

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

## APIs by Example: Use a Work Management API to List Server Jobs

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

[Carsten Flensburg](#)

Carsten Flensburg

Thu, 02/26/2009 (All day)

In the preceding installment of APIs by Example, I discussed the concept of user-defined servers and the considerations and techniques involved in creating server jobs. One of the requirements that user-defined server jobs need to address is registration of the server job, using the Change Job (QWTCHGJB) API, so that the operating system can recognize it as a server job. Following this registration, you can use, for example, System i Navigator to look up and work with server jobs, both user-defined and system-supplied server jobs.

There's also a Work Management API that can locate and return information about user-defined and system server jobs. The Open List of Jobs (QGYOLJOB) API has parameters that let you specify job-selection criteria, including the job server type attribute. Using this selection criterion, you can limit the returned list of jobs to server jobs. You can specify any number of server types to return, or you can specify the special value \*ALL if you want a list of all server jobs on your system. This article demonstrates using QGYOLJOB to provide to provide a Work with Server Jobs (WRKSVRJOB) command.

There's a minor challenge involved in specifying the QGYOLJOB server type job selection parameter--there's no API available to return a list of registered servers and their server type. Thorough investigation reveals that practically the only resource available on a i that provides server information is the QATOCSTART file that contains the TCP/IP servers, Network File System (NFS) servers, and host servers. This file also provides the information required to start and end these servers. Although QATOCSTART contains the server special value (\*SMTP, \*HTTP, etc.) it does not contain the server type.

While there is no immediate resolution to the problem at hand, server type information is actually documented in the Server table available in the Information Center--a page called "i5/OS TCP/IP troubleshooting: Server table" (at the end of this article, I provide links to the 5.4 and 6.1 server tables) provides a wealth of information about servers, server jobs, job descriptions, subsystems, and server type. So I've taken the liberty of creating my own server table that provides the mapping between server and server type for the WRKSVRJOB command, based on a combination of the information found in the aforementioned server table and that found in the QATOCSTART file.

This approach requires you to update the server table whenever user-defined servers are added--and if new system servers are introduced as part of a system upgrade. But it is as good as it gets under the given circumstances. Hopefully IBM will address this shortcoming at some point in the future and provide an API to list the server registry. Due to some inconsistencies in the mentioned documentation I've excluded a few servers from my homegrown server table, but as you will see it's quite easy to enhance the server list, should you choose to include some of the missing servers at a later point. My selective approach will however not prevent servers excluded from my server table to

surface in the WRKSVRJOB's server list panel if you specify SERVER(\*ALL) on the WRKSVRJOB command, so use this option to get a complete picture of the server job activities on your system.

Adding entries to the server table is simply a matter of using a file editor as for example DFU and specify the values defining the new server. The following fields are included in the CBX200F server table:

Display File Field Description							
WYNDHAMW							
21-02-09							
17:59:19							
File . . . . . :		CBX200F		Record length . . :		66	
Library . . . . :		QGPL		Field count . . :		4	
Record format . . :		CBX200R					
File type . . . . :		PF		Include field . .			
Access path . . . :		*ARRIVAL		Include text . . .			
Field	Data type	Buffer	Length	Dig	Dec	Key	Text
SVRSPV	Char	1	10				Server
special value							
SVRTYP	Char	11	30				Server
type							
SVRKEY	Char	41	16				Server
table key							
SVRJOB	Char	57	10				Server job
Bottom							
F3=Exit		F7=Position to		F8=Work with PDM		F9=Display PF access	
paths							
F10=Display data base relations		F11=Display column headings					
F24=More keys							

The *Server special value* (SVRSPV) field uniquely identifies the server (\*SMTP, \*HTTP, \*DATABASE, etc.) and is used to establish the relation to the *Server type* (SVRTYP) used as selection criteria for the QGYOLJOB API. The *Server key* (SVRKEY) field provides the foreign key to the system QATOCSTART table. Finally the *Server job* (SVRJOB) field is an optional feature that would allow to further qualify server jobs by job name--but it is currently not used. To give you an

impression of how the pieces fit together, here's an excerpt from the CBX200F table as it looks when running Query/400 against it:

SVRSPV	SVRTYP	SVRKEY	SVRJOB
*ALL	*ALL	*NONE	*ALL
*ASFTOMCAT	QIBM_AFSTOMCAT_*	*ASFTOMCAT	*ALL
*BOOTP	QIBM_BOOTP	*BOOTP	*ALL
*CIMOM	QIBM_CIMOM	*CIMOM	*ALL
*DBG	QIBM_DEBUG_SERVER	*DBG	*ALL
*DLFM	QIBM_DLFM	*DLFM	*ALL
*DHCP	QIBM_BOOTP_DHCP_RA	*DHCP	*ALL
*DHCP	QIBM_DHCP	*DHCP	*ALL
*DIRSRV	QIBM_DIRSRV_SERVER	*DIRSRV	*ALL
...			
*DTAQ	QIBM_OS400_QZBS_SVR_DTAQ	HOSTDTAQ	*ALL
*FILE	QIBM_OS400_QZBS_SVR_FILE	HOSTFILE	*ALL
*NETPRT	QIBM_OS400_QZBS_SVR_NETPRT	HOSTNETPRT	*ALL
*RMTCMD	QIBM_OS400_QZBS_SVR_RMTCMD	HOSTRMTCMD	*ALL
*SIGNON	QIBM_OS400_QZBS_SVR_SIGNON	HOSTSIGNON	*ALL
*SVRMAP	QIBM_OS400_QZBS_SVR_SVRMAP	HOSTSVRMAP	*ALL
*NFSRPC	QIBM_NFS_RPCD	NFSRPC	*ALL
*NFSBIO	QIBM_NFS_BIOD	NFSBIO	*ALL
*NFSSVR	QIBM_NFS_NFSD	NFSSVR	*ALL

Hopefully you now also have an idea about how to add entries to the table. And now over to the WRKSVRJOB command which displays the following input parameter list when fully prompted:

Work with Server Jobs (WRKSVRJOB)

```

Type choices, press Enter.

Server . . . . . *ALL          *ALL, *ASFTOMCAT,
*BOOTP...
Job name . . . . . *ALL          Name, generic*,
*ALL...
User name . . . . . *ALL          Name, generic*,
*ALL...
Job status . . . . . *ACTIVE      *ACTIVE, *JOBQ,
*OUTQ...
Current user . . . . . *NOCHK      Name, *NOCHK

```

By default all currently active server jobs are displayed and the list is sorted by server type, job name and job number in ascending order. You have the option to narrow the list by specifying a job name, job user or current user as selection criteria and also to expand the list by including other job statuses. The Server special values are provided by means of a choice program. Whenever the command is prompted or function key F4 is pressed in the Server parameter field the program specified as choice program for the Server parameter in the command definition is called.

The choice program retrieves the server table entries and formats the choice text as well as the permissible values list for the prompt display and F4 server list, respectively. The list of server table entries are also retrieved in the WRKSVRJOB command's validity checking program (VCP) to ensure that only valid values are allowed as input to the Server parameter. This way new values entered into the server table are immediately recognized and allowed on the WRKSVRJOB command. The only drawback to this approach is that the help text panel group will need a manual update to reflect the newly added server table entry.

The online help text panel group will explain all the details. Running the above command displays the following list on my system:

```

                                Work with Server Jobs

WYNDHAMW                                21-02-09

17:33:06
Server:  *ALL          Job:  *ALL          User:  *ALL

Status:  *ACTIVE

Type options, press Enter.

    2=Change    3=Hold    4=End    5=Work with    6=Release    8=Spooled
files
    10=Job log   12=Work with user jobs

```

Current					Function/		
Opt	Job	User	Job date	Type	---Status---		
Completion							
	CRTPFRDTA	QSYS	22-02-09	BCH	ACTIVE	DEQW	CMD-
	CRTPFRDTA						
	QGLDPUBA	QDIRSRV	18-02-09	ASJ	ACTIVE	SIGW	PGM-
	QGLDPUBA						
	QGLDPUBE	QDIRSRV	18-02-09	ASJ	ACTIVE	DEQW	PGM-
	QGLDPUBE						
	QDIRSRV	QDIRSRV	18-02-09	BCH	ACTIVE	SIGW	PGM-
	QGLDSVR						
	QTFTP00114	QTCP	18-02-09	BCH	ACTIVE	DEQW	
	QTFTP00169	QTCP	18-02-09	BCH	ACTIVE	DEQW	
	QTFTP00175	QTCP	18-02-09	BCH	ACTIVE	DEQW	
More...							
Parameters or command							
===>							
F3=Exit		F4=Prompt	F5=Refresh	F6=Server commands			
F9=Retrieve							
F11=View 2		F12=Cancel	F21=Print list	F22=Work with active jobs			

The list panel offers to further views including such details as job user, job timestamps, subsystem and server type. Again there's a comprehensive online help text to document the different parts of the list panel and columns. As for the User Interface Manager (UIM) programming involved in the WRKSVRJOB utility it's beyond the scope of this article, but I've included links below to a number of previously published articles discussing in detail the nuts and bolts of writing a UIM list panel and the RPG/IV programming required to make it work.

### This APIs by Example includes the following sources:

```

CBX200  -- RPGLE  -- Work with Server Jobs - CCP
CBX200E -- RPGLE  -- Work with Server Jobs - UIM General Exit Program
CBX200L -- RPGLE  -- Work with Server Jobs - UIM List Exit Program
CBX200C -- RPGLE  -- Work with Server Jobs - Choice Program
CBX200V -- RPGLE  -- Work with Server Jobs - VCP

CBX200B -- SRVSRV -- Work with Server Jobs - Binder source
CBX200S -- RPGLE  -- Work with Server Jobs - Services

CBX200D -- DDL    -- Work with Server Jobs - SQL DDL Script
CBX200H -- PNLGRP -- Work with Server Jobs - Help
CBX200P -- PNLGRP -- Work with Server Jobs - Panel Group
CBX200X -- CMD    -- Work with Server Jobs

```

```
CBX200M  -- CLP      -- Work with Server Jobs - Build command
```

To create all these objects, compile and run CBX200M, following the instructions in the source header. As always, you'll also find compilation instructions in the respective source headers.

Please note the CBX200M uses the Run SQL Statement (RUNSQLSTM) to create the CBX200F table and CBX200L1 index as well as to populate the table. If you do not have the DB2 Query Manager and SQL Development Kit for iSeries installed on your system (product 5722ST1 for release 5.4), you'll need to remove the RUNSQLSTM command statement from the CBX200M CL program and split the CBX200D SQL DDL statements into three separate source members and run the SQL statements using the Create QM Query (CRTQMQR) and Start QM Query (STRQMQR) commands:

```
CRTQMQR  QMQR (CBX200D1)
          SRCFILE (QQMQRYSRC)
          SRCMBR (CBX200D1)
          SRTSEQ (*JOB RUN)
          LANGID (*JOB RUN)
          REPLACE (*YES)
```

followed by:

```
STRQMQR  QMQR (CBX200D1)
          NAMING (*SYS)
```

Run the above command sequence for each of the three SQL DDL source members and before you run the altered version of CBX200M. Note that the created objects will end up in the current library of the job running the STRQMQR command, so remember to set the current library appropriately before executing the aforementioned command. I've included the alternative three SQL DDL statements in source members CBX200D1, CBX200D2 and CBX200D3 for your convenience.

### IBM documentation:

System i Server table 5.4:

<http://publib.boulder.ibm.com/infocenter/iseriess/v5r4/index.jsp?topic=/rzaku/rzakuservertable.htm>

System i Server table 6.1:

<http://publib.boulder.ibm.com/infocenter/systems/scope/i5os/topic/rzaku/rzakuservertable.htm>

### Previously published related articles:

APIs by Example: Use Security and Job APIs with User Defined Servers:

<http://systeminetwork.com/article/apis-example-use-security-and-job-apis-user-defined-servers>

APIs by Example: User Interface Manager APIs:

<http://systeminetwork.com/article/apis-example-user-interface-manager-apis>

APIs by Example: UIM & Work Management APIs, Part 1:

<http://systeminetwork.com/article/apis-example-uim-work-management-apis-part-1>

APIs by Example: UIM & Work Management APIs, Part 2:

<http://systeminetwork.com/article/apis-example-uim-work-management-apis-part-2>

APIs by Example: UIM & Work Management APIs, Part 3:

<http://systeminetwork.com/article/apis-example-uim-work-management-apis-part-3>

**This article demonstrates the following Work Management API:**

Open List of Jobs (QGYOLJOB) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r4/topic/apis/qgyoljob.htm>

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

**Source URL:** <http://iprodeveloper.com/rpg-programming/apis-example-use-work-management-api-list-server-jobs>