

APIs by Example: Extract Database Journal Entry Images

[Carsten Flensburg](#)

Mon, 09/09/2013 - 11:55am

Add extracted data to an output file



Combining good ideas and smart techniques adds a certain dimension to my efforts when I'm searching for topics to cover in APIs by Example. In "[Thinking Outside the SQL Box, Part 2](#)" (February 2012), John Vriezen describes a clever way to use SQL to extract file data from journal entry images. Although I read his article, I never got around to putting the technique to use. Then, a while later, *iPro Developer* reader Arthur Adams presented me with an idea to implement a CL command that can extract and recover database journal entry images and add them to an output file.

So in this article, I'll provide the Extract Journal Data (EXTJRNDTA) command, as well as a couple of CL commands that help EXTJRNDTA perform a database journal extract. For full instructions on how to create the EXTJRNDTA command and its prerequisite objects, see the "How to Compile" sidebar below.

Mapping Journal Entries to Their Original Format

All due credit goes to John Vriezen and Arthur Adams for conceiving the method and the idea, respectively. John's ingenious SQL method of mapping database journal entries back into their original file record format is based on two SQL statements: CREATE TABLE and ALTER TABLE.

The CREATE TABLE statement lets you create a table that's a product of a SELECT statement pointing to one or more already existing tables. The resulting table's record format is a concatenation of the record formats of the files you specify for the SELECT statement. Now, if you run Display Journal (DSPJRN) and indicate one database file as the command's primary selection parameter and output the journal entries to an output file, the output file will consist of a journal entry header section containing several columns that define the journal entry's metadata, followed by a single column named *Journal Entry Specific Data* (JOESD) that holds an image of the database file's record.

Depending on the database operation being performed, the latter column amounts to the record data of a new record, the record data before or after an update of the record, or the record data prior to a delete operation. In some cases, you'll want to map the journal entry data image back to the original record format in order to recover the field-formatted data. For example, some files are locked 24/7, so running the Remove Journalized Changes (RMVJRNCHG) command isn't an option; or the recovery of a failed batch update is so complex that you have to perform it programmatically to ensure database integrity; or you need to investigate a file update and therefore must inspect the before and after values of updated database records. These are some of the scenarios I've encountered when trying to map the journal entry data image back to the original record format, and the EXTJRNDTA command has proven useful in these situations.

The RTVRCDLEN and RTVTMPNAM Commands

The EXTJRNDTA command employs two other new CL commands: Retrieve File Record Length (RTVRCDLEN) and Retrieve Temporary Name (RTVTMPNAM). RTVRCDLEN retrieves the total record length for database files and is based on the Retrieve Database File Description (QDBRTVFD) API, which can retrieve all kinds of information about a database file. I've covered the [QDBRTVFD API](#) in several [previous APIs by Example articles](#) (to access those articles, see the links in the "Find Out More" section below).

The RTVTMPNAM command composes and returns a unique temporary object name and is based on the tmpnam() C library function. The tmpnam() function is implemented in two versions on IBM i. The native library version, tmpnam, generates a unique object name in the QTEMP library, and the IFS version, _C_IFS_tmpnam, produces a unique file name in the /tmp directory of the root file system. In this case, I'm using the native library version to create a unique object name in the QTEMP library. This object name is then

assigned to the DSPJRN command’s output file, which will temporarily hold the journal entries in scope for extraction. Let’s look at the whole process in more detail.

How EXTJRNDTA Works

The EXTJRNDTA command processing program performs the following steps to retrieve a selection of journal entries and add them to the specified output file:

- 1. Retrieves the record length for the specified database file for which journal entries are to be listed by the DSPJRN command.
- 2. Retrieves the temporary object name to use for the DSPJRN output file name.
- 3. Formats the EXTJRNDTA command parameters used as input to the DSPJRN command.
- 4. Runs the DSPJRN command and adds the selected journal entries to the output file.
- 5. Executes the RUNSQL command to CREATE TABLE based on the DSPJRN output file format appended to the record format of the original database file.
- 6. Runs the ADDENVVAR SQL_VFY_ALTER_IGNORE command to suppress SQL inquiry message CPA32B2, as explained in “[APIs by Example: Exit Points, APIs, and Environment Variables](#)” (July 2013).
- 7. Executes the RUNSQL command ALTER TABLE DROP COLUMN JOESD to remove the JOESD column and thereby overlay this column with the original file format’s fields.
- 8. Runs the RMVENVVAR SQL_VFY_ALTER_IGNORE command to re-enable SQL inquiry message CPA32B2.
- 9. Executes the CPYF command to copy the DSPJRN output file created in step 4 to the empty file created in step 5.
- 10. Deletes the temporary file created in step 4.

Now the EXTJRNDTA output file contains all selected journal entries, with each entry consisting of the journal entry header information followed by the original file’s fields. Figure 1 shows the fully prompted EXTJRNDTA command.

Figure 1: Extract Journal Data (EXTJRNDTA) command prompt

```
Extract Journal Data (EXTJRNDTA)

Type choices, press Enter.

Journaled physical file:
  File . . . . . Name
  Library . . . . . *LIBL Name, *LIBL, *CURLIB
  Member . . . . . *FIRST Name, *FIRST, *ALL, *NONE
Output file . . . . . Name
  Library . . . . . *CURLIB Name, *CURLIB
Range of journal receivers:
  Starting journal receiver . . *CURRENT Name, *CURRENT, *CURCHAIN
  Library . . . . . Name, *LIBL, *CURLIB
  Ending journal receiver . . . Name, *CURRENT
  Library . . . . . Name, *LIBL, *CURLIB
Starting date and time:
  Starting date . . . . . *BEGIN Date, *CURRENT, *BEGIN
  Starting time . . . . . *AVAIL Time, *AVAIL
Ending date and time:
  Ending date . . . . . *END Date, *CURRENT, *END
  Ending time . . . . . *AVAIL Time, *AVAIL
Number of journal entries . . . *ALL Number, *ALL
Job name . . . . . *ALL Name, *ALL
  User . . . . . Name
  Number . . . . . 000000-999999
Program . . . . . *ALL Name, *ALL
User profile . . . . . *ALL Name, *ALL
```

Figures 2 and 3 display the command prompts of two other commands— RTVTMPNAM and RTVRCDLEN, respectively —included with the code associated with this article.

Figure 2: Retrieve Temporary Name (RTVTMPNAM) command prompt

```
Retrieve Temporary Name (RTVTMPNAM)
```

```
Type choices, press Enter.

CL var for TEMPNAME      (10) . .      Character value
```

Figure 3: Retrieve File Record Length (RTVRCDLEN) command prompt

```
Retrieve File Record Length (RTVRCDLEN)

Type choices, press Enter.

File . . . . . Name
Library . . . . . *LIBL Name, *LIBL, *CURLIB
Record format . . . . . *FIRST Name, *FIRST
CL var for RCDLEN      (5 0) . .      Number
```

Note that to suppress SQL inquiry message CPA32B2, you must download and install the code associated with my article [“APIs by Example: Exit Points, APIs, and Environment Variables.”](#)

The header part of the EXTJRNDTA output file contains the fields in Figure 4.

Figure 4: Fields in the EXTJRNDTA output file header

Field	Data type	Length	Dig	Dec	Text
JOENTL	Zoned	5	5	0	Length of entry
JOSEQN	Zoned	10	10	0	Sequence number
JOCODE	Char	1			Journal Code
JOENTT	Char	2			Entry Type
JODATE	Char	6			Date of entry: Job date
JOTIME	Zoned	6	6	0	Time of entry: hhmmss
JOJOB	Char	10			Name of Job
JOUSER	Char	10			Name of User
JONBR	Zoned	6	6	0	Number of Job
JOPGM	Char	10			Name of Program
JOOBJ	Char	10			Name of Object
JOLIB	Char	10			Objects Library
JOMBR	Char	10			Name of Member
JOCTRR	Zoned	10	10	0	Count or relative record nbr.
JOFLAG	Char	1			Flag: 1 or 0
JOCCID	Zoned	10	10	0	Commit cycle identifier
JOINCDAT	Char	1			Incomplete Data: 1 or 0
JOMINESD	Char	1			Minimized ESD: 0, 1 or 2
JORES	Char	6			Not used

The fields that define the journaled file’s record format and that contain the extracted file data from the converted journal entries immediately follow the JORES field. To copy the extracted file data back into the original file (or a copy of this file), you can use the following CPYF command:

```
CPYF FROMFILE(JRNOUT)
      TOFILE(ORGFIL)
      MBROPT(*ADD)
      FROMRCD(4711)
      NBRRCD(1)
      FMTOPT(*MAP *DROP)
```

This command will add relative record number 4711 from the produced output file JRNOUT to the original file ORGFIL.

Again, thanks to John and Arthur for inspiring the EXTJRNDTA command.

Find Out More

[“APIs by Example: Analyzing Logical Files Using the ODBRTVFD File API”](#)

[“APIs by Example: Physical File Triggers and the ODBRTVFD API”](#)

[“APIs by Example: Retrieve Journal Entries \(QjoRetrieveJournalEntries\)”](#)

[“Retrieve Journal Entry Exit Program”](#)

[“Thinking Outside the SQL Box, Part 1”](#)

[“Thinking Outside the SQL Box, Part 2”](#)

[“Process Mining: A Living View of Systems”](#)

IBM i 7.1 Information Center documentation

[ALTER TABLE \(SQL\)](#)

[CREATE TABLE \(SQL\)](#)

[Display Journal \(DSPJRN\)](#)

[“IBM i Journal Management 7.1”](#)

[Retrieve Database File Description \(QDBRTVFD\) API](#)

[tmpnam\(\)—Produce Temporary File Name](#)

Articles at iProDeveloper.com:

[“Analyze Your Audit Journal with RPG”](#)

[“APIs by Example: Analyzing Logical Files Using the QDBRTVFD File API”](#)

[“APIs by Example: Displaying and Locating a Physical File's Access Paths”](#)

[“APIs by Example: Journal APIs Solving Spooled Files Mysteries”](#)

[“APIs by Example: Physical File Triggers and the QDBRTVFD API”](#)

[“APIs by Example: Print File Field Description”](#)

[“APIs by Example: Retrieve Journal Entries \(QjoRetrieveJournalEntries\)”](#)

[“APIs by Example: Working with Database Files, Fields and More”](#)

[“IFS Journal Monitor”](#)

[“IFS Journal Monitor—Part 1”](#)

[“Job Accounting and the Retrieve Journal Entries API”](#)

[“New Command to Print Journaled and Non-Journalled Objects”](#)

[“Retrieve Journal Entry Exit Program”](#)

[“Temporary Files in the IFS”](#)

[“Thinking Outside the SQL Box, Part 2”](#)

How to Compile

Below, you'll find instructions for creating the Extract Journal Data (EXTJRNDTA) command and all its prerequisite objects. The following sources are included with the code download associated with this article:

CBX2621—RPGLE: Retrieve File Record Length—CPP

CBX2621H—PNLGRP: Retrieve File Record Length—Help

CBX2621X—CMD: Retrieve File Record Length

CBX2622—RPGLE: Retrieve Temporary Name—CPP

CBX2622H—PNLGRP: Retrieve Temporary Name—Help

CBX2622X—CMD: Retrieve Temporary Name

CBX263—CLP: Extract Journal Data—CPP

CBX263H—PNLGRP: Extract Journal Data—Help

CBX263V—CLP: Extract Journal Data—VCP

CBX263X—CMD: Extract Journal Data

CBX263M—CLP: Extract Journal Data—Build Command

To create the above exit programs, compile and run the CBX263M CL program, following the instructions in the source header. You'll also find compilation instructions in the respective source headers of the individual sources.

Source URL: <http://iprodeveloper.com/application-development/apis-example-extract-database-journal-entry-images>