

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

APIs by Example: Server Support APIs Enable the Change of NetServer Information

[System iNetwork Programming Tips Newsletter](#)

[Carsten Flensburg](#)

Carsten Flensburg

Thu, 09/22/2011 (All day)

Last time, I presented the three Server Support APIs capable of Starting and Ending the IBM i NetServer as well as retrieving information about the IBM i NetServer ("[Server Support APIs Providing NetServer Management Facilities](#)," August 25, 2011, article ID 66153 at [SystemiNetwork.com](#)). Today's installment of APIs by Example focuses on the Server Support APIs available to change the NetServer configuration information on your IBM i. Whereas a single API is responsible for retrieving the NetServer configuration details, no less than three APIs are required to change the three different categories of NetServer information, into which this information was divided when the associated change APIs were designed.

The three APIs in question are the Change Server Guest (QZLSCHSG) API, which lets you change the user profile used by the IBM i NetServer when an unknown user requests access to shared resources on the system; the Change Server Name (QZLSCHSN) API, which can alter server name information by which the IBM i NetServer is known on the network; and the Change Server Information (QZLSCHSI) API, which provides access to updating the general configuration information for the IBM i NetServer. I've packaged all three APIs into the Change NetServer Attributes (CHGNETATR) CL command, offering the context of employing the mentioned APIs in this APIs by Example.

In addition to the CHGNETATR command, for completeness, I also wrote a Change TCP Server Autostart (CHGTCPSVRA) command, which provides an interface to update the autostart setting of any of the TCP servers available on the IBM i, including the IBM i NetServer, of course. I've added the function key F10 to the Display NetServer Attributes (DSPNETATR) command's display panel to provide a shortcut to the CHGTCPSVRA command. The DSPNETATR command was presented in the previous installment of this column, and I've included a link to that article below.

The sources that have been updated to support the CHGTCPSVRA command shortcut are available with the source download for this article. Just follow the instructions at the end of this article to add this new support to the Display NetServer Attributes display panel. The TCP server autostart information is stored in the system table QATOCSTART in library QUSRSYS. To access and update this information, I'm using the C Library Function Record I/O APIs. I've demonstrated these APIs on previous occasions in APIs by Example and also included links to these articles below. The record I/O APIs let you perform I/O operations against a file at the record level. The obligation then lies with the programmer to ensure that the constraints of the record layout in terms of the fields' data type, length, decimal precision, and so forth are observed and handled correctly when accessing and updating the field data. In the aforementioned examples discussing the C Library Function Record I/O APIs, a more generic approach is demonstrated in the code accompanying the articles.

For the purpose at hand, my main objective was to safeguard against IBM adding new fields to the QATOCSTART file at the end of the record format, which is the way IBM usually implements such

design changes. To provide for extra fields being added to the record format in the future, I explicitly define the length of the externally defined data structure, thereby well exceeding the current length of the external file's record format:

```

**-- I/O feedback structure:
D RIOFB          Ds          Based( pRIOFB )   Qualified
D  pKey          *
D  pSysParm      *
D  RcdRrn        10u 0
D  NbrBytRw      10i 0
D  BlkCnt        5i 0
D  BitFld        1a
D                20a
**-- Record buffer:

D QATOCSTART     E DS          65535          ExtName( QATOCSTART )
Qualified

```

This prevents a truncation error from occurring when reading the record buffer using the C record I/O API `Rreadk` (read a record by key), should the record format length increase at a later time. When updating the record buffer, I then use the actual number of bytes read returned from the `Rreadk` API to ensure that the exact record length is passed to the `Rupdate` (update a record) API, as demonstrated by the code snippet below:

```

/Free

pRIOFB = Rreadk( pRFILE
                : %Addr( QATOCSTART )
                : %Size( QATOCSTART )
                : KEY_EQ
                : %Addr( PxServer ) + 2
                : %Len( PxServer )
                );

If  RIOFB.NbrBytRw > *Zero;
  QATOCSTART.AUTOSTART = PxAutStr;

pRIOFB = Rupdate( pRFILE
                 : %Addr( QATOCSTART )
                 : RIOFB.NbrBytRw
                 );

EndIf;

/End-Free

```

Another approach providing record format update independence would be using SQL, either HLL embedded SQL or the SQL CLI APIs. As I discussed last time, the SQL CLI APIs do not require any products to be licensed and installed, as opposed to embedded SQL, so for most purposes I use SQL CLI when I need my code to run on machines on which I cannot rely on the *DB2 Query Manager and SQL Development Kit for IBM i* product being installed. I have included an example of how to write a TCP server autostart attribute get- and set-function using the SQL CLI APIs, just in case you'd

prefer that solution over the C Library Function Record I/O APIs. For more information on the SQL CLI APIs, please refer to the article "APIs by Example: Directing API Output to Output Files Using the SQL CLI APIs," linked below.

Now on to the core of the matter, the Change TCP Server Autostart (CHGTCPSVRA) command, which has the following appearance when fully prompted:

```

Change TCP Server Autostart (CHGTCPSVRA)

Type choices, press Enter.

Server . . . . . *ASFTOMCAT,
*BOOTP, *CIMOM...
Autostart . . . . . *SAME *SAME, *NO, *YES
    
```

The CHGTCPSVRA command supports a number of facilities assisting in specifying the command input. The command's choice program (CBX238C) provides the list of choices appearing to the right of the *Server* prompt, as well as the list of all available server special values displayed when function key F4 is pressed with the cursor placed in the Server input field. The CHGTCPSVRA command's prompt override program (CBX238O) retrieves the autostart value currently in effect for the TCP server selected, when the server special value identifying the server is entered and you press Enter.

Finally, the command validity checking program (CBX238V) ensures that only a valid TCP server special value was entered, before control is passed to the command processing program CBX238. Additionally a command help text panel group explaining both the command and also the command's parameters in detail is included to take care of the questions that may remain.

As for the three Server Support APIs involved in the CHGNETATR command I present today, they all expose quite simple parameter interfaces. The Change Server Guest (QZLSCHSG) API in fact requires only two parameters:

Required parameter group:			
1	Guest user profile	Input	Char(10)
2	Error code	I/O	Char(*)

You specify the name of the user profile that the IBM i NetServer should assign to unknown users requesting access to shared resources, or alternatively specify a blank parameter value to indicate that unknown users should not be allowed access to shared resources. The standard API error code parameter is the only other parameter required. Note that both *IOSYSCFG and *SECADM special authority is needed to call the QZLSCHSG API successfully, and that changes will not be reflected until the NetServer is restarted.

The Change Server Name (QZLSCHSN) API's parameter count exceeds that of the QZLSCHSG API, but is still relatively simple:

```


```

Required parameter group:

1	Server name	Input	Char (15)
2	Domain name	Input	Char (15)
3	Text description	Input	Char (50)
4	Error code	I/O	Char (*)

Optional parameter:

5	Allow system name	Input	Char (1)
---	-------------------	-------	----------

The first three parameters define the server name by which the system is known in the Windows Network Neighborhood, the domain name or workgroup in which the server appears, and the textual description of the system that appears on the network, respectively. All these parameters must be submitted, even if you want to change only one or two of them, consequently forcing you to retrieve the current values for those parameters that require no change. This requirement is eliminated for the fifth and optional parameter, *Allow system name*, because if no change is required you simply leave out this parameter on the API call, thereby avoiding an update being performed by the QZLSCHSN API.

Finally, the Change Server Information (QZLSCHSI) API on one hand has only four parameters yet enables you to change up to 15 IBM i NetServer configuration parameters in a single call. This capability is owed to the QZLSCHSI API's first parameter, a data structure comprising the mentioned 15 configuration parameters. In addition to the NetServer configuration parameters, the API is also capable of enabling a list of NetServer users that have previously been disabled due to security violations, as for example specifying invalid passwords. Here's the parameter list in its entirety:

Required parameter group:

1	Request variable	Input	Char (*)
2	Length of request variable	Input	Binary (4)
3	Format	Input	Char (80)
4	Error code	I/O	Char (*)

The second parameter specifies the length of the request variable submitted as the first API parameter. The Format parameter, specified as number three, defines whether the first parameter contains NetServer configuration parameters that should be changed, or a list of NetServer users that should be enabled. The former is named ZLSSo100, the latter ZLSSo200. As always, I've included links at the end of the article to the relevant API documentation, allowing you to go into more detail, if needed.

The three parameters are all employed by the Change NetServer Attributes (CHGNETATR) CPP to perform the updates entered in the command prompt. Again a prompt override program (POP) is included to retrieve the NetServer attributes currently applying and display these values when the CHGNETATR command is prompted. Any currently pending changes are ignored. The prompt override technique used was discussed in an earlier article, ["APIs by Example: New Data Queue Attributes and New Data Queue Command."](#) to which a link is also provided below.

Using the prompt override technique described in the article, I'm able to distinguish between parameters that have been altered and parameters left unchanged. This is useful especially due to the three APIs involved in performing a full update, allowing me to call only the relevant API(s) if a partial update was requested. To show you the end result, I ran the CHGNETATR command on my system, and here's what the command prompt eventually displayed looked like:

Change NetServer Attributes (CHGNETATR)

Type choices, press Enter.

Server name WYNDHAMW

Domain WYNDWORLD

Text 'description' 'Os/400'

Allow system name *NO *NO, *YES, *SAME

Guest user profile GUEST Name, *NONE, *SAME

Coded character set identifier *JOB 1-65535, *JOB,
*SAME

Inactivity time-out 6000 Number, *NOMAX,
*SAME, *DFT

Opportunistic lock time-out . . . 30 Number, *DISABLED,
*SAME...

Browsing interval 720000 1-720000, *NONE,
*MAX...

WINS primary address '10.249.78.14'

WINS secondary address *NONE

Scope identifier *NONE

Allow WINS proxy *NO *NO, *YES, *SAME,
*DFT

Server role *NONE *NONE, *SERVER,
*SAME, *DFT

Authentication method *ENCPWDONLY *ENCPWDONLY,
*NETAUTONLY...

Authentication method *NONE *NONE,
*NEGOTIATED...

Minimum message severity 0 0-99, *NONE, *SAME

LAN manager authentication *IGNORE *IGNORE,
*FALLBACK, *SAME...

Note that any change performed using the CHGNETATR command will not take effect until the next time the IBM i NetServer is started. You can see pending changes by using the Display NetServer Attributes (DSPNETATR) command. The CHGNETATR command and all its parameters are

documented in the associated online help text. Running the CHGNETATR command requires *IOSYSCFG special authority, and changing the Guest user profile additionally requires *SECADM special authority.

In the next installment of APIs by Example, I complete the Server Support API tour and introduce you to the Work with NetServer (WRKNETSVR) command as well as a set of NetServer share and NetServer session management commands based on equivalent Server Support APIs.

This APIs by Example includes the following sources:

```
CBX235H  -- PNLGRP -- Display NetServer Attributes - Help
CBX235P  -- PNLGRP -- Display NetServer Attributes - Panel Group
CBX235C  -- RPGLE  -- Display NetServer Attributes - UIM Cond Program

CBX237   -- RPGLE  -- Change NetServer Attributes - CPP
CBX237H  -- PNLGRP -- Change NetServer Attributes - Help
CBX237H2 -- PNLGRP -- Change NetServer Attributes - Help
CBX237O  -- RPGLE  -- Change NetServer Attributes - POP
CBX237X  -- CMD    -- Change NetServer Attributes

CBX238   -- RPGLE  -- Change TCP Server Autostart - CPP
CBX238C  -- RPGLE  -- Change TCP Server Autostart - Choice program
CBX238H  -- PNLGRP -- Change TCP Server Autostart - Help
CBX238O  -- RPGLE  -- Change TCP Server Autostart - POP
CBX238V  -- RPGLE  -- Change TCP Server Autostart - VCP
CBX238X  -- CMD    -- Change TCP Server Autostart

CBX237M  -- CLP     -- Change NetServer Attributes - Build command
CBX238M  -- CLP     -- Change TCP Server Autostart - Build command

CBX238S  -- RPGLE  -- SQL CLI Get & Set TCP Server Autostart - Example
```

To create all these command objects, compile and run the CBX237M and CBX238M CL programs, following the instructions in the source headers. You'll also find compilation instructions in the respective source headers. The CBX237M CL program takes care of re-creating the Display NetServer Attributes command objects based on the new source versions, provided that these have been copied to the appropriate source files.

Related articles:

[APIs by Example: Parsing Externally Defined Record Buffers](#)

[APIs by Example: Update Externally Defined Record Buffer](#)

[APIs by Example: Server Support APIs Providing NetServer Management Facilities](#)

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

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

This article demonstrates the following Server Support APIs:

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

[Change Server Guest \(QZLSCHSG\) API](#)

[Change Server Information \(QZLSCHSI\) API](#)

[Change Server Name \(QZLSCHSN\) API](#)

[i5/OS Support for Windows Network Neighborhood Server APIs](#)

[ILE C/C++ Library Functions \(Record I/O APIs\)](#)

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

Source URL: <http://iprodeveloper.com/rpg-programming/apis-example-server-support-apis-enable-change-netserver-information>