



APIs by Example: Display Log Message Details - DSPHSTLOG Command, Continued

[System iNetwork Programming Tips Newsletter](#)

[Carsten Flensburg](#)

Carsten Flensburg

Thu, 07/28/2011 (All day)

The [Display History Log \(DSPHSTLOG\) command I presented in the preceding installment of APIs by Example introduced an enhanced interface to the system history log messages](#) (June 23, 2011, article ID 66046). The command supports new selection criteria in contrast to the native Display Log (DSPLOG) command, letting you easily filter to see only the messages of specific interest, and many message attributes were added to the DSPHSTLOG command's list panel's alternate view, including message ID, message severity, message type, qualified sending job name, and sending date and time.

The DSPHSTLOG command at this point, however, lacks an option to display the listed history log messages' formatted second-level message text and other information associated with the history log message. So with today's installment, I add an F9 function key to the history log message list panel. Positioning the cursor on the log message of interest and pressing F9 causes message details and second-level message text to be displayed. From the resulting display panel, you can then press F9 again to see all remaining log message details.

As I discuss in the preceding installment, the DSPHSTLOG command is based on the Open List of History Log Messages (QMHOHST) API, which returns the selected history log messages in a predefined format named HSTLO100. The HSTLO100 return format contains all the information necessary to retrieve and construct the message text and the second-level message text, respectively. Here's a list of some of the most prominent pieces of information embedded in the HSTLO100 format:

- Message ID
- Message severity
- Message type
- Message file
- Message file library
- Date sent
- Time sent
- Sending job name, user, and number
- Message text or immediate text
- Message replacement data
- Message text and data CCSIDs

For the history log message list initially displayed by the DSPHSTLOG command, the information directly available in the HSTLO100 format is sufficient. To display and format the second-level help text, however, some processing has to be done because, as far as the second-level message text is concerned, only the message replacement data is returned by the QMHOHST API. To understand

this requirement in more detail, let's look at a message description of a message often found in the history log—the CPF1164 message identifier sent whenever a job is ended:

Message identifier	CPF1164	Name
Message file	QCPFMSG	Name
Library	QSYS	Name, *LIBL, *CURLIB
First-level message text	'Job &3/&2/&1 ended on &5 at &4; &6 seconds used; end code &8 &13.'	
Second-level message text . . .	'&N Cause : Job &3/&2/&1 complet ed on &5 at &4 after it used &6 seconds processing unit time. The job had endin g code &8. The job ended after &7 routing steps with a secondary ending code of & 9. The job ending codes and their meanings are as follows: &P 0 - The job com pleted normally. &P 10 - The job completed normally during controlled ending or controlled subsystem ending. &P 20 - The job exceeded end severity (ENDSEV job a ttribute). &P 30 - The job ended abnormally. &P 40 - The job ended b ...	
Severity code	0	0-99, *SAME
Message data fields formats:		
Data type	*CHAR	*SAME, *NONE, *QTDCHAR...
Length	10	Number, *VARY
*VARY bytes or dec pos	0	Number
Data type	*CHAR	*QTDCHAR, *CHAR, *HEX...
Length	10	Number, *VARY
*VARY bytes or dec pos	0	Number
Data type	*CHAR	*QTDCHAR, *CHAR, *HEX...

Length	6	Number, *VARY
*VARY bytes or dec pos	0	Number
Data type	*CHAR	*QTDCHAR, *CHAR,
*HEX...		
Length	8	Number, *VARY
*VARY bytes or dec pos	0	Number
More...		

As you can see, the CPF1164 message description contains a number of substitution variables (&1, &2, &3, etc.), each defined by the subsequent *Message data field formats*. The QMHOLHST API returns only the replacement data required to replace these substitution variables, not the text surrounding the substitution variables. In order to construct a result displaying the final second-level text, the replacement data and the CPF1164 message description's second-level message text must be merged.

This merge is achieved by using the Retrieve Message (QMHRTVM) API, which takes a message ID, a message file, and message file library name together with the message replacement data and produces a second-level help text combining the static message text with the variable replacement data. I've demonstrated the QMHRTVM API in a couple of previous APIs by Example articles, to which I've provided links at the end of this article, so more details and examples of using this API can be found there. You can see an example of the CPF1164 message's fully substituted and formatted second-level help text later in this article.

Formatting the message text represents another challenge. In the CPF1164 message description above, notice a couple of format-control characters, &N and &P, used to signal where and how the message text should skip to the next line when displayed on a screen or printed. Format-control characters thereby enable you to optimize the readability and clarity of any given message description. Three different format-control characters are available. The following is an excerpt from the Add Message Description command's help text for the second-level message text explaining the visual effect of each format-control character:

Message help can be formatted for the work station using three format control characters. Each must be followed by a blank.

```
& N  Forces the message help to a new line (column 2). If
the help is longer than
      one line, the next lines are indented to column 4 until
the end of the help
      or until another format control character is found.
```

```

    & P Forces the message help to a new line, indented to
column 6. If the help is
        longer than one line, the next lines start in column 4
until the end of the
        help or until another format control character is found.

    & B Forces the message help to a new line, starting in
column 4. If the help is
        longer than one line, the next lines are indented to
column 6 until the end
        of the help or until another format control character is
found.

```

When running a native command displaying or printing message information, the system is in charge of turning the format-control characters into correspondingly formatted output. In the case at hand, this obligation lies with the API programmer. The first step in achieving correctly formatted output is taken by instructing the QMHRTVM API to include the format-control character in the returned second-level message text. The QMHRTVM API's ninth parameter defines whether format-control characters should be left in the returned message text or removed from it, so that's a quite simple task.

The next step is to locate the format-control characters in the second-level message text and to apply the formatting rules defined for each in the context given, which for both displayed and printed output equates to a 78-byte line of text. Both display and printer files are defined with an 80-byte width, leaving 78 bytes per line for the content. I've broken the formatting rules down into four associated subprocedures in the CBX234 program, which is responsible for producing the Additional Log Message Information evoked by the Display History Log list panel's function key F9:

<pre> FmtMsgStr() - Format message string string and text array </pre>	<pre> - formats a complete message returns the result in a </pre>
<pre> FmtMsgLin() - Format message line string into one </pre>	<pre> - formats a partial message message line </pre>
<pre> FndFmtIns() - Find format instruction control character in </pre>	<pre> - finds the next format a partial message string </pre>
<pre> GetIndPos() - Get indent positions message line most recently character </pre>	<pre> - establishes the amount of indentation required by the read format control </pre>

The FmtMsgStr() subprocedure runs the FmtMsgLin() subprocedure, processing the message string chunk by chunk. The FmtMsgLin() subprocedure in turn employs the FndFmtIns() and GetIndPos() subprocedures to format both the message text and the second-level message text. Only the latter

contains format-control characters, though. The core programming logic around the formatting task is displayed in the excerpt from the CBX234 program below:

```

/Free

a.    If  PxPrmLst.MsgId = *Blanks;
      MsgTxt = PxPrmLst.MsgDta;
      Else;
b.      MsgTxt = RtvMsgTxt( PxPrmLst.MsgFil_q
                          : PxPrmLst.MsgId
                          : PxPrmLst.MsgDta
                          );

      EndIf;

c.      MsgTxtFmt = FmtMsgStr( MSG_TXT + MsgTxt: NbrLinMsg );

d.      If  PxPrmLst.MsgId > *Blanks;
      SecLvl = RtvSecLvl( PxPrmLst.MsgFil_q
                        : PxPrmLst.MsgId
                        : PxPrmLst.MsgDta
                        );

e.      SecLvlFmt = FmtMsgStr( SecLvl: NbrLinSec );
      EndIf;

/End-Free

```

The steps outlined above are explained in the following section:

- a) If the log message processed is an immediate text, the message text returned by the QMHOLHST API is used directly.
- b) Otherwise the message text is retrieved from the message description by using the RtvMsgTxt() function and eventually the QMHRTVM API, specifying the qualified message file, the message ID, and the message data as input parameters.
- c) The message text is formatted by the FmtMsgStr() subprocedure and returned in the MsgTxtFmt array.
- d) If the log message processed is a predefined message, the second-level message text is retrieved from the message description by using the RtvSecLvl() function and eventually the QMHRTVM API, specifying the qualified message file, the message ID, and the message data as input parameters.
- e) The second-level message text is formatted by the FmtMsgStr()

```
subprocedure and
    returned in the SecLvlFmt array.
```

Subsequently the MsgTxtFmt and SecLvlFmt arrays in turn are written line by line to the Additional Log Message Information list panel and eventually either displayed or printed. To see what the outcome looks like, let me begin with the DSPHSTLOG command, which I ran with the following parameters:

```
DSPHSTLOG PERIOD((*AVAIL 220711) (*AVAIL *CURRENT))
```

The above command produced the output shown below, which pretty much resembles the example output shown last time, except for the new function key F9 included in the list panel's function key section:

Display History Log	
WYNDHAMW	22-07-11
20:32:07	
Message	
Job 129457/QTMHHTTP/QHTTP ended on 22-07-11 at 00:00:01; 0,018 seconds used; e	
Job 130049/QSYS/QYMEARCPMA started on 22-07-11 at 00:00:02 in subsystem QSYSWR	
Job 130050/QSYS/QYMEPFRCVT started on 22-07-11 at 00:00:02 in subsystem QSYSWR	
Log version QHST11202A in QSYS closed and should be saved.	
Job 130052/QPM400/Q1PPMSUB started on 22-07-11 at 00:00:02 in subsystem QSYSWR	
Job 130051/QPM400/Q1PDR started on 22-07-11 at 00:00:02 in subsystem QSYSWRK i	
Job 130053/QSYS/CRTPFRTDTA started on 22-07-11 at 00:00:02 in subsystem QSYSWRK	
Job 130052/QPM400/Q1PPMSUB ended on 22-07-11 at 00:00:02; 0,027 seconds used;	
Job 130054/QTMHHTTP/QHTTP started on 22-07-11 at 00:00:03 in subsystem QSYSWRK	
Job 130055/QSYS/QPMHDWRC started on 22-07-11 at 00:00:04 in subsystem QSYSWRK	
Job 130049/QSYS/QYMEARCPMA ended on 22-07-11 at 00:00:04; 0,135 seconds used;	
Job 130055/QSYS/QPMHDWRC ended on 22-07-11 at 00:00:05; 0,239 seconds used; en	
Job 130050/QSYS/QYMEPFRCVT ended on 22-07-11 at 00:00:06; 0,254 seconds used;	
Job 129455/QSYS/CRTPFRTDTA ended on 22-07-11 at 00:00:08; 0,865 seconds used; e	
Job 130056/VSIOWNER/RMTJRNMON started on 22-07-11 at 00:00:13 in subsystem OMS	
Job 130056/VSIOWNER/RMTJRNMON ended on 22-07-11 at 00:00:13; 0,014	

```
seconds use
  Job 130057/VSIOWNER/OMSBLDCST started on 22-07-11 at 00:00:13 in
subsystem OMS
  Job 130059/VSIOWNER/ASTRA7PG started on 22-07-11 at 00:00:13 in
subsystem OMS4

More...
F3=Exit    F9=Display message    F11=View 2    F12=Cancel    F17=Top
F18=Bottom
```

Placing the cursor on the first log message and pressing function key F9 brings up the Additional Log Message Information panel, showing message details as well as the formatted message text and second-level message text with the actual message replacement data in place of the substitution variables displayed in the CPF1124 message description shown earlier:

```

Additional Log Message Information

Message ID . . . . :   CPF1164                Severity . . . . . :    00

Message type . . . :   Completion

Date sent  . . . . :   22-07-11                Time sent  . . . . :
00:00:01

Message . . . . :   Job 129457/QTMHHTTP/QHTTP ended on 22-07-11 at
00:00:01;
  0,018 seconds used; end code 0

X.

Cause . . . . . :   Job 129457/QTMHHTTP/QHTTP completed on 22-07-11
at
  00:00:01 after it used 0,018 seconds processing unit time.  The
job had
  ending code 0. The job ended after 1 routing steps with a
secondary ending
  code of 0.  The job ending codes and their meanings are as
follows:
    0 - The job completed normally.

    10 - The job completed normally during controlled ending or
controlled
  subsystem ending.

    20 - The job exceeded end severity (ENDSEV job attribute).
```

```

30 - The job ended abnormally.

More...
Press Enter to continue.

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

```

The Additional Log Message Information panel supports cursor-sensitive help text explaining the panel's sections and fields, and pressing F6 lets you print the displayed output. F9 takes you to the Display Message Details panel, including the following message information:

```

                                Display Message Details

Message ID . . . . . :   CPF1164                Severity . . . . . :    00

Date sent  . . . . . :   22-07-11                Time sent  . . . . . :
00:00:01
Message type . . . . . :   Completion

From . . . . . :   QTMHHTTP                CCSID . . . . . :
65535

From job . . . . . :   QHTTP

User . . . . . :   QTMHHTTP

Number . . . . . :   129457

Time sent . . . . . :   00:00:01,160324

```



```

Bottom
Press Enter to continue.

```

```

F1=Help    F3=Exit    F12=Cancel

```

In contrast to the native DSPLOG command, the Display Message Details panel doesn't include the sending program (and, if available, the sending program instruction) for inquiry and reply type of messages. This omission is caused by the QMHOLHST API not including these two pieces of information in the API's HSTLO100 return format. Whether this is intentional or an oversight in the design of the QMHOLHST API, I don't know. I have, however, included the code necessary to provide for this information about the sending program, should it be added in the future. I'll let you know if that happens.

This APIs by Example includes the following sources:

```

CBX233  -- RPGLE  -- Display History Log - CCP
CBX233E -- RPGLE  -- Display History Log - UIM General Exit Program
CBX233L -- RPGLE  -- Display History Log - UIM List Exit Program
CBX233P -- PNLGRP -- Display History Log - Panel Group

CBX234  -- RPGLE  -- Additional Log Message Information

CBX234E -- RPGLE  -- Additional Log Message Information - UIM Exit
Program
CBX234H -- PNLGRP -- Additional Log Message Information - Help

CBX234P -- PNLGRP -- Additional Log Message Information - Panel Group

CBX234M -- CLP    -- Additional Log Message Information - Build
objects

```

To create all these objects, compile and run the CBX234M program, following the instructions in the source header. You'll also find compilation instructions in the respective source headers.

To support the Additional Log Message Information functionality, I've updated some of the sources provided in the June 23, 2011, article (the sources listed above with names beginning with CBX233). Be sure to download and replace these source members before you run the CBX234M program.

Related articles:

[APIs by Example: New Open List API Offers Enhanced Interface to History Log](#)

[APIs by Example: Message Handling APIs & Additional Message Info Support](#)

[i Can: The Secret History Log Enhancements](#)

[APIs by Example: Copying System i Message Descriptions](#)

[APIs by Example: Finding IBM i Message Descriptions](#)

This article demonstrates the following APIs:

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

[Open List of History Log Messages \(QMHOLHST\) API](#)

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

Source URL: <http://iprodeveloper.com/rpg-programming/apis-example-display-log-message-details-dsphstlog-command-continued>