

[print](#) | [close](#)

## APIs by Example: Update Externally Defined Record Buffer

[System iNetwork Programming Tips Newsletter](#)

[Carsten Flensburg](#)

Carsten Flensburg

Thu, 06/30/2005 (All day)

This week's APIs By Example revisits the challenge of handling an externally defined record buffer. In a previous installment (see the link below), I demonstrated how to parse data from an externally defined record buffer. This time I will show how you can update an externally defined record buffer using various APIs, MI built-ins and C library functions.

Using this technique, many types of file and data exchanges could be soft coded throughout the whole exchange process. However, note that updating production data at the buffer level requires careful design and a high level of precaution -- this example primarily intends to offer a starting point for such an endeavour.

To implement the buffer update technique, the following steps are involved in the process:

- a) The list fields (QUSLFLD) API returns the buffer location, length, and data type of the fields of the specified file.
- b) The Open a Record File `_Ropen` C library function opens the file for update.
- c) The Read a Record by RRN `_Rreadd` reads and locks the record identified by the specified relative record number.
- d) Numeric values are converted to the correct format and buffer location by the Convert External Format to Numeric Value `_CVTEFN` MI built-in.
- e) Character as well as date and time values are simply moved to the correct buffer location by the Memory Move `_MEMMOVE` MI built-in.
- f) Once the buffer update is complete the Update a Record `_Rupdate` C library function writes the buffer back to the file record.
- g) Finally the file is closed using the Close a File `_Rclose` C library function.

All the above functions are located in a number of subprocedures in the service program CBX138S and controlled by the `PutFldVal()` subprocedure in that service program. The field value update can be performed as either a single pass process or as a session process, separating the three steps:

- |   |
|---|
| <div data-bbox="193 1827 1110 2080" data-label="List-Group"><ol style="list-style-type: none"><li>1. Initial step:<ul style="list-style-type: none"><li>- Create user space</li><li>- List file fields to user space</li><li>- Open data base file</li><li>- Read file record by rrn</li></ul></li><li>2. Intermediate step:<ul style="list-style-type: none"><li>- Locate field buffer position</li><li>- Define field data type</li></ul></li></ol></div> |
|---|

3. Final step:	<ul style="list-style-type: none"> <li>- Define field attributes</li> <li>- Put field value to record buffer</li> <li>- Update file record</li> <li>- Close file</li> <li>- Delete user space</li> </ul>
----------------	--

The session process offers some performance advantage over the more simple single pass process. A further performance optimization could be achieved by promoting the read file record and the update file record to their own individual steps. This way file open and close would only occur once for each file being updated and not for each of the file's records, assuming that more than one record is to be updated, of course.

Please note that if the session approach is chosen, the record update does not occur until the final 'Terminate request' process is run. Any preceding field value updates performed during the session will be lost in that event.

To test the PutFldVal() procedure, create the CBX138F file from its source member and add two blank records using the Initialize Physical File Member (INZPFM) command:

```
INZPFM( CBX138F ) TOTRCDS( 2 )
```

The CBX138T test program puts field values to the CBX138F file. For each stage of the test program execution, a RUNQRY command is run against the CBX138F file to let you verify the result of the performed PutFldVal function. You could also simply start a debug session against this program and step through the code lines to watch the process as it unfolds.

When the debug session is positioned on a PutFldVal() procedure Statement, you can use F22 to step into the subprocedure and watch the statements being executed there. The F10 step instruction applies while in the subprocedure.

Please note that you must ensure that the field values for date and time data type fields are formatted correctly with respect to both format and separator, otherwise a data mapping error will occur when the record buffer is written to the file, and the field update(s) will be lost.

Also note that if your geographic location uses different symbols to edit numeric values than the ones used in this example, you can define the appropriate currency, comma, and decimal point symbols in the 'Mask' data structure in the CBX138S service program.

The following sources are included in this API by Example:

CBX138F -- Test data file

CBX138S -- PutFldVal() service program

CBX138T -- PutFldVal() test program

Compilation instructions are found in the source headers.

APIs by Example -- Parsing externally defined record buffer:

<http://www2.systeminetwork.com/article.cfm?ID=17381>

This article demonstrates the following APIs:

Create User Space (QUSCRTUS) API:

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

Retrieve Pointer to User Space (QUSPTRUS) API:

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

Delete User Space (QUSDLTUS) API:

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

List Fields (QUSLFLD) API:

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

Convert External Format to Numeric Value \_CVTEFN MI built-in:

<http://publib.boulder.ibm.com/iserics/v5r2/ic2924/info/mi/CVTEFN.htm>

The \_CVTEFN receiver\_attributes parameter is documented here:

<http://publib.boulder.ibm.com/iserics/v5r2/ic2924/info/mi/CVTCN.htm>

Memory Move \_MEMMOVE MI built-in:

<http://publib.boulder.ibm.com/iserics/v5r2/ic2924/info/mi/MEMMOVE.htm>

Open a Record File for I/O Operations \_Ropen C library function:

<http://publib.boulder.ibm.com/iserics/v5r2/ic2924/books/c415607107.htm#HDRROPEN>

Close a File \_Rclose C library function:

<http://publib.boulder.ibm.com/iserics/v5r2/ic2924/books/c415607107.htm#HDRRCLOSE>

Read a Record by Relative Record Number \_Rread C library function:

<http://publib.boulder.ibm.com/iserics/v5r2/ic2924/books/c415607107.htm#HDRRREADRN>

Update a Record \_Rupdate C library function:

<http://publib.boulder.ibm.com/iserics/v5r2/ic2924/books/c415607107.htm#HDRRUPDATE>

You can retrieve the source code for this API example from

[http://www.pentontech.com/IBMContent/Documents/article/51146\\_26\\_UpdExtRcd.zip](http://www.pentontech.com/IBMContent/Documents/article/51146_26_UpdExtRcd.zip).

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