSEC OF QSYSINC-QCBLLESRC.
COPY QSPRJOBQ OF QSYSINC-QCBLLESRC.
01 JOBQ.
   05 JobQName          PIC X(10) VALUE "QBATCH".
   05 JobQLib           PIC X(10) VALUE "*LIBL".

PROCEDURE DIVISION.
MAIN-LINE.
   MOVE 0 TO BYTES-PROVIDED OF QUS-EC.
   CALL QSPRJOBQ USING BY REFERENCE QSP-JOBQ0100,
                      BY CONTENT  LENGTH OF QSP-JOBQ0100,
                      BY CONTENT  "JOBQ0100",
                      BY CONTENT  JOBQ,
                      BY REFERENCE QUS-EC.
   DISPLAY NUMBER-JOBS OF QSP-JOBQ0100.
   STOP RUN.
```

The Error Code Structure

ILE COBOL Include

```
*****
*Record structure for Error Code Parameter
****                                     ***
*NOTE: The following type definition only defines the fixed
*   portion of the format. Varying length field Exception
*   Data will not be defined here.
*****
01 QUS-EC.
   05 BYTES-PROVIDED          PIC S9(00009) BINARY.
   05 BYTES-AVAILABLE        PIC S9(00009) BINARY.
   05 EXCEPTION-ID           PIC X(00007).
   05 RESERVED               PIC X(00001).
*   05 EXCEPTION-DATA        PIC X(00001).
```

QSPRJOBQ – A Sample Retrieve API
Retrieve Job Queue Information ILE COBOL Program to
Display the Number of Jobs on a JOBQ

```

IDENTIFICATION DIVISION.
PROGRAM-ID. RJOBQCBL.

DATA DIVISION.
WORKING-STORAGE SECTION.
COPY QUSEC OF QSYSINC-QCBLLESRC.
COPY QSPRJOBQ OF QSYSINC-QCBLLESRC.
01 JOBQ.
   05 JobQName          PIC X(10) VALUE "QBATCH".
   05 JobQLib           PIC X(10) VALUE "*LIBL".

PROCEDURE DIVISION.
MAIN-LINE.
   MOVE 0 TO BYTES-PROVIDED OF QUS-EC.
   CALL QSPRJOBQ USING BY REFERENCE QSP-JOBQ0100,
                      BY CONTENT  LENGTH OF QSP-JOBQ0100,
                      BY CONTENT  "JOBQ0100",
                      BY CONTENT  JOBQ,
                      BY REFERENCE QUS-EC.
   DISPLAY NUMBER-JOBS OF QSP-JOBQ0100.
   STOP RUN.

```

QSPRJOBQ – A Sample Retrieve API
Retrieve Job Queue Information ILE COBOL Program to
Display the Number of Jobs on a JOBQ – Check for Errors

```

PROCESS NOMONOPRC.
IDENTIFICATION DIVISION.
PROGRAM-ID. RJOBQCBL2.

DATA DIVISION.
WORKING-STORAGE SECTION.
COPY QUSEC OF QSYSINC-QCBLLESRC REPLACING
   ==01 QUS-EC==
   BY ==01 QUS-EC IS TYPEDEF==.
COPY QSPRJOBQ OF QSYSINC-QCBLLESRC.

01 JobQ.
   05 JobQName          PIC X(10) VALUE "QBATCH".
   05 JobQLib           PIC X(10) VALUE "*LIBL".

01 ErrCde.
   05 Error-Format      TYPE QUS-EC.
   05 Error-Data        PIC X(100).

```

QSPRJOBQ – A Sample Retrieve API

Retrieve Job Queue Information ILE COBOL Program to Display the Number of Jobs on a JOBQ – Check for Errors

```

PROCEDURE DIVISION.
MAIN-LINE.
    MOVE LENGTH OF ErrCde TO Bytes-Provided OF ErrCde.
    CALL QSPRJOBQ USING BY REFERENCE QSP-JOBQ0100,
                        BY CONTENT  LENGTH OF QSP-JOBQ0100,
                        BY CONTENT  "JOBQ0100",
                        BY CONTENT  JobQ,
                        BY REFERENCE ErrCde.
    IF Bytes-Available OF ErrCde > 0
        IF Exception-ID OF ErrCde = "CPF3307"
            DISPLAY "Error " Exception-ID OF ErrCde
                " on JobQ " Error-Data(1:10),
        ELSE
            DISPLAY "Error " Exception-ID OF ErrCde " returned"
    ELSE
        DISPLAY NUMBER-JOBS OF QSP-JOBQ0100.
    STOP RUN.

```

QDCRCTLD – Retrieve Controller Description

ILE COBOL Include for Format CTLD0200

```

01 QDC-CTLD0200.
05 CTLD0100.
    09 BYTES-RETURNED          PIC S9(00009) BINARY.
    09 BYTES-AVAILABLE         PIC S9(00009) BINARY.
    09 NUM-ATTACHED-DEVICES    PIC S9(00009) BINARY.
    09 DATE-INFO-RETRIEVED     PIC X(00007) .
    09 TIME-INFO-RETRIEVED     PIC X(00006) .
    09 CONTROLLER-NAME         PIC X(00010) .
    09 CONTROLLER-CATEGORY     PIC X(00010) .
    09 ONLINE-AT-IPL           PIC X(00010) .
    09 TEXT-DESC                PIC X(00050) .
    09 RESERVED                PIC X(00003) .
    05 OFFSET-ATTACHED-DEVICES PIC S9(00009) BINARY.
    05 LENGTH-ATTACHED-DEVICES PIC S9(00009) BINARY.

```

QDCRCTL D – Retrieve Controller Description ILE COBOL Include for Format CTLD0200 attached devices

```

01 QDC-LIST-ATTACHED-DEVICES.
   05 ATTACHED-DEVICE-NAME PIC X(00010) .
   05 DEVICE-CATEGORY      PIC X(00010) .
   05 DEVICE-TYPE          PIC X(00010) .
   05 DEVICE-TEXT-DESC     PIC X(00050) .

```

QDCRCTL D – Retrieve Controller Description Retrieve the devices

```

PROCESS NOMONOPRC.
IDENTIFICATION DIVISION.
* CRTBND CBL BNDDIR (QC2LE)
PROGRAM-ID. RCTLDCBL.

DATA DIVISION.
WORKING-STORAGE SECTION.
COPY QUSEC OF QSYSINC-QCBLLESRC.

01 Receiver-Size PIC S9(09) BINARY VALUE 8.
01 Ctld          PIC X(10) VALUE "CTL01".
01 Counter       PIC S9(09) BINARY.
01 Base-Ptr      POINTER.
01 Device-Ptr    POINTER.

LINKAGE SECTION.
COPY QDCRCTL D OF QSYSINC-QCBLLESRC.

```

Do not monospace procedure names

Use the QC2LE *BNDDIR to access the C-runtime library functions for memory management: malloc, realloc, and free.

The QDCRCTL D structures will be based.

QDCRCTL D – Retrieve Controller Description

Retrieve the devices

PROCEDURE DIVISION.
MAIN-LINE.

```

MOVE 0 TO Bytes-Provided OF QUS-EC.
CALL LINKAGE PRC "malloc" USING BY VALUE Receiver-Size
    RETURNING Base-Ptr.
SET ADDRESS OF QDC-CTLD0200 TO Base-Ptr.
CALL "QDCRCTL D" USING BY REFERENCE QDC-CTLD0200,
    BY REFERENCE Receiver-Size,
    BY CONTENT "CTLD0200",
    BY CONTENT CtlD,
    BY REFERENCE QUS-EC.
PERFORM TEST BEFORE UNTIL Bytes-Available OF QDC-CTLD0200
    <= Receiver-Size
    CALL LINKAGE PRC "realloc"
        USING BY VALUE Base-Ptr,
        BY VALUE Bytes-Available OF QDC-CTLD0200
        RETURNING Base-Ptr,
    MOVE Bytes-Available OF QDC-CTLD0200
        TO Receiver-Size,
    SET ADDRESS OF QDC-CTLD0200 TO Base-Ptr,
    CALL "QDCRCTL D" USING
        BY REFERENCE QDC-CTLD0200,
        BY REFERENCE Receiver-Size,
        BY CONTENT "CTLD0200",
        BY CONTENT CtlD,
        BY REFERENCE QUS-EC
END-PERFORM.

```

malloc, realloc, and free
are part of C runtime so
BNDDIR(QC2LE) should
be used on CRTBNDCBL

QDCRCTL D – Retrieve Controller Description

Retrieve the devices

```

IF Num-Attached-Devices OF QDC-CTLD0200 > 0
    SET Device-Ptr TO Base-Ptr,
    SET Device-Ptr UP BY Offset-Attached-Devices
        OF QDC-CTLD0200,
    SET ADDRESS OF QDC-LIST-ATTACHED-DEVICES TO Device-Ptr,
    PERFORM Num-Attached-Devices OF QDC-CTLD0200 TIMES
        DISPLAY Attached-Device-Name
            OF QDC-LIST-ATTACHED-DEVICES,
        SET Device-Ptr UP BY Length-Attached-Devices
            OF QDC-CTLD0200,
        SET ADDRESS OF QDC-LIST-ATTACHED-DEVICES TO
            Device-Ptr
    END-PERFORM.
CALL LINKAGE PRC "free" USING BY VALUE Base-Ptr.
STOP RUN.

```

Procedure to Create *USRSPCs and return a pointer to the *USRSPC - COBOL

```

PROCESS NOMONOPRC.

IDENTIFICATION DIVISION.
PROGRAM-ID. "CrtUsrSpc".

DATA DIVISION.
WORKING-STORAGE SECTION.
COPY QUSEC OF QSYSINC-QCBLLESRC.

01 Initial-Size                PIC S9(9) BINARY VALUE 1.
01 Space-Description           PIC X(50) VALUE SPACES.
01 User-Space-Pointer          POINTER.

LINKAGE SECTION.
01 Qualified-Space-Name        PIC X(20) .

```

Procedure to Create *USRSPCs and return a pointer to the *USRSPC - COBOL

```

PROCEDURE DIVISION USING Qualified-Space-Name
                        RETURNING User-Space-Pointer.

MAIN-LINE.
    MOVE 0 TO Bytes-Provided OF QUS-EC.
    * Create the user space
    CALL "QUSCRTUS" USING
        BY REFERENCE Qualified-Space-Name,
        BY CONTENT    "          ",
        BY REFERENCE Initial-Size,
        BY CONTENT    x"00",
        BY CONTENT    "*CHANGE  ",
        BY REFERENCE Space-Description,
        BY CONTENT    "*YES      ",
        BY REFERENCE QUS-EC.

    * Get a pointer to the user space
    CALL "QUSPTRUS" USING
        BY REFERENCE Qualified-Space-Name,
        BY REFERENCE User-Space-Pointer,
        BY REFERENCE QUS-EC.

    GOBACK.

```

List APIs Have a Common Header Which Describes the Contents of the List/*USRSPC The ILE COBOL Include File - QUSGEN

```
*****
*Type Definition for the User Space Generic Header.
*****
01 QUS-GENERIC-HEADER-0100.
*
* Lots removed to fit
*
    05 INFORMATION-STATUS                PIC X(00001) .
*
* Lots removed to fit
*
    05 OFFSET-LIST-DATA                  PIC S9(00009) BINARY.
    05 SIZE-LIST-DATA                    PIC S9(00009) BINARY.
    05 NUMBER-LIST-ENTRIES                PIC S9(00009) BINARY.
    05 SIZE-EACH-ENTRY                    PIC S9(00009) BINARY.
*
* Lots removed to fit
*
```

List Objects API (QUSLOBJ) List all files in a library - COBOL

```
PROCESS NOMONOPRC.

IDENTIFICATION DIVISION.
* CRTBNDCBL BNDDIR(CRTLIB)
PROGRAM-ID. LISTFILE.

DATA DIVISION.
WORKING-STORAGE SECTION.
COPY QUSEC OF QSYSINC-QCBLLESRC.
01 Space-Name.
    05 Spc-Name                PIC X(10) VALUE "OBJLIST".
    05 Spc-Library              PIC X(10) VALUE "QTEMP".
01 Object-Library-Name.
    05                          PIC X(10) VALUE "*ALL".
    05 Object-Library           PIC X(10) .
01 Generic-Header-Pointer      POINTER.
01 List-Pointer                POINTER.

LINKAGE SECTION.
COPY QUSGEN OF QSYSINC-QCBLLESRC.
COPY QUSLOBJ OF QSYSINC-QCBLLESRC.
01 Library-Name                PIC X(10) .
```

List Objects API (QUSLOBJ)

List all files in a library - COBOL

```

PROCEDURE DIVISION USING Library-Name.
MAIN-LINE.
    MOVE 0 TO BYTES-PROVIDED OF QUS-EC.
    CALL LINKAGE PRC "CrtUsrSpc" USING
        BY REFERENCE Space-Name
        RETURNING   Generic-Header-Pointer.
    MOVE Library-Name TO Object-Library OF Object-Library-Name.
    CALL "QUSLOBJ" USING BY REFERENCE Space-Name,
        BY CONTENT "OBJL0100",
        BY REFERENCE Object-Library-Name,
        BY CONTENT "*FILE      ",
        BY REFERENCE QUS-EC.
    SET ADDRESS OF QUS-GENERIC-HEADER-0100 TO
        Generic-Header-Pointer.

```

List Objects API (QUSLOBJ)

List all files in a library - COBOL

```

IF INFORMATION-STATUS OF QUS-GENERIC-HEADER-0100 = "C"
OR INFORMATION-STATUS OF QUS-GENERIC-HEADER-0100 = "P"
    SET List-Pointer TO Generic-Header-Pointer,
    SET List-Pointer UP BY
        OFFSET-LIST-DATA OF QUS-GENERIC-HEADER-0100,
    SET ADDRESS OF QUS-OBJL0100 TO List-Pointer,
    PERFORM NUMBER-LIST-ENTRIES
        OF QUS-GENERIC-HEADER-0100 TIMES
        DISPLAY "File: " OBJECT-NAME-USED
            OF QUS-OBJL0100,
    SET List-Pointer UP BY SIZE-EACH-ENTRY
        OF QUS-GENERIC-HEADER-0100,
    SET ADDRESS OF QUS-OBJL0100 TO List-Pointer
    END-PERFORM.
DISPLAY "End of List".
STOP RUN.

```

Open List Objects API (QGYOLOBJ)

List all files in a library - COBOL

```

PROCESS NOMONOPRC.
IDENTIFICATION DIVISION.
PROGRAM-ID. LISTFILE2.
* CRTBNDCBL BNDDIR(QC2LE)
DATA DIVISION.
WORKING-STORAGE SECTION.
COPY QGYOLOBJ OF QSYSINC-QCBLLESRC REPLACING
    ==01 QGY-OLOBJ-SELCONTROL==
        BY ==01 QGY-OLOBJ-SELCONTROL IS TYPEDEF==
    ==01 QGY-OLOBJ-RECVAR==
        BY ==01 QGY-OLOBJ-RECVAR IS TYPEDEF==.
COPY QUSEC OF QSYSINC-QCBLLESRC.
01 Receiver                      PIC X(4096).
01 Number-Of-Records             PIC S9(9) BINARY VALUE 50.
01 Next-Record                   PIC S9(9) BINARY.
01 Object-Library-Name.
    05                           PIC X(10) VALUE "*ALL".
    05 Object-Library            PIC X(10).
01 Keys-To-Return                PIC S9(9) BINARY VALUE 0.
01 Keys                          PIC S9(9) BINARY VALUE 0.
01 Sel-Pointer                   POINTER.
01 List-Pointer                  POINTER.
LINKAGE SECTION.
01 Library-Name                  PIC X(10).
01 Sel-Control.
    05 Select-Format             TYPE QGY-OLOBJ-SELCONTROL.
    05 Status-Field              PIC X(01).
01 List-Entry                    TYPE QGY-OLOBJ-RECVAR.

```

Open List Objects API (QGYOLOBJ)

List all files in a library - COBOL

```

PROCEDURE DIVISION USING Library-Name.
MAIN-LINE.
    MOVE 0 TO BYTES-PROVIDED OF QUS-EC.
    MOVE Library-Name TO Object-Library OF Object-Library-Name.
    MOVE 0 TO NUM-KEYS OF QGY-OLOBJ-SORTINFO.
    MOVE LOW-VALUES TO QGY-OLOBJ-AUTHCONTROL.
    MOVE LENGTH OF QGY-OLOBJ-AUTHCONTROL TO
        FORMAT-LENGTH OF QGY-OLOBJ-AUTHCONTROL.
    CALL LINKAGE PRC "malloc" USING
        BY VALUE LENGTH OF Sel-Control
        RETURNING Sel-Pointer.
    SET ADDRESS OF Sel-Control TO Sel-Pointer.
    MOVE LOW-VALUES TO Sel-Control.
    MOVE LENGTH OF Sel-Control TO
        FORMAT-LENGTH OF Sel-Control.
    MOVE 0 TO SEL-OR-OMIT-STATUS OF Sel-Control.
    MOVE LENGTH OF QGY-OLOBJ-SELCONTROL TO
        STATUS-OFFSET OF Sel-Control.
    MOVE 1 TO NUM-STATUS OF Sel-Control.
    MOVE " " TO Status-Field OF Sel-Control.

```

Open List Objects API (QGYOLOBJ)

List all files in a library - COBOL

```

CALL "QGYOLOBJ" USING BY REFERENCE Receiver,
                      BY CONTENT  LENGTH OF Receiver,
                      BY REFERENCE QGY-OLOBJ-LISTINFO,
                      BY REFERENCE Number-Of-Records,
                      BY REFERENCE QGY-OLOBJ-SORTINFO,
                      BY REFERENCE Object-Library-Name,
                      BY CONTENT  "*FILE      ",
                      BY REFERENCE QGY-OLOBJ-AUTHCONTROL,
                      BY REFERENCE Sel-Control,
                      BY REFERENCE Keys-To-Return,
                      BY REFERENCE Keys,
                      BY REFERENCE QUS-EC.
IF INFO-COMPLETE OF QGY-OLOBJ-LISTINFO = "C"
OR INFO-COMPLETE OF QGY-OLOBJ-LISTINFO = "P"
  PERFORM DISPLAY-FILES-FOUND
ELSE
  DISPLAY "Error encountered",
  STOP RUN.
PERFORM GET-NEXT-FILES WITH TEST BEFORE UNTIL
  FIRST-RECORD OF QGY-OLOBJ-LISTINFO +
  RECORDS-RETD OF QGY-OLOBJ-LISTINFO
  GREATER THAN TOTAL-RECORDS OF QGY-OLOBJ-LISTINFO
  AND LIST-STATUS OF QGY-OLOBJ-LISTINFO EQUAL 2.
DISPLAY "End of List".
CALL "QGYCLST" USING
  BY REFERENCE REQUEST-HANDLE
  OF QGY-OLOBJ-LISTINFO,
  BY REFERENCE QUS-EC.
STOP RUN.

```

Open List Objects API (QGYOLOBJ)

List all files in a library - COBOL

```

DISPLAY-FILES-FOUND.
  SET List-Pointer TO ADDRESS OF Receiver,
  PERFORM RECORDS-RETD OF QGY-OLOBJ-LISTINFO TIMES
  SET ADDRESS OF List-Entry TO List-Pointer,
  DISPLAY "File: " OBJ-NAME OF List-Entry,
  SET List-Pointer UP BY RECORD-LENGTH
  OF QGY-OLOBJ-LISTINFO,
  END-PERFORM.
GET-NEXT-FILES.
  COMPUTE Next-Record EQUAL
  FIRST-RECORD OF QGY-OLOBJ-LISTINFO
  + RECORDS-RETD OF QGY-OLOBJ-LISTINFO.
CALL "QGYGTLE" USING
  BY REFERENCE Receiver,
  BY CONTENT  LENGTH OF Receiver,
  BY REFERENCE REQUEST-HANDLE
  OF QGY-OLOBJ-LISTINFO,
  BY REFERENCE QGY-OLOBJ-LISTINFO,
  BY REFERENCE Number-Of-Records,
  BY REFERENCE Next-Record,
  BY REFERENCE QUS-EC.
IF INFO-COMPLETE OF QGY-OLOBJ-LISTINFO = "C"
OR INFO-COMPLETE OF QGY-OLOBJ-LISTINFO = "P"
  PERFORM DISPLAY-FILES-FOUND
ELSE
  CALL "QGYCLST" USING
    BY REFERENCE REQUEST-HANDLE
    OF QGY-OLOBJ-LISTINFO,
    BY REFERENCE QUS-EC,
  DISPLAY "Error encountered",
  STOP RUN.

```

A Program to use an Integrated File System File COBOL

```

PROCESS NOMONOPRC.
IDENTIFICATION DIVISION.
PROGRAM-ID. IFS.
* CRTBNDCBL BNDDIR(QC2LE)

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
    LINKAGE TYPE PROCEDURE FOR "open"
    USING ALL DESCRIBED.

```

errno is part of C runtime so we need to use BNDDIR(QC2LE) with CRTBNDCBL

open is an ILE API with optional parameters so we use ALL DESCRIBED

A Program to use an Integrated File System File COBOL

```

DATA DIVISION.
WORKING-STORAGE SECTION.
01 File-Desc          PIC S9(009) BINARY.
01 Return-Value       PIC S9(009) BINARY.
01 O-Flag             PIC S9(009) BINARY.
01 File-Mode          PIC S9(009) BINARY.
01 Conversion-ID      PIC S9(009) BINARY VALUE 0.
01 Create-CCSID       PIC S9(009) BINARY VALUE 0.
01 Some-Text          PIC X(050) VALUE "Hello World".
01 Bytes-To-Write     PIC S9(009) BINARY.
01 Returned-Data      PIC X(050).
01 Error-Ptr          POINTER.

77 O-RDONLY           PIC S9(009) BINARY VALUE 1.
77 O-RDWR             PIC S9(009) BINARY VALUE 4.
77 O-CREAT            PIC S9(009) BINARY VALUE 8.
77 O-EXCL             PIC S9(009) BINARY VALUE 16.
77 O-CCSID            PIC S9(009) BINARY VALUE 32.
77 O-TEXTDATA         PIC S9(009) BINARY VALUE 16777216.
77 O-TEXT-CREAT       PIC S9(009) BINARY VALUE 33554432.

77 S-IRGRP            PIC S9(009) BINARY VALUE 32.
77 S-IWUSR            PIC S9(009) BINARY VALUE 128.
77 S-IRUSR            PIC S9(009) BINARY VALUE 256.

```

A Program to use an Integrated File System File COBOL

```

PROCEDURE DIVISION.
MAIN-LINE.
    COMPUTE O-Flag = O-RDWR + O-CREAT + O-EXCL + O-CCSID
                + O-TEXTDATA + O-TEXT-CREAT.
    COMPUTE File-Mode = S-IRUSR + S-IWUSR + S-IRGRP.
    CALL LINKAGE PRC "open" USING
        BY CONTENT z"/COMMON_File.txt",
        BY VALUE O-Flag,
        BY VALUE File-Mode,
        BY VALUE Conversion-ID,
        BY VALUE Create-CCSID,
        RETURNING File-Desc.

    IF File-Desc = -1
        DISPLAY "Failure on open",
        PERFORM Error-Found.
    COMPUTE Bytes-To-Write = FUNCTION LENGTH
                            (FUNCTION TRIMR(Some-Text)).
    CALL LINKAGE PRC "write" USING
        BY VALUE File-Desc,
        BY REFERENCE Some-Text,
        BY VALUE Bytes-To-Write
        RETURNING Return-Value.
    IF Return-Value < Bytes-To-Write
        DISPLAY "Did not write all data".

```

A Program to use an Integrated File System File COBOL

```

CALL LINKAGE PRC "close" USING BY VALUE File-Desc,
                                RETURNING Return-Value.

IF Return-Value = -1
    DISPLAY "Failure on 1st close",
    PERFORM Error-Found.
COMPUTE O-Flag = O-RDONLY + O-TEXTDATA.
CALL LINKAGE PRC "open" USING
    BY CONTENT z"/COMMON_File.txt",
    BY VALUE O-Flag,
    RETURNING File-Desc.

IF File-Desc = -1
    DISPLAY "Failure on 2nd open",
    PERFORM Error-Found.
CALL LINKAGE PRC "read" USING
    BY VALUE File-Desc,
    BY REFERENCE Returned-Data,
    BY VALUE LENGTH OF Returned-Data,
    RETURNING Return-Value.

IF Return-Value < Bytes-To-Write
    DISPLAY "Not all data returned"
ELSE
    DISPLAY Returned-Data.
CALL LINKAGE PRC "close" USING BY VALUE File-Desc,
                                RETURNING Return-Value.
STOP RUN.

```

A Program to use an Integrated File System File COBOL

```
Error-Found.
  CALL LINKAGE PRC "__errno" RETURNING Error-Ptr.
  SET ADDRESS OF Error-Nbr TO Error-Ptr.
  DISPLAY "Failure errno:" Error-Nbr.
  STOP RUN.
```

Determining Size of a CL Variable

• Command GETVARSIZ:

```
CMD      PROMPT('Get Variable Size')
PARM     KWD(VAR) TYPE(*CHAR) LEN(1) RTNVAL(*YES) +
          MIN(1) VARY(*YES *INT2) PROMPT('Variable +
          name')
PARM     KWD(VARSIZ) TYPE(*INT4) RTNVAL(*YES) +
          MIN(1) PROMPT('Variable size')
```



Tool

• CL Program GETVARSIZ – written to work on releases prior to V5R4

```
PGM  PARM(&VAR &VARSIZ)
DCL  VAR(&VAR)  TYPE(*CHAR) LEN(9999)
DCL  VAR(&VARSIZ) TYPE(*INT)

CHGVAR  VAR(&VARSIZ) VALUE(%BIN(&VAR 1 2))
ENDPGM
```

• To create:

- CrtCmd Cmd(GetVarSiz) Pgm(GetVarSiz) Allow(*IPgm *BPgm *IMod *BMod)
- CrtBndCL Pgm(GetVarSiz)

QSPRJOBQ – A Sample Retrieve API

Retrieve Job Queue Information CL Program to Display the Number of Jobs on a JOBQ

```

PGM
DCL      VAR(&RECEIVER)    TYPE(*CHAR)  LEN(144)
DCL      VAR(&NBR_JOBS)    TYPE(*INT)    STG(*DEFINED)  DEFVAR(&RECEIVER 49)
DCL      VAR(&NBR_JOBS_C)  TYPE(*CHAR)  LEN(10)
DCL      VAR(&LENGTHRVCV)  TYPE(*INT)
DCL      VAR(&JOBQ)        TYPE(*CHAR)  LEN(20)
DCL      VAR(&JOBQNAME)    TYPE(*CHAR)  STG(*DEFINED) +
      LEN(10)  DEFVAR(&JOBQ)
DCL      VAR(&JOBQLIB)     TYPE(*CHAR)  STG(*DEFINED) +
      LEN(10)  DEFVAR(&JOBQ 11)
DCL      VAR(&ERRBYTPRV)   TYPE(*INT)  VALUE(0)

CHGVAR   VAR(&JOBQNAME)    VALUE('QBATCH')
CHGVAR   VAR(&JOBQLIB)     VALUE('*LIBL')
GETVARSIZ VAR(&RECEIVER)   VARSIZ(&LENGTHRVCV)
CALL     PGM(QSPRJOBQ)    PARM(&RECEIVER &LENGTHRVCV +
      'JOBQ0100' &JOBQ &ERRBYTPRV)
CHGVAR   VAR(&NBR_JOBS_C)  VALUE(&NBR_JOBS)
SNDPGMMSG MSG(&NBR_JOBS_C) TOPGMQ(*EXT)
ENDPGM

```

List Objects API (QUSLOBJ)

List all files in a library - CL

```

PGM      PARM(&LIBRARY)
DCL      VAR(&LIBRARY)    TYPE(*CHAR)  LEN(10)

/* Generic header information */
DCL      VAR(&USRSPCPTR)   TYPE(*PTR)
DCL      VAR(&LISTHDR)     TYPE(*CHAR)  STG(*BASED)  LEN(192)  BASPTR(&USRSPCPTR)
DCL      VAR(&LISTSTS)     TYPE(*CHAR)  STG(*DEFINED) LEN(1)  DEFVAR(&LISTHDR 104)
DCL      VAR(&LISTSTOFS)   TYPE(*INT)    STG(*DEFINED) DEFVAR(&LISTHDR 125)
DCL      VAR(&LISTENTNBR)  TYPE(*INT)    STG(*DEFINED) DEFVAR(&LISTHDR 133)
DCL      VAR(&LISTENTSIZ)  TYPE(*INT)    STG(*DEFINED) DEFVAR(&LISTHDR 137)

/* List object detail information */
DCL      VAR(&LISTENTPTR)   TYPE(*PTR)
DCL      VAR(&LISTENT)     TYPE(*CHAR)  STG(*BASED) +
      LEN(30)  BASPTR(&LISTENTPTR)
DCL      VAR(&OBJNAME)     TYPE(*CHAR)  STG(*DEFINED) +
      LEN(10)  DEFVAR(&LISTENT 1)

/* Miscellaneous variables */
DCL      VAR(&CURENT)      TYPE(*INT)
DCL      VAR(&TEXT)        TYPE(*CHAR)  LEN(50)
DCL      VAR(&QUALIBNAM)   TYPE(*CHAR)  LEN(20) +
      VALUE('*ALL' ' ')

CALL     PGM(QUSCRTUS)    PARM('OBJLIST QTEMP ' ' +
      ' X'00000001' X'00' '*CHANGE ' &TEXT)

```

List Objects API (QUSLOBJ)

List all files in a library - CL

```

/* Get list of objects */
CHGVAR VAR(%SST(%QUALLIBNAM 11 10)) VALUE(&LIBRARY)
CALL PGM(QUSLOBJ) PARM('OBJLIST QTEMP ' +
'OBJL0100' %QUALLIBNAM '*FILE')

/* Get pointer to user space */
CALL PGM(QUSPTRUS) PARM('OBJLIST QTEMP ' +
&USRSPCPTR)

/* Was the list built successfully? */
IF COND(((&LISTSTS = 'C') *OR (&LISTSTS = 'P')) +
*AND (&LISTENTNBR > 0)) THEN(DO)

/* Address the first list entry */
CHGVAR VAR(&LISTENTPTR) VALUE(&USRSPCPTR)
CHGVAR VAR(%OFFSET(&LISTENTPTR)) +
VALUE(%OFFSET(&LISTENTPTR) + &LISTENTOFS)

/* Process all returned entries */
DOFOR VAR(&CURRENT) FROM(1) TO(&LISTENTNBR)
/* Write the object information */
SNDPGMMSG MSG(&OBJNAME) TOPGMQ(*EXT)
/* Address the next list entry */
CHGVAR VAR(%OFFSET(&LISTENTPTR)) +
VALUE(%OFFSET(&LISTENTPTR) + &LISTENTSIZ)
ENDDO
SNDPGMMSG MSG('End of list') TOPGMQ(*EXT)
ENDDO
ENDPGM

```

A Program to use an Integrated File System File CL

```

PGM
DCL VAR(&FILE_DESC) TYPE(*INT)
DCL VAR(&RC) TYPE(*INT)
DCL VAR(&O_FLAG) TYPE(*INT)
DCL VAR(&FILE_MODE) TYPE(*INT)
DCL VAR(&CONV_ID) TYPE(*INT) VALUE(0)
DCL VAR(&CRT_CCSD) TYPE(*INT) VALUE(0)
DCL VAR(&SOME_TEXT) TYPE(*CHAR) VALUE('Hello +
World')
DCL VAR(&BYTES_OUT) TYPE(*INT)
DCL VAR(&RTN_DATA) TYPE(*CHAR)
DCL VAR(&LEN_RTNDTA) TYPE(*INT)
DCL VAR(&NULL) TYPE(*CHAR) LEN(1) VALUE(X'00')
DCL VAR(&FILE_NAME) TYPE(*CHAR) +
VALUE('/COMMON_File.txt')

```

A Program to use an Integrated File System File CL

```

DCL      VAR(&O_RDONLY)  TYPE(*INT)  VALUE(1)
DCL      VAR(&O_RDWR)    TYPE(*INT)  VALUE(4)
DCL      VAR(&O_CREAT)    TYPE(*INT)  VALUE(8)
DCL      VAR(&O_EXCL)     TYPE(*INT)  VALUE(16)
DCL      VAR(&O_CSID)     TYPE(*INT)  VALUE(32)
DCL      VAR(&O_TEXTDATA) TYPE(*INT)  VALUE(16777216)
DCL      VAR(&O_TEXTCRT)  TYPE(*INT)  VALUE(33554432)

DCL      VAR(&S_IRGRP)    TYPE(*INT)  VALUE(32)
DCL      VAR(&S_IWUSR)    TYPE(*INT)  VALUE(128)
DCL      VAR(&S_IRUSR)    TYPE(*INT)  VALUE(256)

DCL      VAR(&ERROR_PTR)  TYPE(*PTR)
DCL      VAR(&ERROR_NBR)  TYPE(*INT)  STG(*BASED) +
      BASPTR(&ERROR_PTR)
DCL      VAR(&ERROR_CHR)  TYPE(*CHAR)  LEN(10)

```

A Program to use an Integrated File System File CL

```

CHGVAR   VAR(&O_FLAG)  VALUE(&O_RDWR + &O_CREAT + +
&O_EXCL + &O_CSID + &O_TEXTDATA + &O_TEXTCRT)
CHGVAR   VAR(&FILE_MODE) VALUE(&S_IRUSR + &S_IWUSR + &S_IRGRP)
CHGVAR   VAR(&FILE_NAME) VALUE(&FILE_NAME *TCAT &NULL)
CALLPRC  PRC('open')  PARM((&FILE_NAME) (&O_FLAG *BYVAL) +
(&FILE_MODE *BYVAL) (&CONV_ID *BYVAL) +
(&CRT_CSID *BYVAL)) RTNVAL(&FILE_DESC)
IF       COND(&FILE_DESC = -1) THEN(DO)
      SNDPGMMSG  MSG('Failure on open') TOPGMQ(*EXT)
      GOTO  CMDLBL(ERROR)
      ENDDO

CHGVAR   VAR(&BYTES_OUT) VALUE(11)
CALLPRC  PRC('write')  PARM((&FILE_DESC *BYVAL) +
(&SOME_TEXT) (&BYTES_OUT *BYVAL)) RTNVAL(&RC)
IF       COND(&RC < &BYTES_OUT) THEN(SNDPGMMSG +
      MSG('Did not write all data') TOPGMQ(*EXT))

CALLPRC  PRC('close')  PARM((&FILE_DESC *BYVAL)) RTNVAL(&RC)
IF       COND(&RC = -1) THEN(DO)
      SNDPGMMSG  MSG('Failure on 1st close') TOPGMQ(*EXT)
      GOTO  CMDLBL(ERROR)
      ENDDO

```

A Program to use an Integrated File System File CL

```

CHGVAR      VAR(&O_FLAG) VALUE(&O_RDONLY + &O_TEXTDATA)
CALLPRC     PRC('open') PARM((&FILE_NAME) (&O_FLAG *BYVAL) +
                              (&FILE_MODE *BYVAL)) RTNVAL(&FILE_DESC)
IF          COND(&FILE_DESC = -1) THEN(DO)
            SNDPGMMSG MSG('Failure on 2nd open') TOPGMQ(*EXT)
            GOTO      CMDLBL(ERROR)
            ENDDO

GETVARSIZ   VAR(&RTN_DATA) VARSIZ(&LEN_RTNDTA)
CALLPRC     PRC('read') PARM((&FILE_DESC *BYVAL) (&RTN_DATA) +
                              (&LEN_RTNDTA *BYVAL)) RTNVAL(&RC)
IF          COND(&RC < &BYTES_OUT) THEN(SNDPGMMSG +
            MSG('Not all data returned') TOPGMQ(*EXT))
ELSE        CMD(SNDPGMMSG MSG(&RTN_DATA) TOPGMQ(*EXT))
CALLPRC     PRC('close') PARM((&FILE_DESC *BYVAL)) RTNVAL(&RC)
RETURN

ERROR:      CALLPRC     PRC('__errno') RTNVAL(&ERROR_PTR)
            CHGVAR     VAR(&ERROR_CHR) VALUE(&ERROR_NBR)
            SNDPGMMSG  MSG(&ERROR_CHR) TOPGMQ(*EXT)
            RETURN
            ENDPGM

```

A Program to use an Integrated File System File CL

- To create the CL program IFS requires two steps if not on V6R1
 - CRTCLMOD MODULE(IFS)
 - CRTPGM PGM(IFS) BNDDIR(QC2LE)
- With V6R1 the QC2LE *BNDDIR is automatically included
 - CRTBNDCL PGM(IFS)