

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

APIs by Example: Message Handling APIs & Additional Message Info Support

[*System iNetwork Programming Tips Newsletter*](#)

[Carsten Flensburg](#)

Carsten Flensburg

Thu, 10/22/2009 (All day)

For native IBM i commands such as Work with Writers (WRKWTR), Work with User Jobs (WRKUSRJOB), Work with Submitted Jobs (WRKSBMJOB), and Work with Active Jobs (WRKACTJOB), it's possible to reply to pending inquiry messages for the writer or job in question by means of these commands' list panels' option 7=Display message. The internal IBM program providing this functionality is called QMHSCVL and is also employed by other message-related commands, including Display Messages (DSPMSG), Work with Messages (WRKMSG), and Display Job Log (DSPJOBLOG). If you run any of these commands and place the cursor on a message and press F1, you invoke the QMHSCVL program and see the Additional Message Information panel for that message.

A couple of the general print APIs include information for writers in a message wait state about the message queue and message key identifying the actual message waiting for a reply, and with release 6.1 the same information has been made available for jobs in general with some of the job-related work management APIs. I've including links to the specific APIs at the end of this article. You can take advantage of these APIs and the information made accessible by them to create your own writer or job commands and utilities and include message-handling facilities similar to those presented by IBM's commands. There's, however, currently no API available providing a public and supported interface to the QMHSCVL functionality. So that's what today's APIs by Example seeks to redress.

Bruce Vining, former system API lead at IBM, in a discussion a while ago on midrange.com revealed that IBM actually for some time has been working on providing such an API but that until now other activities of higher priority have been in the way of a successful completion of that endeavor. So when I recently encountered a situation calling for a message reply function very much like the one offered by the QMHSCVL program, I decided to build my own Additional Message Information (AMI) function. And because from time to time I've noted some interest for this type of message support in the System iNetwork forums and elsewhere, I present the outcome here.

I should point out that IBM's QMHSCVL function supports both program and nonprogram messages, the former holding messages being sent to a job's program or external message queue and the latter holding messages being sent to a message queue object of type *MSGQ. This means that whether you display messages in a message queue or messages from your job's program message queue or job log, you employ that same function. The intention of the Additional Message Information function presented here, however, is to support messages sent to message queues only.

Messages being sent to a program message queue can be processed and replied to only from within the job running the programs issuing the messages, and most often the request for a public and supported interface to the QMHSCVL function has to do with the requirement of being able to reply to messages being sent to a message queue object, typically for writers or jobs hanging in a message

wait condition, waiting for somebody to reply to the message in either the writer's or the operator's message queue. Adding program message support to the version presented here would be quite simple, however, given the requirement. Anyway, let's look at the steps and information involved in creating the Additional Message Information function.

A qualified message queue name and the message key identifying the message to display from the basic information are needed. Armed with this information you can:

1. use the Receive Nonprogram Message (QMHRRCVM) API to retrieve the message text, message ID, message type, message data, and other relevant message information.
2. use the Retrieve Message (QMHRRTVM) API to retrieve and initially format the message's second-level help text, replacing substitution variables.
3. format the message text and second-level help text for the specific output line width and take embedded format instructions into proper consideration.
4. display the message information. For inquiry messages, a reply input option is offered.
5. if a reply is entered, use the Send Reply Message (QMHSNDRM) API to send the message reply and redisplay the message.

As for many of the commands and utilities I've presented earlier in this column, much of the user dialog is taken care of by the User Interface Manager (UIM) panel group constituting the AMI user interface. This significantly reduces the coding efforts involved and also ensures consistency in both appearance and functionality. Below I've included an example of how the Additional Message Information panel looks when displayed for the CPA7025 inquiry message issued when an attempt is made to delete a journal receiver that has not been fully saved:

Additional Message Information			
Message ID	CPA7025	Severity	99
Message type	Inquiry		
Date sent	17-10-09	Time sent	
	02:00:25		
Message : Receiver AUDRCV2856 in QGPL never fully saved. (I C)			
Cause : An attempt was made to delete a receiver that was never fully saved after the receiver was detached with a CHGJRN command.			
If this message was issued during automatic system cleanup (through Operational Assistant options or the STRCLNUP command), then the journal receiver AUDRCV2856 in library QGPL was deleted.			
Recovery : If the receiver is to be deleted anyway, enter I to			

```

        continue processing, otherwise, enter C to cancel processing.

        If this message was issued during automatic system cleanup
(through
    Operational Assistant options or the STRCLNUP command), you can
avoid the
    message by including the journal receivers in your normal backup
procedures.

More...
Reply . . . :   I

Press Enter to continue.

F3=Exit    F6=Print    F9=Display message details    F12=Cancel

```

Function key F9 is cursor sensitive for inquiry messages that have been replied to. If the cursor is placed on the reply value, F9 will show the message details of the reply message rather than the inquiry message itself. This way you can quickly determine who replied to the message and when the reply was sent. Placing the cursor anywhere else will cause the message details of the original message to be displayed when F9 is pressed. Function key F6 prints the displayed message and the message details to the current job's output queue. Cursor-sensitive help text is provided to explain all the AMI panel details.

Compared to the system QMHSCVLV function, I've omitted a couple of function keys, including F14=Work with problem and F21=Select assistance level. I might include the F14=Work with problem function key in a later version, but as for assistance level, I find no real value provided in the distinction between basic and intermediate level implemented in IBM's version, so I've simply combined what I find the best parts from both. Should you disagree, you of course have the option of making the appropriate changes in your copy of the source.

To give you an immediate chance to perform a test drive of this homegrown version of the Additional Message Information (AMI) panel, I've written and included a Display Message Queue (DSPMSGQ) command with this article, and it lists all entries of a specified message queue that meet the selection criteria entered, and for each message calls the CBX209 AMI program. Here's the DSPMSGQ command prompt:

```

                                Display Message Queue (DSPMSGQ)

Type choices, press Enter.

Message queue . . . . . Name
Library . . . . . *LIBL      Name, *LIBL,

```

```

*CURLIB
  Message type . . . . . *ALL          *ALL, *INFO, *INQ,
*COPY
  Messages to display first . . . *LAST      *LAST, *FIRST

  Severity code filter . . . . . 0         0-99, *MSGQ

```

The DSPMSGQ command displays the messages found in the specified message queue one by one. You can use the Message type (MSGTYP) and Severity (SEV) parameters to define the scope of messages to be displayed. The panel displaying the message details allows you to send a reply to inquiry messages. Pressing either F3 or F12 from the Additional Message Information display terminates the message queue processing and returns you to the display where you entered the DSPMSGQ command. For more details, please consult the command's help text.

This APIs by Example includes the following sources:

```

CBX209  -- RPGLE  -- Additional Message Information
CBX209E -- RPGLE  -- Additional Message Information - UIM Exit
Program
CBX209H -- PNLGRP -- Additional Message Information - Help

CBX209P -- PNLGRP -- Additional Message Information - Panel Group

CBX2091 -- RPGLE  -- Display Message Queue - CPP
CBX2091H -- PNLGRP -- Display Message Queue - Help
CBX2091V -- RPGLE  -- Display Message Queue - VCP
CBX2091X -- CMD    -- Display Message Queue

CBX209M -- CLP     -- Additional Message Information - Build objects

```

To create all these Additional Message Information objects as well as the Display Message Queue command objects, compile and run the CBX209M program, following the instructions in the source header. As always, the compilation instructions are also included in the respective source headers.

IBM i APIs supporting Message Queue and Message Key Attributes:

[Retrieve Writer Information \(QSPRWTRI\) API](#)

[Retrieve Printer Attributes \(QGYRPRTA\) API](#)

[Retrieve Job Information \(QUSRJOBI\) API \(6.1\)](#)

[Open List of Jobs \(QGYOLJOB\) API \(6.1\)](#)

[Retrieve Thread Attribute \(QWTRTVTA\) API \(6.1\)](#)

IBM documentation:

[A Primer on Message Analysis](#)

[Understanding What Controls the Automatic Reply Function](#)

This article demonstrates the following message handling APIs:

[Receive Nonprogram Message \(QMHRRCVM\) API](#)

[Retrieve Message \(QMHRTVM\) API](#)

[Retrieve Nonprogram Message Queue Attributes \(QMHRMQAT\) API](#)

[Send Reply Message \(QMHSNDRM\) API](#)

[Send Program Message \(QMHSNDPM\) API](#)

[Resend Escape Message \(QMHRSNEM\) API](#)

[Move Program Message \(QMHMOVPM\) API](#)

[Message Handling APIs](#)

[Message Handling Terms and Concepts](#)

[Retrieve the source code for this API example.](#)

Source URL: <http://iprodeveloper.com/rpg-programming/apis-example-message-handling-apis-additional-message-info-support>