


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

APIs by Example: New Open List API Offers Enhanced Interface to History Log

[System iNetwork Programming Tips Newsletter](#)

[Carsten Flensburg](#)

Carsten Flensburg

Thu, 06/23/2011 (All day)

Release 6.1 included a new Open List of History Log Messages (QMHOLHST) API, which in itself offers program access to the contents of the system history log but also further enhances the filtering options compared to the selection parameters offered by the old Display Log (DSPLOG) CL command. In previous installments of APIs by Example, I've presented a number of APIs and accompanying utilities that demonstrated the enhancement opportunities that more recent APIs often make available to API programmers, in comparison to older CL command counterparts.

In today's article, I continue this effort, taking advantage of the QMHOLHST API. In addition to an array of typical Open List API parameters, this API also includes a couple of parameters that let you configure the API output. You can control both the CCSID and the time zone applied to text- and time-related return information, respectively. I tie all this together in the new Display History Log (DSPHSTLOG) command that I present today and continue in an upcoming issue of this column.

As I've demonstrated in earlier APIs by Example articles, using APIs also offers opportunities in terms of adding to the information being displayed or listed as a result of the API-driven utility developed. In this case, the DSPHSTLOG command's list panel includes a separate list view immediately revealing the message ID, severity, and type, as well as sending time and job information. This is the intriguing part of API programming, the ability for you to tailor the functionality and usability to meet your exact requirements, thereby surpassing limitations imposed by the system's native offering.

To get started, let me introduce you to the QMHOLHST API's parameter list:

1	receiver variable	Output	Char (*)
2	length of receiver variable	Input	Binary (4)
3	format name	Input	Char (8)
4	list Information	Output	Char (80)
5	number of records to return	Input	Binary (4)
6	message selection information	Input	Char (*)
7	CCSID	Input	Binary (4)
8	time zone	Input	Char (10)
9	error code	I/O	Char (*)

The QMHOLHST API's first five parameters are common to most *Open List APIs*. The receiver variable, the length of the receiver variable, and the format name all define the area available for the API to return its information as well as the format of the information returned. The fourth parameter is an API standard *List information* data structure returned by the Open List API in order to

communicate the outcome of the API call back to the caller of the API. This information includes the following:

- the total number of records available in the list
- the number of records returned in the receiver variable
- a request handle uniquely identifying the list being built
- the fixed record length of each block of information returned in the receiver variable. For variable-length records, the length is returned in the record
- an *information complete* indicator defining the condition of the information returned
- the date and time the list was created
- a *list status* indicator signaling how to proceed with the list processing
- the size, in bytes, of the information returned in the receiver variable
- the ordinal number in the list of the first record returned in the receiver variable

All this information lets you navigate the open list and process the returned information correctly, without having to make assumptions or hard-code lengths and sizes. I go into more detail about this later.

The fifth parameter, the *number of records to return*, in the context of Open list APIs usually serves two purposes. In the case at hand, there's even a third function of this parameter. The most obvious information conveyed by this parameter is of course how many records you want the QMHOLHST API to return in the receiver variable. The not-so-obvious implication of this parameter is *how* the list is built:

- Specifying a zero for this parameter causes the list to be built asynchronously by a separate server job. No records will be returned when calling the API, and control will be returned to the caller immediately, but the list build will commence and make list records available for subsequent calls to the Get List Entries (QGYGTLE) API, which will then be in charge of processing the entire list.
- Specifying a -1 in turn causes the list to be built synchronously in the same job calling the QMHOLHST API. This implies that control isn't returned to the calling program until the complete list has been built. As many records as will fit into the receiver variable are returned to the API caller, and the rest of the available records must then be processed by means of calls to the QGYGTLE API.
- If a positive number of records is specified, at least that many records are built synchronously (in order to return those records immediately to the caller of this API), and the remainder are built asynchronously by a server job. The remainder of the records in the list can be accessed with the QGYGTLE API.

The QMHOLHST API further uses *number of records* to return a parameter for a purpose relating to the system's implementation of the history log function. All messages sent to the history log message queue QHST in library QSYS are eventually written to the QHST history log files. These writes are, however, buffered, meaning that at some point in time messages may exist in the QHST log message queue that haven't yet been written to the QHST log files. Because the QMHOLHST API is retrieving the history log messages from the QHST log files, and not the ditto message queue, the API supports the special value -2 as the number of records to return. Here's why:

Specifying the value -2 for the parameter in question causes the QMHOLHST API to flush all job messages in the write buffer to the QHST log files, letting a following API call process all history log messages sent at that point in time. For a flush call to the QMHOLHST API to be allowed, you must also specify a zero for the size of the receiver variable parameter, indicating that the regular purpose

of the API is suspended and that no other processing beyond the write buffer flush of the history log is performed.

Once you've completed processing the last history log message returned for one QMHOLHST API call, you can interrogate whether new log messages have arrived in the history log in the meantime by repeating the flush call, followed by a regular call to the QMHOLHST API. For the new call to pick up where you left off, specify the sent date and time from the last history log message received for the starting date and time parameter on the API call.

The sixth API parameter defines the selection information providing the parameters used to include—or exclude—certain history log messages from the returned log message list. The selection information is specified in a data structure format containing the following information:

- start date and time
- end by date and time
- include or exclude specified message IDs indicator
- list of message IDs to either include in or exclude from the returned list
- include or exclude specified message types indicator
- list of message type to either include in or exclude from the returned list
- message severity limit
- list of qualified job names for which to return history log messages

The above selection criteria are all supported by the QMHOLHST API directly, meaning that only log messages meeting the specified selection criteria are returned by the API. Due to its usefulness, I've further added a selection criterion letting you select log messages sent by a specific current user profile in the sending job. This criterion is therefore enforced by the DSPHSTLOG CPP while processing the returned log message list. Note that for release 6.1, a maximum of 100 message IDs can be specified to be either selected or omitted. At release 7.1, this maximum was increased to 200 message IDs.

The seventh QMHOLHST API parameter lets you tell the API which CCSID should be applied when returning textual data and *CCHAR type message data to the API caller. Specifying a zero for this parameter causes the API to return textual information as well as *CCHAR-type message data in the CCSID of the job calling the API. Specifying a value of 65535 causes the mentioned data to be returned in the CCSID in which it was created, meaning that no conversion is performed.

Further, there's an eighth parameter controlling the time zone used for both dates and times used as input to the QMHOLHST API as well as the date and time values in log messages returned by the API. Again, the API is in charge of the necessary conversion operation. You can specify the following values for this parameter:

```
*SYS  Local system value associated with the time zone is specified
by the time zone
      system value

*UTC  Coordinated Universal Time (UTC) value.

*JOB  Local job time value and the associated time zone is specified
by the job
      attribute.

Time
zone  Specifies the name of a time zone description(*TIMZON) object.
```

The final API parameter is the well-known API standard error data structure discussed in much detail in earlier articles, so I leave that out of scope for this article.

To demonstrate the programming logic involved in building and retrieving the list returned by an open list API, I've extracted code snippets from the DSPHSTLOG command's UIM list exit program and adapted it a little to reflect mainly the core steps performed in such an operation:

```

**-- Receiver variable record structure:
D HSTL0100          Ds          32767    Qualified  Based( pHSTL0100 )
D  EntLen           10i  0
D  MsgSev           10i  0
D  MsgId            7a
...
**-- Global constants:
D BLD_SYNCH         c              -1
D MAX_ENT           c          999999
**-- List information:
D LstInf            Ds              Qualified
D  RcdNbrTot        10i  0
D  RcdNbrRtn        10i  0
D  Handle           4a
D  RcdLen           10i  0
D  InfSts           1a
D  Dts              13a
D  LstSts           1a
D                   1a
D  InfLen           10i  0
D  Rcd1             10i  0
D                   40a

/Free

a.  LstApi.RtnRcdNbr = 1;
    LstApi.LstRcdNbr = *Zero;

b.  LstHstLog( RcvVar
              : %Size( RcvVar )
              : 'HSTL0100'
              : LstInf
              : BLD_SYNCH
              : SltInf
              : JOB_CCsid
              : PrmRcd.TimZon
              : ERRC0100
              );

c.  If  ERRC0100.BytAvl = *Zero  And  LstInf.RcdNbrRtn > *Zero;
      ExSr  PrcLstEnt;
    EndIf;

    BegSr  PrcLogLst;

```

```

d.      If  LstApi.RtnRcdNbr > *Zero;
          LstApi.CurRcdNbr = *Zero;

          DoW  LstInf.RcdNbrTot > LstApi.RtnRcdNbr;
              LstApi.RtnRcdNbr += 1;

              GetOplEnt( RcvVar
                          : %Size( RcvVar )
                          : LstInf.Handle
                          : LstInf
                          : MAX_ENT
                          : LstApi.RtnRcdNbr
                          : ERRC0100
                          );

              If  ERRC0100.BytAvl > *Zero;
                  Leave;
              EndIf;

              ExSr  PrcLstEnt;

          EndDo;
      EndIf;

  EndSr;

e.      BegSr  PrcLstEnt;

f.      pHSTL0100 = %Addr( RcvVar );

g.      For Idx = 1  To LstInf.RcdNbrRtn;

          LstApi.LstRcdNbr += 1;

          ... process list entry

h.      If  Idx

```

The process outlined in the above code snippets maps to the explanation below:

a) The record number to be returned is set to 1, i.e., the first re

- b) The QMHOLHST API is called, requesting the list to be built synchronously.
- c) If the API call is successful and at least one record is returned, the first record is processed in e).
- d) If more records are available in the list than were returned in the Get List Entry API (QGYGTLE) is called to retrieve the remainder of the list, which are then processed in e). The QGYGTLE API call is repeated until all list records have been retrieved.
- e) The API return format data structure's address is set to the location of the first list record returned.
- f) Each list entry returned is processed.
- g) If more list entries are available, the return format data structure is moved to the next entry's location. Otherwise processing continues with step a).

The Open List API's ability to return as many records as fit into a large API receiver variable significantly reduces the number of API calls needed to retrieve the complete list, and thereby of course is speeding up the history log processing proportionally. Anyway, translating all the QMHOLHST API parameters explained earlier into a CL command interface in turn equates to the following DSPHSTLOG prompt:

Display History Log (DSPHSTLOG)

Type choices, press Enter.

```

Time period for log output:

  Start time and date:

    Beginning time . . . . . *AVAIL      Time, *AVAIL

    Beginning date . . . . . *CURRENT    Date, *CURRENT,
*BEGIN
  End time and date:

    Ending time . . . . . *AVAIL      Time, *AVAIL

    Ending date . . . . . *CURRENT    Date, *CURRENT,
*END
  Message selection:

    Selection indicator . . . . . *ALL      *ALL, *INCLUDE,
*OMIT
    Message identifier . . . . .         Name

        + for more values

  Type selection:

    Selection indicator . . . . . *ALL      *ALL, *INCLUDE,
*OMIT
    Message type . . . . .         *COMP, *COPY,
*DIAG...
        + for more values

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

    Current user profile . . . . . *ALL      Name, *ALL

    Jobs to display . . . . . *ALL      Name, *ALL

    User . . . . .         Name

    Number . . . . .         000000-999999

        + for more values


    Time zone . . . . . *JOB      Name, *JOB, *SYS,
*UTC
    Output . . . . . *          *, *PRINT

```

As you'll note, apart from displaying the resulting output in a list panel, you can direct the output to a printed list. All DSPHSTLOG command parameters are further explained in the accompanying help text panel group. To produce an example of the DSPHSTLOG command's list panel, I ran the following command on my system:

```
DSPHSTLOG PERIOD((*AVAIL 150611) (*AVAIL *CURRENT))
```

The above command returned the history log list displayed below. You'll note that the list is similar to the one produced by the system's own DSPLOG command, when initially displayed:

Display History Log	
WYNDHAMW	16-06-11
08:59:58	
Message	
Job 063852/QPM400/Q1PDR started on 15-06-11 at 00:00:00 in subsystem QSYSWRK i	
Job 063853/QPM400/Q1PPMSUB started on 15-06-11 at 00:00:00 in subsystem QSYSWR	
Job 061867/QTMHHTTP/QHTTP ended on 15-06-11 at 00:00:02; 0,033 seconds used; e	
Job 063854/QSYS/QYMEARCPMA started on 15-06-11 at 00:00:03 in subsystem QSYSWR	
Job 063855/QSYS/QYMEPFRCVT started on 15-06-11 at 00:00:03 in subsystem QSYSWR	
Job 063853/QPM400/Q1PPMSUB ended on 15-06-11 at 00:00:04; 0,028 seconds used;	
Job 063856/QSYS/CRTPFRTDTA started on 15-06-11 at 00:00:04 in subsystem QSYSWRK	
Job 063857/QSYS/QPMHDWRC started on 15-06-11 at 00:00:06 in subsystem QSYSWRK	
Job 063854/QSYS/QYMEARCPMA ended on 15-06-11 at 00:00:07; 0,318 seconds used;	
Job 063855/QSYS/QYMEPFRCVT ended on 15-06-11 at 00:00:08; 0,597 seconds used;	
Job 063858/QTMHHTTP/QHTTP started on 15-06-11 at 00:00:08 in subsystem QSYSWRK	
Job 063859/QSYS/QPMRSYSCMD started on 15-06-11 at 00:00:08 in subsystem QSYSWR	
Job 061866/QSYS/CRTPFRTDTA ended on 15-06-11 at 00:00:11; 2,556 seconds used; e	
Job 063859/QSYS/QPMRSYSCMD ended on 15-06-11 at 00:00:14; 0,103 seconds used;	
Job 063852/QPM400/Q1PDR ended on 15-06-11 at 00:00:16; 2,421 seconds used; end	
Job 063860/VSIOWNER/RMTJRNMON started on 15-06-11 at 00:00:17 in subsystem OMS	
Job 063860/VSIOWNER/RMTJRNMON ended on 15-06-11 at 00:00:17; 0,017 seconds use	
Job 063861/VSIOWNER/OMSBLDCST started on 15-06-11 at 00:00:17 in subsystem OMS	

More...

F3=Exit F11=View 2 F12=Cancel F17=Top F18=Bottom

I've added system date, time, and name information to the panel's header section, and also a few function keys that let you navigate to the top and the bottom of the list, as well as toggle between the above view and a new one that exposes a range of history log message details otherwise available only when looking up this information individually for each message. In the native DSPLOG command's list panel, the latter is done by using a combination of cursor location and function key F1. Here's how the new alternate view appears on the Display History Log list panel:

```

                                Display History Log

WYNDHAMW                                16-06-11
08:59:58
Msg ID   Sev  Type   From job   Number  User       Date       Time

  CPF1124  00   *INFO  Q1PDR      063852  QPM400     15-06-11
00:00:00,771566
  CPF1124  00   *INFO  Q1PPMSUB   063853  QPM400     15-06-11
00:00:00,800297
  CPF1164  00   *COMP  QHTTP      061867  QTMHHTTP   15-06-11
00:00:02,091363
  CPF1124  00   *INFO  QYMEARCPMA 063854  QSYS       15-06-11
00:00:03,789902
  CPF1124  00   *INFO  QYMEPFRCVT 063855  QSYS       15-06-11
00:00:03,887443
  CPF1164  00   *COMP  Q1PPMSUB   063853  QPM400     15-06-11
00:00:04,181396
  CPF1124  00   *INFO  CRTPFRTDTA 063856  QSYS       15-06-11
00:00:04,239252
  CPF1124  00   *INFO  QPMHDWRC   063857  QSYS       15-06-11
00:00:06,593122
  CPF1164  00   *COMP  QYMEARCPMA 063854  QSYS       15-06-11
00:00:07,506893
  CPF1164  00   *COMP  QYMEPFRCVT 063855  QSYS       15-06-11
00:00:08,114219
  CPF1124  00   *INFO  QHTTP      063858  QTMHHTTP   15-06-11
00:00:08,170259
  CPF1124  00   *INFO  QPMRSYSCMD 063859  QSYS       15-06-11
00:00:08,496378
  CPF1164  00   *COMP  CRTPFRTDTA 061866  QSYS       15-06-11
00:00:11,735984
  CPF1164  00   *COMP  QPMRSYSCMD 063859  QSYS       15-06-11
00:00:14,980942
  CPF1164  00   *COMP  Q1PDR      063852  QPM400     15-06-11
00:00:16,914267
  CPF1124  00   *INFO  RMTJRNMN   063860  VSIOWNER   15-06-11
00:00:17,107948
  CPF1164  00   *COMP  RMTJRNMN   063860  VSIOWNER   15-06-11
00:00:17,114010

```

```

CPF1124  00  *INFO  OMSBLDCST  063861  VSIOWNER  15-06-11
00:00:17,370145

More...
F3=Exit    F11=View 1    F12=Cancel    F17=Top    F18=Bottom

```

As always, all parts of the panel as well as the list columns are documented in the cursor-sensitive help text activated by using function key F1. The DSPHSTLOG command at this point, however, lacks the option of displaying all the information associated with an individual message, including the second-level message text and full qualified sending job name, in a separate panel. I'll dive into more details on this topic and add this missing feature in the next issue of APIs by Example.

This APIs by Example includes the following sources:

```

CBX233  -- RPGLE  -- Display History Log - CCP
CBX233H -- PNLGRP -- Display History Log - Help
CBX233L -- RPGLE  -- Display History Log - UIM List Exit Program
CBX233P -- PNLGRP -- Display History Log - Panel Group
CBX233X -- CMD    -- Display History Log

CBX233M -- CLP    -- Display History Log - Build command

```

To create all these objects, compile and run the CBX233M program, following the instructions in the source header. You'll also find compilation instructions in the respective source headers.

IBM Documentation:

[Process Open List APIs](#)

[QHST history log](#)

This article demonstrates the following APIs:

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

[Get List Entries \(QGYGTLE\) API](#)

[Close List \(QGYCLST\) API](#)

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

Source URL: <http://iprodeveloper.com/rpg-programming/apis-example-new-open-list-api-offers-enhanced-interface-history-log>