

## APIs by Example: Parsing Externally Defined Record Buffers

[System iNetwork](#)

System iNEWS Staff

Thu, 10/02/2003 (All day)

This installment of Carsten Flensburg's APIs by Example shows how to use a host of APIs, MI built-in functions, and C library functions to parse an externally defined record buffer. The main RPG ILE service program, CBX109S, includes the following procedures:

### GetFldVal (Get field value)

Based on a file name, field name, and key value, the corresponding field value is returned in left adjusted character format.

### LstFld (List fields)

Lists to the specified user space a list of the specified file's fields, including name, data type, and length.

### Chain (Read record by key)

Performs the actual keyed access to the file identified by the file pointer passed and returns the record buffer retrieved if a match is found.

### RtvFld (Retrieve field)

The buffer offset, data type and field length of the field name specified is retrieved from the user space field list.

### EditC (Edit by edit code)

Converts the specified buffer location containing a numeric value in internal format to a readable character format as defined by the specified edit code.

### ApyDecFmt (Apply decimal format)

Applies the current job's decimal format to binary fields having decimal positions.

These procedures give you the ability to parse and read buffer data. Using these routines as a guide, you can also reverse the process and write to an externally described record buffer. The \_CVTEFN and \_LBPCYNV(R) MI built-in functions enable you to update numeric fields in a record buffer. Those write routines are, as they say, left as an "exercise for the reader."

Using both the read routines as well as coding your own write routines, you can soft-code file and data exchange programs. Note, though, that updating production at the buffer level requires special precaution and careful design, so test your routines thoroughly before using them in production programs.

To show how to call the routines in CBX109S, test program CBX109T is included in this month's tip. CBX109T retrieves field values from the TCP/IP host table, which is stored in physical file QATOCHOST in library QUSRSYS. You can easily change the file name, field names, and key values specified in CBX109T to read another physical file and fields of your choice.

Below is a list of all the APIs and functions used in CBX109S.

### Object - User space APIs:

#### Create User Space (QUSCRTUS)

This API creates a user space in either the user domain or the system domain.

<http://publib.boulder.ibm.com/iseres/v5r2/ic2924/info/apis/quscrtus.htm>

#### Retrieve Pointer to User Space (QUSPTRUS)

QUSPTRUS retrieves a pointer to the contents of a user-domain user space.

<http://publib.boulder.ibm.com/iseres/v5r2/ic2924/info/apis/qusptrus.htm>

### Delete User Space (QUSDLTUS)

This API deletes user spaces created with the Create User Space (QUSCRTUS) API.

<http://publib.boulder.ibm.com/iserics/v5r2/ic2924/info/apis/qusdltus.htm>

### Database and file APIs:

#### List Fields (QUSLFLD)

This API generates a list of fields within a specified file record format name and places the list in a specified user space. You can use the QUSLFLD API only with database file types, such as \*PF, \*LF, and \*DDMF, and device file types, such as \*ICFF and \*PRTF.

<http://publib.boulder.ibm.com/iserics/v5r2/ic2924/info/apis/quslfld.htm>

### Work management API:

#### QUSRJOBI

This API retrieves specific information about a job. The information is grouped in the various formats available.

<http://publib.boulder.ibm.com/iserics/v5r2/ic2924/info/apis/qusrjobi.htm>

### Edit function API:

#### QECCVTEC

This API translates an edit code specification into an edit mask, which is a byte string used to format a numeric value into a readable character string.

<http://publib.boulder.ibm.com/iserics/v5r2/ic2924/info/apis/QECCVTEC.htm>

### MI Built-ini functions:

#### \_LBEDIT

Transforms a numeric value from its internal format to character form, using the provided edit mask. Late bound here refers to the source value location not having to be provided until runtime.

#### \_MEMMOVE

Copies a string from one pointer specified location to another.

### C library functions:

#### Open record file (\_Ropen)

This function opens the record file specified as defined by the key words in the mode parameter. If the file does not exist it will not be created.

#### Close record file (\_Rclose)

This function closes the previously opened record file identified by the file pointer parameter.

#### Read by key (\_Rreadk)

This function reads a record in a keyed file matching the key value parameter. This key value can be partial. The record is locked unless the No\_Lock option is set.

The C functions are documented at

<http://publib.boulder.ibm.com/iserics/v5r2/ic2924/books/c4156071.pdf>.

You can obtain the RPG IV sample program CBX109TS and CBX109T at

<http://www2.systeminetwork.com/noderesources/code/clubtechcode/ParseReadBuffer.zip>.

The above source code was written by Carsten Flensburg. For questions regarding this tip, contact Carsten at <mailto:flensburg@novasol.dk>.

**Source URL:** <http://iprodeveloper.com/rpg-programming/apis-example-parsing-externally-defined-record-buffers>