

APIs by Example: One Job Schedule Entry API and Four New Job Schedule Entry Commands

[System iNetwork Programming Tips Newsletter](#)

[Carsten Flensburg](#)

Carsten Flensburg

Thu, 02/24/2011 (All day)

In today's APIs by Example, I demonstrate the List Job Schedule Entries (QWCLSCDE) API, which has been around since its introduction with release 2.2 of the IBM i OS. Certainly, this is an aged and typical API in terms of listing its output to a user space (a topic I've covered on earlier occasions in this column). The primary reason for revisiting this API now is a reader request for an API example enhancing the native offer of job schedule entry commands, including the functionality provided by these commands.

One of the requirements that the reader needed to be met included creating a Copy Job Schedule Entry (CPYJOBSCDE) command. Another involved the need to add further selection criteria to the Work with Job Schedule Entries (WRKJOBSCDE) command as well as add more details to the printed output this command offers. In the process, I further decided to include a Display Job Schedule Entry (DSPJOBSCDE) command to support a direct display of a known schedule entry. And the inclusion of a Submit Job Schedule Entry (SBMJOBSCDE) command enabled me to take long-needed control of the job attributes employed when a job schedule entry is submitted immediately.

The workhorse for all four commands is the QWCLSCDE API, whose parameter list is brief and simple, as shown by the following excerpt from the API's documentation:

Required Parameter Group:

1	Qualified user space name	Input	Char(20)
2	Format name	Input	Char(8)
3	Job schedule entry name	Input	Char(10)
4	Continuation handle	Input	Char(16)
5	Error code	I/O	Char(*)

The first parameter defines the qualified name of the user space to which the QWCLSCDE API should return the job schedule entry list. The second parameter identifies the format and detail level of the returned job schedule entries. The SCDL0100 format returns basic (partial) job schedule entry information, whereas the SCDL0200 format returns detailed (full) job schedule entry information.

The third parameter specifies the range of job schedule entries to return. You can specify a specific job schedule entry name, a generic job schedule entry name, or the special value *ALL to return all job schedule entries in the system's default job scheduler. If you choose *ALL, and the number of job schedule entries exceeds the capacity of the user space, the fourth parameter, the continuation handle, will help you get away with it.

The continuation handle must always contain blanks on the first API call, to instruct the API to build the list from the beginning. If, while building the list, the API runs out of user space capacity, it returns a partial list as well as a continuation handle in the Header section of the user space. To have the API continue where it left off on the next API call, you specify the returned continuation handle as the fourth API parameter. You can see an example of how to turn this procedure into RPG/IV code below:

```

/Free

DoU  CntHdl = *Blanks;
     LstJobScdE( UsrSpc_q
                 : 'SCDL0200'
                 : PxJobNam
                 : CntHdl
                 : ERRC0100
                 );

     If  ERRC0100.BytAvl > *Zero;
       ExSr  EscApiErr;
     Else;
       pHdrInf = pUsrSpc + UsrSpc.OfsHdr;
       pLstEnt = pUsrSpc + UsrSpc.OfsLst;

       For  Idx = 1  To  UsrSpc.NumLstEnt;
         ExSr  PrcLstEnt;

         If  Idx

```

Now on to the default job scheduler that comes free with IBM i and which lets you perform simple job scheduling for individual jobs that must run on certain dates or days of the week or month. If you've created a program to perform nightly updates to your accounting system, and you want that program to run at business close on all work days, you could ask the job scheduler to do that for you by issuing the following command:

```

ADDJOBSCDE JOB(ACCUPD)

          CMD(CALL PGM(ACCUPD01))

          FRQ(*WEEKLY)

          SCDDATE(*NONE)

          SCDDAY(*MON *TUE *WED *THU *FRI)

```

```

SCDTIME (20:00:00)

JOB (ACCUPD01)

JOBQ (*JOBQ)

USER (ACCOPR)

MSGQ (*USRPRF)

TEXT ('Nightly update on work days')
```

You can also specify what to do if the system is not running at the point in time the job schedule entry is designated to run; skip running the job, submit the job held, or submit the job released. This and all other job schedule entry attributes are explained in more detail in the IBM documentation that I've added links for at the end of this article. For more advanced job schedule planning, which includes controlling dependencies between jobs and many more scheduling options, IBM also offers a billable product called *Advanced Job Scheduler for i5/OS (5761-JS1 for release 6.1)*, for which I've also provided a link to the IBM documentation.

If you want to schedule a job to run only once, the Submit Job (SBMJOB) command also lets you specify a scheduled date and time and thereby have the system take care of running the job at that specific time, as in the following example:

```

SBMJOB CMD (CALL PGM (ACCUPD02) )
        JOB (ADDUPD2)
        JOB (ACCUPD01)
        SCDDATE (28-02-11)
        SCDTIME (20:00:00)
```

Anyway, as for the task at hand and addressing the work list in the above order, the CPYJOBSCDE command is straightforward from a design standpoint. The QWCLSCDE API is capable of returning all the information for any given job schedule entry. A job schedule entry is uniquely identified by two key attributes, the job schedule entry *name* and the job schedule entry *number*. If there's only one job schedule entry having a specified name, the special value *ONLY should be supported for the job schedule entry number, in order to avoid having to look up the specific entry number. These requirements lead to the following very simple initial CPYJOBSCDE command prompt:

Copy Job Schedule Entry (CPYJOBSCDE)

Type choices, press Enter.

From job name	Name
Entry number *ONLY	000001-999999,
*ONLY	

Once the user has completed entering the job schedule entry information required to identify the entry to copy and has pressed Enter, I use a command prompt override program (POP) to retrieve and fill in all the entry details needed to create a new job schedule entry. In case you want to read up on POP programming, I recently explained the POP concept in more detail in the APIs by Example article "New Data Queue Attributes and New Data Queue Command," to which you'll find a link at the end of this article.

One of the parameters used in the CPYJOBSCDE CPP to eventually add the job schedule entry using the Add Job Schedule Entry (ADDJOBSCDE) command needed special attention. The relative day of month (RELDAYMON) parameter is valid only if a value is specified for the command's SCDDAY parameter and FRQ(*MONTHLY) is specified. This means that I'm not allowed to include the RELDAYMON parameter in the ADDJOBSCDE command string if the aforementioned conditions are not met.

Because the RELDAYMON parameter specifies the value '1' as its default parameter, this value would be returned to the CPP if the user did not type something else in the command prompt for the RELDAYMON parameter. And passing the value '1' to the final ADDJOBSCDE parameter would potentially break the dependency between parameters stated above. To solve this problem, I therefore had to devise a solution that would let me detect when the RELDAYMON parameter had actually been specified.

This could happen either because the POP retrieved a specific value (or a list of values) from the job schedule entry to be copied, or because the user typed one or more values for this parameter in the command prompt. In the event this parameter would not apply to the existing job schedule entry, the QWCLSCDE API would return blanks for the relative day of month parameter, and in terms of command prompting, be equivalent to not entering the parameter.

Apparently at some point somebody in IBM's labs must have foreseen such a subtle requirement, because there's a command parameter attribute called Value to pass if unspecified (PASSVAL) that enables you to have the command definition pass a null pointer to the CPP if the parameter for which this attribute is defined is not specified. This way your CPP knows that nothing was specified for the parameter in question, as opposed to receiving the parameter's default value, not knowing whether this was the user's intention or not.

All that remains now is to check for the address of the RELDAYMON parameter in the CPP and build the ADDJOBSCDE command string in accordance with your findings. In the CPP, I named the RELDAYMON parameter PxRelDom and used a program defined variable named RelDomStr to hold the extracted parameter value, and make sure to check the %Addr() of the PxRelDom parameter:

```
/Free

  If  %Addr( PxRelDom )  *Null;

    For  Idx = 1  to PxRelDom.NbrElm;
      RelDomStr += %TrimR( PxRelDom.DayNbr(Idx)) + ' ';
    EndFor;
  EndIf;

/End-Free
```

If nothing was specified for the RELDAYMON parameter, the RelDomStr variable will contain its initial blank value following the above statement. When building the ADDJOBSCDE command

string, I therefore check the value of the RelDomStr variable and exclude the RELDAYMON keyword from the command string if the variable contains blanks, as demonstrated in the following code snippet:

```
/Free

  CmdStr = 'ADDJOBSCDE JOB(' + %Trim( PxJobNam )      + ') ' +
           'CMD(' + PxCmdStr                        + ') ' +
           'FRQ(' + %Trim( PxScdFrq )                + ') ' +
           'SCDDATE(' + ExtDatVal( PxScdDat )          + ') ' +
           'SCDDAY(' + %Trim( ScdDayStr )              + ') ' +
           'SCDTIME(' + ExtTimVal( PxScdTim )          + ') ' ;

  If RelDomStr > *Blanks;
    CmdStr += 'RELDAYMON(' + %Trim( RelDomStr )        + ') ' ;
  EndIf;

  CmdStr += 'SAVE(' + %Trim( PxSavEnt )                + ') ' +
           'OMITDATE(' + %Trim( OmtDatStr )            + ') ' +
           'RCYACN(' + %Trim( PxRcyAct )              + ') ' ;

  ...

/End-Free
```

When the POP has finished its job, the resulting CPYJOBSCDE command prompt in all details resembles the native Add Job Schedule Entry (ADDJOBSCDE) command prompt and lets you add, modify, and remove the current parameter values to reflect your actual requirements for the new job schedule entry. Press function key F10 to see all command parameters. The associated command help text panel group explains the command details as well as the command's individual parameters. Once you're done setting up the new job schedule entry, press Enter, and the schedule entry will be added to the Job Scheduler.

The native Work with Job Schedule Entries (WRKJOBSCDE) command supports these selection criteria: *Scheduled by user*, *Submit date*, *Job queue*, and *Library*. This is often a too-limited set of criteria if you want to quickly locate job schedule entries of particular interest. Creating my own version of the WRKJOBSCDE command based on the QWCLSCDE API lets me include any criterion that I find useful, so I decided to enhance this offering significantly, as you'll note seeing the Work with Job Schedule Entries 2 (WRKJOBSCD2) command prompt:

```
Work with Job Schedule Entries (WRKJOBSCD2)

Type choices, press Enter.

Job name . . . . . *ALL          Name, generic*,
*ALL
Output . . . . . *              *, *PRINT
```

Print format	*BASIC	*BASIC, *SCD, *JOB
Sequence	*JOB	*JOB, *DATETIME,
*JOBQ		
Status	*ALL	*ALL, *HLD, *SAV,
*SCD		
Frequency	*ALL	*ALL, *ONCE,
*WEEKLY...		
Scheduled by user	*ALL	Name, *ALL
Submit user profile	*ALL	Name, *ALL, *JOBQ
Job queue	*ALL	Name, *ALL, *JOBQ
Library		Name
Job description	*ALL	Name, *ALL, *USRPRF
Library		Name
Command substring	*ALL	
Submit time period:		
Start time and date:		
Beginning time	*ALL	Time, *ALL, *AVAIL
Beginning date		Date, *CURRENT,
*BEGIN		
End time and date:		
Ending time		Time, *AVAIL
Ending date		Date, *CURRENT, *END

In addition to the criteria supported by the native version, the WRKJOBSCD2 command also supports the following:

- Entry status
- Frequency
- Submit user profile
- Job description and library
- Command substring
- Submit time period

In the past, I've often printed all job schedule entries in full format to locate the job schedule entries referring to a specific program by scanning the resulting spooled file output. But using the Command substring selection criterion, you simply specify the program name, and then any job schedule entry containing the program name anywhere in its command string will be included in the resultant job schedule entry list.

Please refer to the command help text for more information on the WRKJOBSCD2 command in general and the parameters in particular. Here's an example of how the WKRJOBSCD2 command's job schedule entry list panel looks:

Work with Job Schedule Entries						
WYNDHAMW						16-02-11
10:09:36						
Type options, press Enter.						
1=Copy 2=Change 3=Hold 4=Remove 5=Display details 6=Release 7=Print 8=Work with last submission 10=Submit immediately						
-----Schedule-----						
Next						Recovery
Opt	Job	Status	Date	Time	Frequency	Action
Submit						
	QS9AUTOPTF	SCD	*FRI	15:00:00	*WEEKLY	*SBMRLS
18-02-11						
	QS9AUTOTST	SCD	*FRI	21:30:00	*WEEKLY	*SBMRLS
18-02-11						
	QS9SACOL	SCD	*ALL	10:08:00	*WEEKLY	*NOSBM
17-02-11						
	QSECIDL1	SCD	*ALL	01:00:00	*WEEKLY	*SBMRLS
17-02-11						
	QSJERRRPT	SAV	15-12-10	05:26:27	*ONCE	*SBMRLS
Bottom						
Parameters or command						
===>						
F3=Exit F4=Prompt F5=Refresh F6=Add F11=Display job						
queue data						
F12=Cancel F22=Display entire field F24=More keys						

I've added an extra list view to include the partial command string and the job user profile. Function key F22 combined with the cursor position shows the full job schedule entry text or the command string in a window. I've added the new Copy Job Schedule Entry (CPYJOBSCDE) command as list option 1 and also included a print option 7, which will print a single list entry's full details. Again I've included cursor-sensitive help text to explain the list panel sections, function keys, and list options.

Further, the WRKJOBSCD2 print formats supported when specifying OUTPUT(*PRINT) have been changed, compared to the original command. I've skipped the detail *FULL format, which will now instead be available for the new Display Job Schedule Entry (DSPJOBSCDE) command, letting you print full details about a single job schedule entry. The *BASIC print format producing a single line for each list entry is still supported. New print detail special values *SCD and *JOB have been added. Both will produce a single line for each printed list entry. The special value *SCD includes all schedule-related entry (run when) information, and special value *JOB likewise includes all job-related (run how) entry information.

Next item on the agenda, the DSPJOBSCDE command, is another new schedule entry command. The native WRKJOBSCDE command offers a job schedule entry display option, but there's no command interface to this functionality. The DSPJOBSCDE command lets you quickly display a known job schedule entry without having to list and locate the entry first. To display schedule entry details for the ACCUPD job schedule entry, simply type the following command:

```
DSPJOBSCDE ACCUPD
```

If more entries with the same name exist and you want to see entry number 000458, you have to qualify the entry name with this entry number:

```
DSPJOBSCDE ACCUPD 000458
```

If more entries with the same name exist and you do not specify an entry number, you'll get an error message indicating the duplicate name issue. Here's the DSPJOBSCDE command prompt in its entirety:

Display Job Schedule Entry (DSPJOBSCDE)

Type choices, press Enter.

Job name	Name
Entry number *ONLY	000001-999999,
*ONLY	
Output *	*, *PRINT

The Display Job Schedule Entry display shown when issuing the DSPJOBSCDE command is similar to the one associated with the WRKJOBSCDE command's list option 5=Display. In addition to the

function keys available on the latter version, I've included function key F8, which invokes the WRKJOBSCD2 command for the specified job name, and function key F13, which runs the Change Job Schedule Entry (CHGJOBSCDE) command for the displayed job schedule entry. Help text is also included for the DSPJOBSCDE command and its display panel.

The final job schedule entry command I present today is the Submit Job Schedule Entry (SBMJOBSCDE) command. The equivalent option 10=Submit immediately from the native WRKJOBSCDE list panel in my opinion suffers from the flaw that the resulting SBMJOB command inherits a lot of submit job attributes from the current job, as opposed to the attributes used when the job scheduler submits the job. This means that if you submit at job schedule entry immediately, it will potentially run with job attributes and libraries different from the ones applying when job the job runs scheduled.

You have the option of changing the *CURRENT job attribute values once the SBMJOB command is prompted prior to executing the command, but all in all it becomes a bit more complicated to immediately run a job schedule entry under the same circumstances as the same job schedule entry would run when scheduled. It is important that you consider this difference in behavior when you use the SBMJOBSCDE or list option 10 from the WRKJOBSCD2 command's list panel, since the outcome obviously might differ from what you'd expect when using the equivalent function from the WRKJOBSCDE command's list panel. Anyway, the SBMJOBSCDE command prompt has the following appearance:

Submit Job Schedule Entry (SBMJOBSCDE)

Type choices, press Enter.

Job name Name

Entry number *ONLY 000001-999999,
*ONLY

Once you specify the job name and if applicable the entry number and press Enter, the SBMJOB command prompt is presented. All job attributes that cannot be taken from the job schedule entry itself will have the special value *JOBID, pointing to the corresponding value stored in the specified job description. Submit job attributes that do not support a *JOBID option will instead use the special value *USRPRF, pointing to the corresponding value stored in the specified user profile or, alternatively, the special value *SYSVAL, pointing to the corresponding system value.

You of course then have the option of modifying any of the submit job attributes as you see fit. When you're done, press Enter to actually submit the job. Be sure to read all the information as well as the restrictions applying to the SBMJOBSCDE command in the accompanying help text.

This APIs by Example includes the following sources:

```
CBX223  -- RPGLE  -- Display Job Schedule Entry - CPP
CBX223H -- PNLGRP -- Display Job Schedule Entry - Help
```

```

CBX223P -- PNLGRP -- Display Job Schedule Entry - Panel Group
CBX223X -- CMD     -- Display Job Schedule Entry

CBX224  -- RPGLE  -- Copy Job Schedule Entry - CPP
CBX224H -- PNLGRP -- Copy Job Schedule Entry - Help
CBX224O -- RPGLE  -- Copy Job Schedule Entry - POP
CBX224X -- CMD     -- Copy Job Schedule Entry

CBX225  -- RPGLE  -- Submit Job Schedule Entry - CPP
CBX225H -- PNLGRP -- Submit Job Schedule Entry - Help
CBX225X -- CMD     -- Submit Job Schedule Entry

CBX226  -- RPGLE  -- Work with Job Schedule Entries 2 - CCP
CBX226E -- RPGLE  -- Work with Job Schedule Entries 2 - UIM Exit Pgm
CBX226H -- PNLGRP -- Work with Job Schedule Entries 2 - Help
CBX226P -- PNLGRP -- Work with Job Schedule Entries 2 - Panel Group

CBX223M -- CLP     -- Display Job Schedule Entry - Build command
CBX224M -- CLP     -- Copy Job Schedule Entry - Build command
CBX225M -- CLP     -- Submit Job Schedule Entry - Build command
CBX226M -- CLP     -- Work with Job Schedule Entries 2 - Build command

```

To create all these objects, compile and run the CBX223M, CBX224M, CBX225M, and CBX226M programs following the instructions in the source headers. You'll also find compilation instructions in the respective source headers.

Thanks to Peter Kemp of Australia for his suggestion to cover the QWCLSCDE API as well as his assistance in testing the commands evolving from this effort. Peter also provided valuable input to the design of the new commands and greatly inspired the development process in general.

Should you be interested in yet another new job schedule entry command called List Job Schedule Entries (LSTJOBSCDE), which writes all selected job schedule entries to an output file, watch for an upcoming issue of the *System iNetwork Systems Management Newsletter*, which will present the LSTJOBSCDE command in the not-too-distant future. You'll [find the newsletter issues at SystemiNetwork.com/sism](#), and you can [subscribe to this and other free System iNetwork newsletters at SystemiNetwork.com/newsletters](#).

IBM Documentation:

[Add Job Schedule Entry \(ADDJOBSCDE\) CL command](#)

[Examples: job schedule entry](#)

[Operations for Job Scheduler Object \(*JOBSCD\)](#)

[Example: Changing a job schedule entry](#)

[Advanced Job Scheduler](#)

[Submit Job \(SBMJOB\) CL command](#)

Related articles:

[APIs by Example: Retrieve Subsystem Entries API \(User Space List API\)](#)

[APIs by Example: New Data Queue Attributes and New Data Queue Command \(POP\)](#)

This article demonstrates the following API:

[List Job Schedule Entries \(QWCLSCDE\) API](#)

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

Source URL: <http://iprodeveloper.com/rpg-programming/apis-example-one-job-schedule-entry-api-and-four-new-job-schedule-entry-commands>