print | close

APIs by Example: Watch Definitions and the Start Watch (QSCSWCH) API

iPro Developer
Carsten Flensburg
Carsten Flensburg
Tue, 05/29/2012 - 6:00am

Store predefined watches in a table with these watch definition CL commands

Click here to download the code bundle.
To report code errors, email **iProDeveloper.com**

Since release 5.4, we've had the ability to define and start watches on your IBM i. A watch is a service monitor process that calls a user-written exit program when a predefined event occurs on your system. Besides monitoring the Licensed Internal Code (LIC) log and the Product Activity Log (PAL), a watch can monitor for specific or generic messages in a job log or all messages of a specific type issued in any job or named jobs on the system. The same range of parameters is also available for monitoring one or more nonprogram message queues, including QSYSOPR and the history log.

You can also associate watches with traces when issuing the Trace Internal (TRCINT), Start Trace (STRTRC), Trace Connection (TRCCNN), Trace TCP/IP Application (TRCTCPAPP), or Start Communications Trace (STRCMNTRC) commands that specify watches parameters. Here, the watches in scope are the former ones initiated using the Start Watch (STRWCH) command or the Start Watch (QSCSWCH) API. The STRWCH command and the QSCSWCH API have been significantly enhanced in both release 6.1 and 7.1. When it comes to system and job monitoring, watches are a powerful tool.

Supported Message ID–type Parameters

Typically, the use of watches relates to permanent monitoring for certain message IDs or message types in job logs or message queues. You employ PAL-, LIC log–, and Trace-associated watches mainly when investigating specific operating system or communication issues. Message ID–type watches support the following range of parameters to identify the events to monitor:

- Message ID
  - *ALL message IDs
  - *Generic message ID
  - Message ID
- Message type
  - *ALL
  - *COMP, *DIAG, *ESCAPE, etc.
- Message severity
  - 0–99
- Message data comparison
  - *MSGDTA
  - *FROMPGM
  - *TOPGM
- Message queue
  - *SYSOPR
  - *HSTLOG
  - *JOBLOG
  - Message queue
- Watched job
  - Job name, *ALL
  - User profile, *ALL
  - Job number, *ALL

Note that the *Watched job* parameter is allowed only when you specify the *Message queue* parameter special value *JOBLOG. Combining the comprehensive set of criteria available lets you pinpoint the specific event to monitor and then take the appropriate action in the associated exit program. Also note that while the Start Watch CL command can't exceed five message ID parameters, the Start Watch API supports up to 100 different message IDs, depending on the values indicated for the *Message queue* and *Watched job* parameters.

Watches, however, are not persistent across IPLs. Therefore, the evoking Start Watch command or API call must be repeated following an IPL or after an End Watch command or End Watch API call issued against a watch. Given the increased number of start watch parameters and the enhanced functionality associated with them, I've created several watch definition CL commands that let you store predefined watches in a table on your system. You can then restart your watches automatically, either as part of your system startup program or as a subsystem autostart job.

The QSCSWCH API Parameter List

The first CL command—Add Watch Definition (ADDWCHDFN)—uses the Start Watch (QSCSWCH) API to start the watch after all the watch definition details have been stored in the watch definition table (to download all the source files, go to iProDeveloper.com/code). Figure 1 shows the QSCSWCH API parameter list as it appears in the IBM Information Center's API section.

Figure 1: The Start Watch (QSCSWCH) API parameter list

```
Required Parameter Group:

    1  Session ID                        Input     Char(10)
    2  Started session ID                Output    Char(10)
    3  Watch program                     Input     Char(20)
    4  Watch for message                 Input     Char(*)
    5  Watch for LIC log entry           Input     Char(*)
    6  Error Code I/O Char(*)

    Optional Parameter Group 1:

    7  Watch session attributes          Input     Char(*)

    Optional Parameter Group 2:

    8  Watch for PAL entry               Input     Char(*)

  Default Public Authority: *EXCLUDE
```

The *Session ID* parameter specifies the name of the session to start. You can use the special value *GEN to have the API generate the session ID for you. In that case, the generated session identifier is returned in the second API parameter, *Started session ID*. The *Watch program* parameter defines the qualified name of the watch exit program to call when the watch event occurs. API parameters 4, 5, and 8 are mutually exclusive and specify whether the watch should monitor for a Message ID event, an LIC log event, or a Product Activity Log event. I'll get back to these parameters in a moment.

The sixth parameter, the standard API error data structure, probably requires no further explanation. The seventh (optional) parameter, *Watch session attributes*, is a data structure where you specify whether to call the watch program, along with the watched-for event, when the watch is started and/or is ended. This parameter also lets you indicate the run priority for the job where the watch session work will run.

Back to API parameters 4, 5, and 8, which are similar in the way you define them. These parameters' first four bytes hold an integer that counts the number of subsequent repeating parameter data structure entries. The *Watch for message* parameter supports up to 100 message entries, and the *Watch for LIC log entry* and *Watch for PAL entry* parameters each allows up to five entries. But as mentioned, you can specify only one of these three parameters for a single watch. For the two excluded parameters, you must set the parameter entry count to zero.

The first parameter data structure entry immediately follows the parameter entry count, and the initial four bytes of each data structure contains the actual total length of the current entry. To get to the beginning of the next entry, simply add this length to the current entry's starting point. For API performance reasons, the API documentation recommends keeping each data structure entry on a four-byte boundary. If necessary, adjust each total entry length to ensure that it's evenly divisible by 4.

The many possible combinations of message IDs, message queues, and jobs make initializing the *Watch for message* entry parameter structure particular challenging. To specify more than one message queue for a single message ID, you must submit a corresponding number of *Watch for message* parameter structure entries. Likewise, if you indicate message queue *JOBLOG and want to specify more than one job to monitor, you must define a *Watch for message* parameter structure entry for each job.

For example, to monitor for security violation–related messages in the range CPF2200 to CPF2299 in message queue QSYSOPR as well as in job logs for all jobs with names beginning with CBX* and WEB*, respectively, submit the command in Figure 2,

Figure 2: Command to monitor for security violation messages

```
STRWCH SSNID(MONSECVIOL)
         WCHPGM(CBX250)
         CALLWCHPGM(*WCHEVT)
         WCHMSG((CPF22* *NONE *MSGDTA *ALL *GE 00))
         WCHMSGQ((*SYSOPR) (*JOBLOG))
         WCHJOB((*ALL/CBX*) (*ALL/WEB*))
```

which will start the watch session in Figure 3. The single message ID in turn causes three watch instances to activate—one for each message queue and one for each message queue/job combination.

Figure 3: Watch session

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -


                                 Display Watch
                                                             System:   WYNDHAMW

 Session ID . . . . . :   MONSECVIOL       Started:
  Started by:                                Date . . . . . . . :   12-03-12
    Job name . . . . . :   WYNDH001A         Time . . . . . . . :   10:50:02
    User . . . . . . . :   CARSTEN         Call exit program:
    Number . . . . . . :   570870                                   *WCHEVT
  Watch program  . . . :   CBX250
    Library  . . . . . :     QGPL
  Origin . . . . . . . :   STRWCH
  Run priority . . . . :   25
           Message     ----Message queue-----  -------------Job--------------
  Message  type        Name        Library     Name        User        Number
  CPF22*   *ALL        QSYSOPR     QSYS
  CPF22*   *ALL        *JOBLOG                 CBX*        *ALL        *ALL
  CPF22*   *ALL        *JOBLOG                 WEB*        *ALL        *ALL

                                                                         Bottom
 Press Enter to continue

 F1=Help   F3=Exit   F11=Message details   F12=Cancel


- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Of course, as an API programmer, you're tasked with resolving the watch specification into the correct number and combination of *Watch for message* parameter structures. So I present a take on how to solve that challenge in the ADDWCHDFN command processing program (CPP) CBX247. You can use the source debugger of your choice to step through the program, as the ADDWCHDFN command parameter values are extracted from the CPP parameter structures in the CPP's GetMsgPrm subroutine and subsequently appended to the QSCSWCH API's input parameter structure (described earlier) in the LodMsgStr subroutine.

Figure 4 shows the fully prompted ADDWCHDFN command.

Figure 4: Add Watch Definition (ADDWCHDFN) command prompt

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

                        Add Watch Definition (ADDWCHDFN)

 Type choices, press Enter.

 Session ID . . . . . . . . . .                    Name
 Watch program  . . . . . . . .                    Name
   Library  . . . . . . . . . .     *LIBL          Name, *LIBL, *CURLIB
 Call watch program . . . . . .   *WCHEVT          *WCHEVT, *STRWCH, *ENDWCH

 Watch for message:
   Message to watch . . . . . .   *NONE            Name, generic*, *NONE...
   Comparison data  . . . . . .

   Compare against  . . . . . .                    *MSGDTA, *FROMPGM, *TOPGM
   Message type . . . . . . . .                    *ALL, *COMP, *DIAG...
   Relational operator  . . . .                    *GE, *EQ, *GT, *LT, *LE
   Severity code  . . . . . . .                    0-99
             + for more values
Watched message queue:
   Message queue  . . . . . . .   *SYSOPR          Name, *SYSOPR, *JOBLOG...
     Library  . . . . . . . . .     *LIBL          Name, *LIBL
             + for more values
 Watched job:
   Job name . . . . . . . . . .   *                Name, generic*, *, *ALL
     User . . . . . . . . . . .                    Name, generic*, *ALL
     Number . . . . . . . . . .                    000001-999999, *ALL
             + for more values
 Watch for LIC log entry:
   Major code . . . . . . . . .   *NONE            0000-FFFF, *ALL, *NONE
   Minor code . . . . . . . . .                    0000-FFFF, *ALL
   Comparison data  . . . . . .

   Compare against  . . . . . .                    *ALL, *TDENBR, *TASKNAME...
             + for more values
Watch for PAL entry:
   System reference code  . . .   *NONE            Character value, *NONE, *ALL
   Comparison data  . . . . . .                    Character value, *NONE
   Compare against  . . . . . .                    *RSCNAME, *RSCTYPE, *RSCMODEL
             + for more values
 Run priority . . . . . . . . .   25               1-99
 Text 'description' . . . . . .   *BLANK


- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Comprehensive help text included with ADDWCHDFN documents the command and its parameters. In the Remove Watch Definition (RMVWCHDFN) command prompt (Figure 5), the End Watch (ENDWCH) parameter displays only if the specified watch is currently active.

Figure 5: Remove Watch Definition (RMVWCHDFN) command prompt

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

                       Remove Watch Definition (RMVWCHDFN)

 Type choices, press Enter.

 Session ID . . . . . . . . . .                    Name
 End watch  . . . . . . . . . .   *NO              *YES, *NO


- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

The accompanying help text panel group explains the command and its parameters.

I'll present more Watch Definition commands as well as a watch exit program example detailing the watch program interface in upcoming installments of the APIs by Example column.

How to Compile

Below, you'll find instructions on how to create the Add Watch Definition and Remove Watch Definition commands. The watch definition itself is stored in the table CBX247F, which resides in library QGPL. If you prefer another library, feel free to change the library name in the respective RPGLE sources accordingly.

- CBX247—RPGLE: Add Watch Definition—CPP
- CBX247H—PNLGRP: Add Watch Definition—Help
- CBX247V—RPGLE: Add Watch Definition—VCP
- CBX247X—CMD: Add Watch Definition

- CBX248—RPGLE: Remove Watch Definition—CPP
- CBX248H&mdashPNLGRP: Remove Watch Definition—Help

- CBX248V—RPGLE: Remove Watch Definition—VCP
- CBX248P—RPGLE: Remove Watch Definition—PCP
- CBX248X—CMD: Remove Watch Definition

- CBX247—CLP: Watch Definition Commands—Build commands

To create all above command objects, compile and run the CBX247M CL program, and follow the instructions in the source header. You'll also find compilation instructions in the respective source headers of the individual sources. Note that the code included with this article takes advantage of the most recent enhancements and must be adapted to run successfully on pre-7.1 releases.

Find Out More

- i Can ... Automate Monitoring With Watches

IBM i 7.1 Information Center documentation:

- End Watch (QSCEWCH) API
- Monitoring APIs
- Retrieve Watch Information (QSCRWCHI) API
- Start Watch (QSCSWCH) API

Articles at systeminetwork.com:

- "The Administrator: Watch out!"
- "Tech Tips: Watching for messages"
- "Watching for Messages — Made Easy"

Source URL: http://iprodeveloper.com/rpg-programming/apis-example-watch-definitions-and-start-watch-qscswch-api