1 Rational Open-Access: RPG Edition

Rational Open Access: RPG Edition provides a way for RPG programmers to use the simple and well-understood RPG I/O model to access resources and devices that are not directly supported by RPG.

Open Access opens up RPG's file I/O capabilities, allowing anyone to write innovative I/O handlers to access other devices and resources such as:

- Browsers
- Mobile devices
- Cloud computing resources
- Web services
- External databases
- XML files
- Spreadsheets

An Open Access application has three parts:

1. An RPG program that uses normal RPG coding to define an Open Access file and use I/O operations against the file.
2. A handler procedure or program that is called by Open Access to handle the I/O operations for the file.
3. The resource or device that the handler is using or communicating with.

Open Access is the linkage between parts 1 and 2. Licenced program 5733-OAR is required to use Open Access at runtime.

1.1 Open Access handlers

Open Access does not provide the handlers.

Anyone can write the handlers that extend RPG IV's I/O capabilities to new resources and devices.

- Software tool vendors
- Business partners
- Services organizations
- Users

The handler can map to the RPG device-type whose I/O operations best matches the functions provided by the handler. For example, a user-interface application could map to a WORKSTN file, an Excel document could map to a PRINTER file, and a web service could map to a keyed DISK file.

1.2 Two ways to approach Open Access

1. The handler is written after the application is written

   Example: An existing application that uses 5250 display files modified to use Open Access for the WORKSTN files.

   - The RPG program is modified by adding the HANDLER keyword to the WORKSTN files
   - The handler must handle all the operations and requirements of the existing RPG program.
   - This type of handler will normally be provided by an outside expert such as a software tool vendor or business partner.

2. The handler is written before the application is written

   Example: The RPG programmer wants to use a Web service.

   - The handler provider creates a keyed database file matching the web service.
   - The handler provider can tell the RPG programmer what I/O operations that the handler will support. For example, only OPEN, CHAIN, CLOSE.
   - The RPG programmer codes that file as an externally described keyed DISK file.
   - This type of handler may be written by the same RPG programmer who uses the Open Access file, or it may be provided by an outside expert.

1.3 Two ways to approach Open Access

When a RPG program performs an I/O operation for a “normal” file, a system data management function is called to handle the operation.

When the RPG program performs an I/O operation for an Open Access file, the Open Access handler is called.

The handler receives a data structure parameter with subfields that enable the handler to perform the correct I/O operation, and provide information back to RPG.

2 RPG coding to use Open Access

Other than the HANDLER keyword, there is no new syntax related to using an Open Access file.

2.1 HANDLER keyword

The RPG programmer indicates that a file is an Open-Access file by coding the HANDLER keyword on the File specification. All the operations allowed by RPG for the specified device (DISK, PRINTER, WORKSTN) are available for the Open-Access file.

The HANDLER keyword identifies the program or procedure that will handle all operations for the file.

2.1.1 Example:

```
Fmyfile    cf   e             workstn
handler('MYLIB/MYSRVPGM(hdlMyfile)')
F                                     extdesc('MYFILE')
F                                     usropn
```

The HANDLER keyword has two parameters:

1. Identifies the handler

   This parameter can be one of the following:

   - A character literal or character variable identifying a procedure in a service program. The value must be in the form 'LIBRARY/SRVPGM(PROCEDURE)' or 'SRVPGM(PROCEDURE)'. The names are case-sensitive.
   - A character literal or character variable identifying a program. The value must be in the form 'LIBRARY/PGM' or 'PGM'. The names are case-sensitive.
   - A prototype for a bound procedure.
   - A procedure pointer literal (%PADDR) or variable.

   When the handler is identified by a prototype or by %PADDR, it is resolved at bind time of the RPG program. When the handler is identified by a character literal or character variable, it is resolved each time the Open-Access file is opened.

2. (Optional) Identifies a parameter to be passed to the handler from the RPG program

   This parameter can be used to pass additional information to the handler that is not available through an RPG file operation.

   For example, to process an IFS file, the handler needs to know the path to the IFS file. The RPG programmer can provide this through the second HANDLER keyword parameter. The second parameter is the name of a variable, usually a data structure. In the example, the variable is the data structure "ifsDs". When the handler is called, it will receive a pointer to the ifsDs data structure.

   The provider of the handler would tell the RPG programmer whether the additional information was required, and what data structure to use as a template. Usually, the provider of the handler would provide a template data structure in a /COPY file. In the example, the library MYLIB contains both the handler program READIFS and the /COPY File QRPGLESRC with member READIFS. The copy file has the template data structure readIfs_t.

```
FmyIfsFile if   e   disk  extdesc('MYLIB/READIFS')
F                         USROPN
F                         handler(readIfs : ifsDs)
 /copy MYLIB/QRPGLESRC,READIFS
D ifsDs          ds       likeds(readIfs_t)
 /free
    ifsDs.path = '/home/mydir/myIfsFile.txt';
    open myIfsFile;
```

2.2 Comparison to RPG SPECIAL files

An Open-Access file is similar in nature to a SPECIAL file. A SPECIAL file also uses a user-written program to handle the operations for the file, and it allows additional parameters to be passed to the handler from the RPG program.

Some of the differences between Open-Access files and SPECIAL files are

1. A SPECIAL file only allows the operations available for a sequential (SEQ) file. An Open-Access file can be used for any type of RPG device.

2. A SPECIAL file handler only receives a minimal amount of information about the file operation. An Open-Access file handler receives much more information such as the name of the file, record format, the names and types of the fields.
3. A SPECIAL file handler can only pass back a minimal amount of feedback information: the result status of 0, 1 or 2, and a 5-digit SPECIAL error code value. If an error occurs for a SPECIAL file, the RPG status code is always 1231 (Error in SPECIAL file). An Open-Access file handler has the ability to pass back much more information such as the RPG status code, the file feedback areas, the relative record number, function-key pressed, printer overflow.
4. A SPECIAL file handler can only be a program. An Open Access handler can be a program or a procedure.

3 Coding the handler

The handler can be coded in any ILE language. Include files are available in library QOAR for ILE RPG, ILE C and ILE COBOL. The member name in each include file (QRPGLESRC, H, QCBLLESRC) is QRNOPENACC.

Library QOAR is the library for product 5733-OAR. The include files can be used without having a license for the product.

3.1 The record and key data

There are two modes available for the handler to access and provide the input, output and key data. The handler can choose which mode to use when it is processing the OPEN operation by setting the useNamesValues subfield of the handler parameter to '1'.

Name-value information

The input, output, and search arguments are provided as an array of information about each subfield. The information includes the external short name, the type, the length, decimals, date and time format. The values are provided in a human-readable form; numeric values are formatted as they would be by the %CHAR built-in function. Most values are in the job CCSID, but UCS-2 and Graphic values are in the CCSID used in the external file.

For an input capable operation, READx, CHAIN, EXFMT, the handler must set the values of all the input-only fields. They must be provided in the same way that RPG provides them for the output data and the key data, with numeric values in string form, and date, time and timestamp values formatted exactly as specified by the format and separator. For an EXFMT operation, the handler may also change the values of the input and output fields.

When the value for an input-capable field is set by the handler, the handler must ensure that the value length (valueLenBytes

subfield) is set correctly to match the amount of data provided by the handler.

Notes:

1. The name-value information is available only for I/O to externally-described record formats. It is also available for input operations to externally-described files when there is only one non-ignored record format in the file.
2. When name-value information has been requested by the handler, and the information is not available, either because the file is program-described, or because the operation is an input operation to a file with more than one record format, the request for name-value information will be ignored by RPG. No exception will be given; it is up to the handler to detect this situation. The handler may send an exception, or set the return code a failing status, or the handler may adjust to the situation and use the record data information provided by the data structures matching the I/O buffers and key structures. If the file uses INDARA, the indicators appear in the name-value information.

Data structures matching the I/O buffers and key structures

The data structures are passed to the handler exactly as they would be passed to data-management functions. No name and type information is passed to the handler. If the handler needs the names, types and offsets of the fields in the data structures, the handler must call an API.

During an input-capable operation, READx, CHAIN or EXFMT, the handler must set data for all fields in the input buffer, and in the input null-byte map if there are any null-capable fields. If the file uses INDARA, the handler must work with the indara area of the handler parameter to set the values of any input-capable indicators used by the file.

For a keyed I/O operation, the handler must work with the fields in the key buffer, and in the key null-byte map if there are any null-capable keys.

4 The parameter passed to the Open-Access handler

The same parameter is passed to the handler for all operations to the file. The parameter is a data structure with several types of subfields:

- Some subfields **must** be set by the handler before returning to the RPG program. For example:

  rpgStatus

  The status must be set to the value of an RPG status code representing an I/O error if the handler considered that the operation has failed. A status of zero indicates success.

  If the status is set to a value other than an RPG I/O error status code (a valid RPG status between 1000 and 2000), the resulting behavior of the RPG program is undefined.

  recordName

  For an input operation (READx, CHAIN) to an externally-described file with more than one record format, the record name must be set by the handler so that RPG knows which record format to work with.

  inputDataLen

  For an input operation (READx, CHAIN) to an externally-described file, the length of the input record must be set by the handler. **Note:** This is not necessary if name-value information is being used.

  eof

  Must be set to '1' if a READ operation or a subfile-WRITE operation reaches end of file. A value of '0' indicates that the record was successfully read or written.

  found

  Must be set to '1' if a CHAIN, SETLL or SETGT operation was successful. A value of '0' indicates that the record was not found.

  equal

  Must be set to '1' if a SETLL operation found a record that exactly matched the search argument. A value of '0' indicates that an exact match was not found.

  printerOverflow

Must be set to '1' if a WRITE operation to a printer file caused the page to overflow. A value of '0' indicates that overflow did not occur.

input record

For an input operation (READx, CHAIN, EXFMT), the handler must provide the value of the input record. This can be done through the inputBuffer subfield or through the namesValues subfield.

- Some subfields may be set by the handler before returning to the RPG program if the RPG program depends on the value. For example:

  rrn

  The relative record number is available to the RPG programmer through the RECNO keyword.

  functionKey

  For a WORKSTN input-capable operation, a function key value of 1-24 causes an *INKx indicator to be set on. A value of 121-126 indicates one of the PRINT, ROLLUP, ROLLDOWN, CLEAR, HELP or HOME keys, and causes the status to be set to 1121 – 1126. Any other value causes an I/O error with status 1299.

  Use these constants to set the functionKey subfield:

  ```
  QrnFunctionKey_None
  QrnFunctionKey_01
  …
  QrnFunctionKey_24
  QrnFunctionKey_PRINT
  QrnFunctionKey_ROLLUP
  QrnFunctionKey_ROLLDOWN
  QrnFunctionKey_CLEAR
  QrnFunctionKey_HELP
  QrnFunctionKey_HOME
  ```

- Some subfields provide useful information to the handler For example:

recordLevels

For an externally-described file, it provides a list of the record formats used by the program and their level-indicators.

externalFile

Indicates the file that the RPG program is opening at runtime. It is the value specified by the EXTFILE keyword for the file. If the EXTFILE keyword was not specified, the library is *LIBL and the file is RPGNAME, where "RPGNAME" is the name given to the file on the File specification.

Note: If an override is in effect for the file at run-time, this does not reflect the true file.

externalMember

Indicates the member that the RPG program is opening at runtime. It is the value specified by the EXTMBR keyword for the file. If the MBR keyword was not specified, it is *FIRST.

Note: If an override is in effect for the file at run-time, this does not reflect the true member.

compileFile

Indicates the file used at compile-time to define an externally-described file.

Note: If an override was in effect for the file at compile-time, this reflects the true file used at compile-time. For example, if the RPG file specification or EXTDESC keyword specifies file MYFILE, and there is an override for MYFILE to MYLIB/OTHERFILE, then the compileFile subfield will reflect library MYLIB and file OTHERFILE.

externallyDescribed

Indicates the file was externally-described.

keyedFile

Indicates the DISK file is processed by keys rather than relative-record numbers.

blocked

Indicates that the DISK file is processed in blocks by the RPG program. **Note:** The handler can only provide one record at a time to the RPG program.

- Some subfields allow the handler to maintain private information.

  userArea

  If the handler wants to share additional information with the RPG programmer, the second parameter of the HANDLER keyword can be used to specify a variable to be passed to the handler. The userArea subfield is a pointer to the RPG program variable. The variable can have any data type; usually it is a data structure. The RPG program and the handler must agree on the type; usually they would both use the same /COPY file to define this variable.

  stateInfo

  If the handler wants to maintain private state information across calls to the handler, it can set the stateInfo pointer to the address of a data structure. One way to handle the stateInfo pointer is to allocate the storage for the data structure during the OPEN operation, and to deallocate the storage during the CLOSE operation.

4.1 The main structure of the Open-Access parameter

The structure name is QrnOpenAccess_T.

Subfield	Type	Set by	Used by
structLen	UBIN4	RPG	Handler
parameterFormat	CHAR(8)	RPG	Handler
userArea	Pointer [2]	RPG	Handler and RPG programmer
stateInfo	Pointer [4]	Handler	Handler
recordLevels [1]	Pointer [2]	RPG	Handler
inputBuffer	Pointer [2, 3]	Handler	RPG
inputNullMap [1]	Pointer [2, 3]	Handler	RPG
outputBuffer	Pointer [2, 3]	RPG	Handler
outputNullMap [1]	Pointer [2, 3]	RPG	Handler
namesValues [1]	Pointer [2, 3]	RPG and Handler	RPG and Handler
Key	Pointer [2, 3]	RPG	Handler

Subfield	Type	Set by	Used by
keyNullMap [1]	Pointer [2, 3]	RPG	Handler
keyNamesValues [1]	Pointer [2, 3]	RPG	Handler
Indara	Pointer [2, 3]	RPG and Handler	RPG and Handler
Prtctl	Pointer [2, 3]	RPG and Handler	RPG and Handler
openFeedback	Pointer [4]	Handler	RPG
ioFeedback	Pointer [4]	Handler	RPG
deviceFeedback	Pointer [4]	Handler	RPG
externalFile	QrnObject_T	RPG	Handler
externalMember	CHAR(10)	RPG	Handler
compileFile [1]	QrnObject_T	RPG	Handler
recordName [1]	CHAR(10)	RPG and Handler	RPG and Handler
rpgOperation	UINT(4)	RPG	Handler
rpgStatus	INT(4)	Handler	RPG
inputBufferLen	UINT(4)	RPG	Handler
inputNullMapLen [1]	UINT(4)	RPG	Handler
outputBufferLen	UINT(4)	RPG	Handler
outputNullMapLen [1]	UINT(4)	RPG	Handler
keyLen	UINT(4)	RPG	Handler
keyNullMapLen [1]	UINT(4)	RPG	Handler
inputDataLen	UINT(4)	Handler	RPG
openFeedbackLen	UINT(4)	Handler	RPG
ioFeedbackLen	UINT(4)	Handler	RPG
deviceFeedbackLen	UINT(4)	Handler	RPG
numKeys [1]	UINT(4)	RPG	Handler
Rrn	UINT(4)	RPG and Handler	RPG and Handler
formLen	UINT(4)	RPG	Handler
formOfl	UINT(4)	RPG	Handler
functionKey	UINT(1)	Handler	RPG

Subfield	Type	Set by	Used by
externallyDescribed	Indicator	RPG	Handler
keyedFile	Indicator	RPG	Handler
Blocked	Indicator	RPG	Handler
Eof	Indicator	Handler	RPG
Found	Indicator	Handler	RPG
Equal	Indicator	Handler	RPG
printerOverflow	Indicator	Handler	RPG
inputWithLock	Indicator	RPG	Handler
useNamesValues [1]	Indicator	Handler	RPG

4.1.1 Notes on the QrnOpenAccess_T subfields

1. This subfield is meaningful only for externally-described files.
2. The "Set by" and "Used by" columns refer to the data that the pointer is pointing to.
3. The pointer is set by RPG. The handler sets the data pointed to by the pointer, usually with a variable or data structure based on the pointer.
4. The pointer is set by the handler.

4.1.2 Descriptions of the QrnOpenAccess_T subfields

blocked

'1' if the file is blocked in the RPG program. '0' otherwise.

compileFile

The library and file that the RPG compiler used at compile-time for the description of an externally-described file.

deviceFeedback

A pointer to the information that the handler provides to RPG to set the device-specific feedback part of the file's INFDS. If this information is not supplied by the handler, the device-specific-feedback part of the INFDS is not updated by RPG. The length is specified by deviceFeedbackLen.

deviceFeedbackLen

The length of the device-specific-feedback information provided by the handler.

eof

Set to '1' by the handler if the file reached end of file. The handler can leave it at '0' otherwise.

equal

Set to '1' by the handler if an exact match was found by a SETLL operation. The handler can leave it at '0' otherwise.

externalFile

The library and file that the RPG program is opening at runtime.

externalMember

The member that the RPG program is opening at runtime.

externallyDescribed

'1' if the file is externally-described in the RPG program. '0' otherwise.

formLen

Printer form length. Only meaningful for a program described PRINTER file.

formOfl

Printer form overflow. Only meaningful for a program described PRINTER file.

found

Set to '1' by the handler if a record was found by a positioning operation (CHAIN, SETLL, SETGT). The handler can leave it at '0' otherwise.

functionKey

The function key 1-24 pressed by the user for a WORKSTN input operation. Ignored by RPG if the value is other than 1-24.

indara

A pointer to the INDARA array of 99 indicators used for the WORKSTN or PRINTER I/O operation. NULL if the file is not defined to use INDARA.

inputBuffer

A pointer to a data structure in the format of the input buffer for the file or record. The length is given by inputBufferLen.

NULL if the handler is using name-value information or if the input buffer is irrelevant to the operation.

inputBufferLen

The length of the input buffer. Zero if the input buffer is not relevant, or if name-value information is being used.

inputDataLen

When the file is externally-described and the record format was not specified by the RPG program, this subfield must be set by the handler to indicate the length of the data provided in the input buffer.

inputNullMap

A pointer to the null-byte map of the input buffer for the file or record. The length is given by inputNullMapLen. NULL if the handler is using name-value information or if the input buffer is irrelevant to the operation.

inputNullMapLen

The length of the input null-byte map. Zero if the input null-byte map is not relevant, or if name-value information is being used.

inputWithLock

'1' if the input operation is intended to lock the record for later update or delete. '0' otherwise.

ioFeedback

A pointer to the information that the handler provides to RPG to set the I/O feedback part of the file's INFDS. If this information is not supplied by the handler, the I/O-feedback part of the INFDS is not updated by RPG. The length is specified by ioFeedbackLen.

ioFeedbackLen

The length of the I/O-feedback information provided by the handler.

key

A pointer to a data structure in the format of the key structure for the file or record. The length is given by keyLen. NULL if the handler is using name-value information or if the key is irrelevant to the operation.

keyLen

The length of the key buffer. Zero if the key buffer is not relevant, or if name-value information is being used.

keyNamesValues

A pointer to the QrnNamesValues_T structure listing the names, types, and values of each subfield in the key structure. NULL if the handler is not using name-value information, or if the operation does not involve keys.

keyNullMap

A pointer to the null-byte map of the key information for the file or record. The length is given by keyNullMapLen. NULL if the handler is using name-value information or if the key is irrelevant to the operation.

keyNullMapLen

The length of the key null-byte map. Zero if the key null-byte map is not relevant, or if name-value information is being used.

keyedFile

'1' if the file is keyed in the RPG program. '0' otherwise.

namesValues

A pointer to the QrnNamesValues_T structure listing the names, types, and values of each subfield in the record. NULL if the handler is not using name-value information, or if the operation does not involve input or output.

numKeys

The number of keys for a keyed-operation to an externally-described file or format.

openFeedback

A pointer to the information that the handler provides to RPG to set the open-feedback part of the file's INFDS. If this information is not supplied by the handler, the open-feedback part of the INFDS is not updated by RPG. The length is specified by openFeedbackLen.

openFeedbackLen

The length of the open-feedback information provided by the handler.

outputBuffer

A pointer to a data structure in the format of the output buffer for the file or record. The length is given by outputBufferLen. NULL if the handler is using name-value information or if the output buffer is irrelevant to the operation.

outputBufferLen

The length of the output buffer. Zero if the output buffer is not relevant, or if name-value information is being used.

outputNullMap

A pointer to the null-byte map of the output buffer for the file or record. The length is given by outputNullMapLen. NULL if the handler is using name-value information or if the output buffer is irrelevant to the operation.

outputNullMapLen

The length of the output null-byte map. Zero if the output null-byte map is not relevant, or if name-value information is being used.

parameterFormat

The format of the structure

printerOverflow

Set to '1' by the handler if overflow was detected on an output operation to a printer file. The handler can leave it at '0' otherwise.

prtctl

A pointer to the print-control structure for a PRINTER file. NULL if the file is not a PRINTER file.

recordLevels

A pointer to the QrnRecordLevels_T structure listing the record-level indicators for each record in the file. NULL if the file is program described.

recordName

RPG sets this to the record name if the I/O operation specifies the record name or if the file has only one non-ignored record format in the RPG program. Otherwise, RPG sets this to blank. When it is blank, and it is an input operation to an externally-described file, the handler must set this subfield to the record that was used.

rpgOperation

The operation being performed by the RPG program. Constants QrnOperation_* define the possible values. Usually, this maps directly to an RPG operation code, but some operation codes have more than one possible rpgOperation value. For example, the SETLL operation may set QrnOperation_SETLL,

QrnOperation_POSITION_START or QrnOperation_POSITION_END. that was used.

rpgStatus

The handler leaves this subfield set to zero to indicate that the operation was successful. If the handler determines that the operation is not successful, the handler may set this subfield to an RPG status code to have RPG signal the exception associated with that status code.

rrn

Provided by RPG as the relative record number for an output operation or the search argument for a keyed operation. Provided by the handler for an input operation to a database file or a subfile.

stateInfo

Private area defined by the Handler

structLen

The length of the structure

useNamesValues

Set to '1' by the handler if the handler uses the name-value information rather than the data structure information for I/O buffers and keys. The handler can leave it at '0' otherwise. This subfield should only be changed during the OPEN operation.

userArea

Handler-defined area owned by the RPG program

4.2 The names-values structure listing all the fields

The structure name is QrnNamesValues_T.

Subfield	Type	Set by	Used by
num	UBIN4	RPG	Handler
Field	QrnNameValue_T array	RPG and Handler	RPG and Handler

4.2.1 Descriptions of the QrnNamesValues_T subfields

field

The array of name-value information

num

The number of fields described by the information.

4.3 The name-value structure for a single field

The structure name is QrnNameValue_T.

Subfield	Type	Set by	Used by
externalName	CHAR(10)	RPG	Handler
dataType	UINT(1)	RPG	Handler
numericDefinedLen	UINT(1)	RPG	Handler
decimals	UINT(1)	RPG	Handler
dtzFormat	UINT(1)	RPG	Handler
dtSeparator	CHAR(1)	RPG	Handler
input	Indicator	RPG	Handler
output	Indicator	RPG	Handler
isNullCapable	Indicator	RPG	Handler
hasNullValue	Indicator	RPG and Handler	RPG and Handler
valueLenBytes	UINT(4)	RPG and Handler	RPG and Handler
valueMaxLenBytes	UINT(4)	RPG	Handler
valueCcsid	INT(4)	RPG	Handler
value	Pointer [1, 2]	RPG	Handler

Notes on the QrnNameValue_T subfields

1. The "Set by" and "Used by" columns refer to the data that the pointer is pointing to.
2. The pointer is set by RPG. The handler sets the data pointed to by the pointer, usually with a variable or data structure based on the pointer.

4.3.1 Descriptions of the QrnNameValue_T subfields

dataType

The data type of the field. Constants QrnDatatype_* define the values.

decimals

The number of decimal places for a decimal field. Meaningful for numeric fields only.

dtSeparator

The separator for a date or time field. Meaningful for date or time fields only.

dtzFormat

The format of a date, time, or timestamp field. Meaningful for date, time, or timestamp fields only. Constants QrnDtzFormat_* define the values.

externalName

The name of the field in the externally-described file.

hasNullValue

'1' indicates that the field has a null value, '0' indicates that the field does not have a null value. Set by RPG to indicator whether an output field or key field has a null value. Set by the handler to indicator whether an inputfield has a null value. Meaningful only when isNullCapable has a value of '1'.

input

The field is input capable. Mainly useful for the EXFMT operaton.

isNullCapable

'1' if the field is null-capable. '0' otherwise.

numericDefinedLen

The defined length of a numeric field. For a decimal type, packed, zoned, or binary, it is the total number of digits. For a float or integer type, it is the number of bytes. Meaningful for numeric fields only.

value

The value of the field in human-readable form. For numeric values, this is the same value that is provided by the %CHAR built-in function. For date, time, and timestamp values, this is the same value that is provided by the %DATE built-in function in the same format used by the field in the file. Unicode (UCS-2) and Graphic data is in the CCSID of the field in the file. The data

for all other types is in the job CCSID. When the handler sets the data, it must ensure that the length of the data does not exceed the number of bytes specified by valueMaxLenBytes.

valueCcsid

The CCSID of the data in the value pointer. A value of zero indicates that the data is in the job CCSID.

valueLenBytes

Indicates the number of bytes in the data pointed to by the value pointer. Set by RPG to for an output field or a key field. Set by the handler for an input field. It must not be greater than valueMaxLenBytes.

valueMaxLenBytes

Indicates the number of bytes in the buffer pointed to by the value pointer.

4.3.2 Operations

QrnOperation_OPEN

RPG operation: Implicit or explicit OPEN operation.

QrnOperation_POSITION_START

RPG operation: SETLL *START

Handler action: Set the file cursor to the beginning of the file.

QrnOperation_POSITION_END

RPG operation: SETLL *END

Handler action: Set the file cursor to the end of the file.

QrnOperation_READ

RPG operation:

- Implicit or explicit sequential read operation for a database file or record
- Read operation for a user-interface file or record

Handler actions:

- Move the file cursor forward, and either return data or set the "eof" subfield to '1'.

QrnOperation_READC

RPG operation: Read next changed subfile record

Handler action: Move the subfile cursor forward to the next changed record, and either return data or set the "eof" subfield to '1'.

Note: The data for a subfile includes both the input and output capable fields.

QrnOperation_READE

RPG operation: Read equal key, with a search argument specified

Handler action: Move the file cursor forward to the next record, and return data if the record matches the key. Otherwise, set the "eof" subfield to '1'.

Note: See QrnOperation_READE_CURRENT for Read equal key with no search argument specified.

QrnOperation_READP

RPG operation: Sequential read previous operation

Handler action: Move the file cursor backward, and either return data or set the "eof" subfield to '1'.

QrnOperation_READPE

RPG operation: Read equal key previous, with a search argument specified

Handler action: Move the file cursor back to the previous record, and return data if the record matches the key. Otherwise, set the "eof" subfield to '1'.

Note: See QrnOperation_READPE_CURRENT for Read equal key previous with no search argument specified.

QrnOperation_CHAIN

RPG operation:

- If the "keyedFile" subfield is '0', the operation is random retrieval from a file by relative-record number
- If the "keyedFile" subfield is '1', the operation is random retrieval from a file by key

Handler action: If the record specified by the relative record number or key is available, return the data and set the "found" subfield to '1'. Otherwise, set the "found" subfield to '0'.

Note: The data for a subfile includes both the input and output capable fields.

QrnOperation_EXFMT

RPG operation: Write and read a format from a user-interface file

Handler actions:

- Present the output data to the user including any subfile records associated with the record format
- Receive the input data from the user, including input data for any subfile records associated with the record format
- Set the input data in the input buffer or input name-value elements

- If there are any subfile records associated with the record format
  - Save whether each record was changed by the user
  - Save any subfile data for further QrnOperation_READC or QrnOperation_CHAIN operations for the subfile

QrnOperation_SETGT

RPG operation: Set greater than

Handler actions:

- Set the file cursor to the first record greater than the search argument
- Set the "found" subfield to "1" if there is such a record

QrnOperation_SETLL

RPG operation: Set lower limit

Handler actions:

- Set the file cursor to the first record less than or equal to the search argument
- Set the "found" subfield to "1" if there is such a record
- Set the "equal" subfield to "1" if the record is an exact match for the search argument

QrnOperation_UNLOCK

RPG operation: Unlock record

QrnOperation_UPDATE

RPG operation: Update the current record

Handler action: Update the record if it is locked, or set the status to a failing value if it is not locked.

QrnOperation_WRITE

RPG operation: Write record

Handler action: For a write to a subfile record, save the output data so it can be return as part of the data on a subsequent QrnOperation_READC or QrnOperationCHAIN operation.

QrnOperation_DELETE

RPG operation: Delete record by relative record number or key

Note: See QrnOperation_DELETE_CURRENT for Delete the current record with no search argument specified.

QrnOperation_FEOD

RPG operation: Force end of data

QrnOperation_CLOSE

RPG operation: Implicit or explict close of the file

QrnOperation_DELETE_CURRENT

RPG operation: Delete the current record

Handler action: Delete the current record if it is locked, or set the status to a failing value if it is not locked.

Note: See QrnOperation_DELETE for Delete the current record with a search argument specified.

QrnOperation_READE_CURRENT

RPG operation: Read equal key, with no search argument specified

Handler action: Move the file cursor forward to the next record, and return data if the record matches the key of the previously current record. Otherwise, set the "eof" subfield to '1'.

Note: See QrnOperation_READE for Read equal key with a search argument specified.

QrnOperation_READPE_CURRENT

RPG operation: Read equal key previous, with no search argument specified

Handler action: Move the file cursor back to the previous record, and return data if the record matches the key of the previously current record. Otherwise, set the "eof" subfield to '1'.

Note: See QrnOperation_READPE for Read equal key previous with a search argument specified.

5 Summary

- Using Open Access is both extremely simple (HANDLER keyword) and arbitrarily complex (the handler).

- The intention for Open Access is for RPG programmers to be able to use their existing expertise in using the RPG file I/O model, while using others' expertise in accessing new resources and devices.

- In most cases, the handler will be provided by an outside provider.