


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

APIs by Example: Server Support APIs Providing NetServer Management Facilities

[System iNetwork Programming Tips Newsletter](#)

[Carsten Flensburg](#)

Carsten Flensburg

Thu, 08/25/2011 (All day)

The IBM i Support for Windows Network Neighborhood, also known as IBM i NetServer, lets you access IBM i file and print resources directly from your PC through file and printer shares. This facility enables you to map IBM i file systems and directories to network drives on your PC as well as direct PC printer output to IBM i-configured printers. Information about your system's NetServer configuration can be retrieved by using the Navigator for i interface, which also supports the associated NetServer management functions.

If you're an API programmer, however, you also have the option of creating your own command interfaces to the NetServer configuration information and management functions. As part of the Server Support APIs, IBM provides a set of APIs offering a programming interface to IBM i NetServer functions. In today's APIs by Example, I discuss some of these APIs and demonstrate their use in the context of four new CL commands that let you end and start the NetServer, as well as display the NetServer configuration information and NetServer connection statistics.

The End Server (QZLSEENDS) API and the Start Server (QZLSSTRS) API both employ very simple parameter lists. In addition to the API error data structure that both APIs support, the only other parameter is the *Reset server* parameter offered by the QZLSSTRS API. The End NetServer (ENDNETSVR) and Start NetServer (STRNETSVR) commands therefore specify the APIs directly as CPPs. As demonstrated in earlier APIs by Example articles, a command definition can submit an API error data structure as a single integer set to the value zero, in turn causing the API to signal any encountered errors in the form of an exception message sent directly to the API caller.

Because this is exactly the behavior that a separate CPP would be expected to perform, I can rely on the API directly instead of introducing a middle layer between command and API. Here's how the API error data structure is defined in the command definition source:

Parm	ERRC0100	*Char	4	+
	Constant (x'00000000')			

The above command constant parameter specification causes a single integer of the value zero to be passed to the API being called, prompting the API to communicate any error conditions encountered in the form of diagnostic and escape messages. As mentioned, the Start Server API also lets you specify whether the NetServer should be reset before being started, and that's simply mapped directly to the command definition as the one-byte character parameter the QZLSSTRS API expects, and restricted to the two supported values, '0' and '1', respectively:

Parm	RESET	*Char	1	+
	Dft (*NO)			
				+

```

Rstd( *YES )      +
SpcVal(( *NO      '0' )      +
        ( *YES    '1' ))      +
Expr( *YES )      +
Prompt( 'Reset server' )

```

To make the two parameter values immediately comprehensible to the user, they are presented on the command prompt as special values *NO and *YES, but the command processor passes the replacement value specified for each special value to the CPP. Specifying, for example, RESET(*YES) causes the value '1' to be passed to the CPP. Anyway, here's the resulting STRNETSVR command prompt:

```

                                Start NetServer (STRNETSVR)

Type choices, press Enter.

Reset server . . . . . *NO      *NO, *YES

```

The NetServer configuration and statistics information is, along with a lot of NetServer session, share, and user information, available through both the List Server Information (QZLSLSTI) API and the Open List of Server Information (QZLSOLST) API. Both APIs support the exact same return data formats as well as an *Information qualifier* parameter that lets you further narrow the scope of the returned information. The QZLSLSTI API returns information through a user space, whereas the QZLSOLST API returns information through the API call itself, plus subsequent calls to the Get List Entries (QGYGTLE) API, if more data is available than returned in the initial call.

Both the user space List API type and the Open List API type have been explained in much detail in earlier articles, to which I've included links at the end of this article. For the purpose at hand, I needed to retrieve NetServer configuration and NetServer statistics information. Both API return formats offering this information provide involve a simple, fixed-format return string. So I chose the QZLSOLST Open List API, thereby avoiding the need to include a user space in the equation. Let's have a brief look at the QZLSOLST API parameter list:

Required parameter group:			
1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	List Information	Output	Char(80)
4	Format name	Input	Char(8)
5	Information qualifier	Input	Char(15)
6	Error code	I/O	Char(*)
Optional parameter 1:			
7	Session user	Input	Char(10)

Optional parameter 2:

8	Expanded workstation name	Input	Char (*)

The *Receiver variable* is where the API returns the requested information—or as much of it as the size of the receiver variable defined by the second API parameter permits. The *List Information* parameter is used by an Open List API to communicate back to the API caller about the number of list records returned, the total number of list records available, the length of each list record, and the state of the returned information and a couple of other things. The Open List API caller uses this information to safely and correctly process the data returned.

The *Format name* parameter defines the type of information the API returns. For the QZLSOLST API (as well as the QZLSLSTI API), the following return formats are offered:

- ZLSLO100 - Share information
- ZLSLO101 - Same as ZLSLO100, plus extension information
- ZLSLO200 - Configuration information
- ZLSLO201 - Same as ZLSLO200, plus additional fields
- ZLSLO300 - Session information
- ZLSLO400 - Statistical information
- ZLSLO600 - Session connection information
- ZLSLO700 - Share connection information
- ZLSLO800 - Share type information
- ZLSLO900 - Disabled user profiles

I mentioned the *Information qualifier* parameter earlier, and it serves different purposes for the various return formats. For example, when specifying the ZLSLO400 statistical information format, you can use the Information qualifier to specify that the NetServer accumulated statistical information should be reset by submitting the special value *RESET. Another possible purpose of the Information qualifier is to point to one of the API's optional parameters number 7 and 8, to include only information applying to the session, user, or workstation specified here.

Anyway, the Display NetServer Attributes (DSPNETATR) command is based on the QZLSOLST API return format ZLSLO201, providing extended NetServer configuration information. The DSPNETATR command has the following appearance when prompted:

Display NetServer Attributes (DSPNETATR)

Type choices, press Enter.

Output * *, *PRINT

The DSPNETATR command includes a help text panel group to document the details. Below is an example of the NetServer configuration information returned on my system:

Display NetServer Attributes	
WYNDHAMW	21-08-11
15:43:14	
Server	WYNDHAM
Domain	WYNDWORLD
Text 'description'	OS/400
Automatic start	*YES
Current status	RUNNING
Guest support	*YES
Guest user profile	GUEST
Coded character set id (CCSID)	*JOB
Allow system name	*NO
Server role	*NONE
Inactivity time-out	6000
Opportunistic lock time-out	30
Browsing interval	720000
Authentication method	*ENCPWDONLY
Message authentication	*NONE
LAN Manager authentication	*IGNORE
Allow WINS proxy	*NO
Primary WINS address	10.249.40.65
Secondary WINS address	

```
Scope identifier . . . . . :  
  
More...  
F3=Exit      F5=Refresh      F6=End NetServer      F9=Display  
statistics  
F11=Display pending values      F12=Cancel
```

Again, help text is provided to explain the display panel and its various parts. You can use function key F11 to include the configuration values pending in order to see what configuration parameters will take effect the next time the server is started. Function key F6 prompts the End NetServer command, provided that the server is currently running. Otherwise function key F6 prompts the Start NetServer command. Finally, function key F9 runs the Display NetServer Statistics (DSPNETSTC) command; I share more about this command shortly.

In addition to the information accessible through Server Support API calls, I use APIs to retrieve NetServer information not exposed directly by APIs. One such piece of information is the *Autostart server* configuration setting, defining whether the NetServer should be started automatically when the Start TCP/IP (STRTCP) command is run. This value is stored in the system table QATOCSTART in library QUSRSYS. Because this table's format might be changed by IBM at some point in the future, I decided to use SQL to retrieve the value of the mentioned table's AUTOSTART column.

And because the SQL CLI APIs, as opposed to RPG IV embedded SQL, do not require the *DB2 Query Manager and SQL Development Kit for IBM i* product to be licensed and installed, the SQL CLI APIs were chosen to perform the duty of reading the QATOCSTART table's *NETSVR record and extracting the NetServer autostart setting. I've included a link to an earlier APIs by Example article employing the SQL CLI APIs and pointing to more information on these APIs at the end of this article, in case you'd like to learn more about the SQL CLI topic.

Another NetServer detail that I was unable to obtain using the Server Support APIs was the current status of the NetServer, in terms of the server currently running or not. A quick research on the topic in IBM's online documentation revealed that this condition is defined by a specific job named QZLSSERVER running in subsystem QSERVER. If the QZLSSERVER job is not active, then NetServer is not active—and vice versa. Converting that criterion to code implies the use of the Open List of Jobs (QGYOLJOB) API, which supports both a job name and an active subsystem name as selection parameters. The latter is supported on the QGYOLJOB API selection format OLJS0200.

The active subsystem name is, however, valid only if you specify the QGYOLJOB API return format OLJB0300, specifically targeting active jobs. I mention this requirement because specifying an active subsystem name while using return formats OLJB0100 or OLJB0200 does not return an error. The active subsystem name specified is simply ignored, so you might end up assuming the API output reflects the specified subsystem name criterion, and not discover until later that it actually does not.

Now, on to the final piece I want to present today, the DSPNETSTC command, which is taking advantage of the QZLSOLST API return format ZLSLo400, delivering the NetServer accumulated statistics information. The DSPNETSTC command prompt is shown below:

Display NetServer Statistics (DSPNETSTC)

Type choices, press Enter.

Reset statistics *NO *NO, *YES

Output * *, *PRINT

The DSPNETSTC command offers you the option of resetting the NetServer accumulated statistics, causing all statistics registers to be set to zero. You can then use the function key F5 on the Display NetServer Statistics panel to update and incrementally accumulate the NetServer statistics:

Display NetServer Statistics

WYNDHAMW

21-08-11

15:46:01

Server started : 19-08-11 05:31:41

Statistics started : 20-08-11 21:07:08

Average response time : 4

Number of open files : 1846

Number of queued print jobs : 0

Number of session starts : 229

Number of disconnected sessions:

Automatically : 46

Normally : 181

Number of password violations : 3

Number of guest users : 314

```
Number of bytes sent . . . . . : 16022388
```

```
Number of bytes received . . . . . : 7104079
```

```
Bottom
```

```
F3=Exit    F5=Refresh    F12=Cancel    F13=Reset statistics
```

Function key F13 lets you reset the NetServer statistics directly from the above display panel. For all other details, please refer to the panel's help text. In an upcoming issue of APIs by Example, I continue the coverage of Server Support APIs and add to the collection of NetServer commands.

This APIs by Example includes the following sources:

```
CBX235    -- RPGLE    -- Display NetServer Attributes - CPP
CBX235E   -- RPGLE    -- Display NetServer Attributes - UIM General Exit
CBX235H   -- PNLGRP   -- Display NetServer Attributes - Help
CBX235P   -- PNLGRP   -- Display NetServer Attributes - Panel Group
CBX235X   -- CMD      -- Display NetServer Attributes

CBX2352H  -- PNLGRP   -- End NetServer - Help
CBX2352X  -- CMD      -- End NetServer
CBX2353H  -- PNLGRP   -- Start NetServer - Help
CBX2353X  -- CMD      -- Start NetServer

CBX236    -- RPGLE    -- Display NetServer Statistics - CPP
CBX236H   -- PNLGRP   -- Display NetServer Statistics - Help
CBX236M   -- CLP      -- Display NetServer Statistics - Build command
CBX236X   -- CMD      -- Display NetServer Statistics

CBX235M   -- CLP      -- Display NetServer Attributes - Build command
CBX236P   -- PNLGRP   -- Display NetServer Statistics - Panel Group
```

To create all these objects, compile and run the CBX235M and CBX236M programs, following the instructions in the source headers. You'll also find compilation instructions in the respective source headers.

For the DSPNETSVR CPP CBX235 to compile, you need to download and copy the SQLCLI_H member mentioned in earlier APIs by Examples articles to a QRPGLSRC source file in your job's library list. I've provided a link to a zip file containing the correct version of the SQLCLI_H copy member below.

The prerequisite SQLCLI_H copy member is available as part of the following download:

http://www.pentontech.com/IBMContent/Documents/article/56657_600_CliClob.zip

Related articles:

[APIs by Example: Directing API Output to Output Files Using the SQL CLI APIs](#)

[APIs by Example: Use a Work Management API to List Server Jobs](#)

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

[APIs at Work — with Jobs \(Open List APIs explained\)](#)

IBM documentation:

[Getting started with NetServer](#)

[Introduction to IBM iSeries NetServer Documents](#)

[Adding a Printer That Uses an iSeries NetServer Share](#)

[API error code parameter format](#)

This article demonstrates the following Server Support APIs:

[Open List of Server Information \(QZLSOLST\) API](#)

[Start Server \(QZLSSTRS\) API](#)

[End Server \(QZLSEENDS\) API](#)

[IBM i Support for Windows Network Neighborhood Server APIs](#)

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

Source URL: <http://iprodeveloper.com/rpg-programming/apis-example-server-support-apis-providing-netserver-management-facilities>