

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

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

[System iNetwork Programming Tips Newsletter](#)

[Carsten Flensburg](#)

Carsten Flensburg

Thu, 01/29/2009 (All day)

I have previously covered the Profile Handle APIs that let you change a job's current user on the fly. Given the changes to the profile handle APIs that came with release 5.3, I'm revisiting this topic and adding a discussion of the server job concept. The profile-swapping technique that the Profile Handle APIs enable directly targets the requirement of server jobs being able to impersonate the user profiles on the behalf of which the server jobs perform their duties. Combined with the ability to initialize certain critical job attributes to reflect the user profile being swapped to (and back), provided by the Change Job (QWTCGJB) API, today's example should arm you with the API knowledge and techniques required to program and build your own servers.

To demonstrate the API programming involved I've created the Override User Profile (OVRUSRPRF) command. And to set you up to configure and implement user-defined TCP/IP servers that let you control and manage your own server jobs using the same interfaces and commands as IBM's TCP/IP servers, I've included links to a couple of excellent IBM articles discussing and demonstrating in great detail how this is done.

IBM added the ability to create user-defined servers with release 5.2, thereby offering in-house-developed servers the same facilities to control and manage your own server jobs as used for performing these tasks for IBM's TCP/IP servers. This implies that you can use the Start TCP/IP Server (STRTCPSVR) and End TCP/IP Server (ENDTCPSVR) commands to start and end user-defined servers, respectively, and the Change TCP/IP Server (CHGTCPSVR) command to control whether the user-defined server should start automatically together with the other servers that have been configured with the AUTOSTART(*YES) attribute. Likewise, you can use all available Navigator for i facilities that provide management and monitoring interfaces to TCP/IP servers and server jobs.

As for green-screen facilities, I also need to mention the Add TCP/IP Server (ADDTCPSVR) and the Remove TCP/IP Server (RMVTCPSVR) commands that provide the actual means of creating and removing a user-defined server, respectively. Note that both commands require Input/Output System Configuration (*IOSYSCFG) and All Object (*ALLOBJ) special authorities.

For a user-defined server to be fully and safely operational, quite a few pieces need to be fitted correctly together. As I mentioned, I've included links (at the end of this article) to a couple of articles written by IBM experts revealing the facts of the matter. In one article, Dawn May discusses the requirements and considerations related to creating a user-defined server, and in another article Bob Bittner walks you through all the steps that need to be performed in the user-defined server configuration process and shows you how to set up the OpenSSH server that IBM offers as part of the free product [5733-SC1 IBM Portable Utilities for i5/OS](#), as a user-defined server.

Bittner's article, however, doesn't explain how the OpenSSH server itself is installed and configured, but as luck would have it Scott Klement, the editor of this newsletter, has previously written an article thoroughly explaining all the bits and parts involved in meeting this challenge, and I provide the link at the end of this article. So as a bonus, following the documentation and the instructions in the

aforementioned articles you'll be able to set up OpenSSH as a TCP/IP service on your system. Given the wealth of information available in said articles, the scope of this article will remain at the level of briefly describing the main considerations and efforts that are part of creating a user-defined server. For this purpose, let's have a look at the ADDTCPSVR command prompt:

Add TCP/IP Server (ADDTCPSVR)

Type choices, press Enter.

Server special value	Character value
Program to call	Name
Library	Name
Server name	
Server type	
Autostart	*NO *YES, *NO
Text 'description'	*BLANK

Note that all values except the key parameter *Server special value* are subject to change at any point using the CHGTCPSVR command. Here's a brief explanation of the ADDTCPSVR command's parameters:

- The *Server special value* identifies the new, user-defined server and is added to the selection of available servers presented when you prompt for example the STRTCPSVR server and press F4 for the *Server* parameter.
- The *Program to call* parameter specifies the qualified name of the program called whenever an action (e.g., start, end, autostart) is performed against the server. This program is then responsible for carrying out the requested action. For a description of the parameter structure that the system uses to communicate with the exit program, following the "User Defined Servers Exit Program" link provided at the end of this article.
- The *Server name* is the name shown on the Navigator for i panels for this server.
- The *Server type* is a 30-byte character string that defines this specific server to the system. In order for the system to recognize the job as a server job, the server type job attribute must be set by the Change Job (QWTCGJB) API in the job(s) performing the server duties. The server type job attribute is also available as a selection criterion for the Open List of Jobs (QGYOLJOB) API, enabling you to select all or specific server jobs and retrieve information about these jobs in a program calling this API.
- The *Autostart* parameter defines whether this server will be automatically started when TCP/IP is started.

- The *Text 'description'* defines the textual description stored with the server entry in the system's server table.

When setting up the configuration and runtime environment for the server job, you need to address the following issues:

- In what subsystem should the server job run?
- Which user profile should be used to initiate the server job?
- Should the server job be enabled to run under the user profile being serviced, and how?
- How should the job description used for the server job be created?
- What considerations apply when starting and ending server jobs?

Please check out Dawn May's comprehensive article ["What You Need to Know to Develop a User-Defined TCP/IP Server"](#) for a detailed discussion of the above considerations and recommendations on how to approach these questions. Regarding the requirement of making a server job responsive to SIGTERM signals, I'll point you to [yet another article written by Scott Klement and dealing with this practice](#). For a practical example of how to implement a user-defined server, Bob Bittner's thorough article ["Exploit IBM i5/OS workload management capabilities for server applications"](#) will assist you in performing a step-by-step process setting up OpenSSH as a user-defined server on a system. Using these articles as a guideline will help you define and create your own user-defined servers, but in case you want to actually install and run the OpenSSH server on your system, turn to Scott Klement's detailed article ["OpenSSH: The Swiss Army Knife for Secure Networking"](#) for instructions on how to install and configure OpenSSH.

As for creating a user-defined server yourself, the core of the challenge that remains when you need to authenticate and swap to a user profile for which your server will perform its workload can be solved using the Profile Handle or Profile Token APIs. For an example of the latter, see the relevant link at the end of this article. Today's example uses the Profile Handle APIs to set a job's current user profile and optionally run the Change Job API to initialize the user-profile-related job attributes for the job, as well as ensure that these job attributes reflect the current user. The same APIs also support reinstating the initial user profile and job attributes once the current user has completed its work.

The sequence of events taking place to swap the current user profile of a job together with the related job attributes is the following:

1. Generate a profile handle for the job's current (initial) user profile by using the QsyGetProfileHandleNoPwd API. This task is quite simple because the current user profile should have enough authority to perform this function for itself.
2. Using the QsyGetProfileHandle API, generate a profile handle for the user profile that should be swapped to. To do so, you need to either specify a password for the user profile in question or have *ALLOBJ and *SECADM special authority or *READ authority to the user profile, depending on the user profile's current status. If you don't provide a password, the same API as in step 1 is used to produce the profile handle.
3. Set the job's current profile to new profile using the profile handle produced in step 2 with the QsySetToProfileHandle API.
4. Run the QWTCGJB API specifying format JOBCo300 to initialize user profile related job attributes to reflect the new current user of the job.
5. Perform the workload on behalf of the current user.

6. With the QsySetToProfileHandle API, reinstate the initial user profile as current user, using the profile handle produced in step 1.
7. Run the QWTCHGJB API specifying format JOBCo300 to initialize user-profile-related job attributes to reflect that the initial user profile was reinstated as the current user of the job. Note that the affected job attributes will be set to the values following a normal job initiation process, and not the values they had immediately before step 4.
8. Run the QsyReleaseProfileHandle API two times, specifying the profile handles generated in step 1 and 2, respectively. This step is particularly important because profile handles are a limited resource in a job, and a maximum of 20,000 handles can be created per job. After that, the space to store them is full, and the Get Profile Handle APIs will stop generating profile handles.

Apart from demonstrating the collaboration between the Profile Handle APIs and the Change Job API, the idea behind the Override User Profile (OVRUSRPRF) command is to provide a convenient shortcut to impersonate a user profile for the purpose of testing its authorization and function access. Because no adopted authority is involved, however, you will be granted access only to run the OVRUSRPRF having either *ALLOBJ and *SECADM special authority or *READ authority to the user profile overridden to-- unless you specify the user profile's password. For a means of obtaining a temporary elevation of your user profile's authority, see the Override Group Profile (OVRGRPPRF) command. I've provided a link to that APIs by Example article below.

Also note that to ensure a predictable outcome when running the OVRUSRPRF command, you cannot run it recursively; you have to complete a currently active OVRUSRPRF command before you're allowed to run it again. To give you an impression of the functionality provided with the OVRUSRPRF I've included the fully prompted OVRUSRPRF command's input panel below. Note that the *User password* and *Password value* parameters are mutually exclusive and initially not displayed. Which one appears is controlled by the *Password option* parameter:

Override User Profile (OVRUSRPRF)

Type choices, press Enter.

User profile	Name	
Password option	*PWD	*PWD, *PWDVAL
Override job attribute	*NONE	*NONE, *ALLSUP,
*NONASP...		
+ for more values		
Call program	*CMDENT	Name, *CMDENT,
*INLPGM		
Library		Name, *LIBL, *CURLIB
Display initial menu	*NO	*YES, *NO
Execute command		

User password		
Password value *NOPWDSTS	*NOPWD	*NOPWD, *NOPWDCHK,

The OVRUSRPRF command changes the current job to run under the specified user profile, which becomes the job's current user profile. The profile override locks the specified user profile and changes the current thread to run under the user profile and its group profiles. Any open files and objects allocated by the initial profile are accessible to the new profile.

By default, no other attributes associated with the user or group profile are replaced. The qualified job name doesn't change to reflect the new user profile. However, any object created by the thread while running under the new profile is owned by the new profile or its group profile. If the job is running single threaded and the job user identity has not been explicitly set by an API, the job user identity is changed to the name of the new profile. If the job is running multithreaded, the job user identity doesn't change.

Any spooled files created are, by default, owned by the new profile. This option is controlled by the spool file owner (SPLFOWN) parameter on the Create Printer File (CRTPRTF) command and is done by putting the file under a QPRTJOB job. Any spooled file command that references the spooled file with the job special value * will access only those files created before the initial user profile was overridden.

A QPRTJOB job is the name of a job that files are spooled under when the current job's user name isn't the same as the user profile currently running. For example, if you use the OVRUSRPRF command to override the profile to user JOE and create a spooled file, the file is spooled under job *nnnnnn*/JOE/QPRTJOB. This behavior ensures that user JOE owns the spooled file, and if that user uses the WRKSPLF command, the file is displayed.

Regarding the optional job attribute override, please note that setting the auxiliary storage pool (ASP) group of a job is performed by the Set ASP Group (SETASPGRP) command and all restrictions and conditions applying to the SETASPGRP command therefore apply if you specify this value or the single value *ALLSUP. As an example of such a restriction, it is not possible to run the SETASPGRP command if Programming Development Manager (PDM) functions are currently active. For your convenience, I've added the option *NONASP, which includes all currently supported job attributes except *ASPGRP. For more details about the restrictions that apply to the SETASPGRP command, please look up the command's help text. In case you prefer to specify individual job attributes, you also have the option of specifying a single or a combination of the job attributes available. Use function key F4 to see the list.

The OVRUSRPRF command also lets you specify a program to be called during the user profile override, specify whether the initial menu of the user profile being overridden should be displayed, and specify a command to be executed while the override is active. Note that these actions are performed in the sequence mentioned .

When the OVRUSRPRF command ends, the initial user profile is reinstated as the job's current user profile. Look up the OVRUSRPRF command's online help text for more information, including the restrictions that apply to running the OVRUSRPRF command.

This APIs by Example includes the following sources:

```

CBX199    -- RPGLE    -- Override User Profile - CPP
CBX199H   -- PNLGRP   -- Override User Profile - Help
CBX199V   -- RPGLE    -- Override User Profile - VCP
CBX199X   -- CMD      -- Override User Profile

CBX199M   -- CLP      -- Override User Profile - Build commands

```

To create all these objects, compile and run CBX199M, following the instructions in the source header. As always, compilation instructions are also in the respective source headers.

Implementing User Defined Servers Documentation:

User Defined Servers:

<http://as400bks.rochester.ibm.com/infocenter/systems/scope/index.html/topic/rzam3/rzam3kickoff.htm>

User Defined Servers Exit Program:

<http://as400bks.rochester.ibm.com/infocenter/systems/scope/index.html/topic/apis/xusrdfnsv.htm>

What You Need to Know to Develop a User-Defined TCP/IP Server (Dawn May, IBM)

<http://www.ibmssystemsmag.com/ibmi/may05/technicalcorner/8627p1.aspx>

Exploit IBM i5/OS workload management capabilities for server applications (Bob Bittner, IBM):

<http://www-03.ibm.com/servers/enabled/site/education/wp/7f4a/7f4a.pdf>

OpenSSH: The Swiss Army Knife for Secure Networking (Scott Klement):

<http://systeminetwork.com/article/openssh-swiss-army-knife-secure-networking>

End Your Programs Instantly When a *CNTRLD Shutdown Is Requested (Scott Klement):

<http://systeminetwork.com/article/end-your-programs-instantly-when-cntrld-shutdown-requested>

5733-SC1 -- IBM Portable Utilities for i5/OS:

<http://www-03.ibm.com/servers/enabled/site/porting/tools/openssh.html>

IBM documentation:

IBM Memo to Users 5.3 – Changes to profile handle and token APIs might require source code changes:

<http://publib.boulder.ibm.com/infocenter/iseres/v5r3/topic/rzahg/rzaq9.pdf#page=44>

System i Server table 5.4:

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

Add TCP/IP Server command:

<http://publib.boulder.ibm.com/infocenter/iseres/v5r4/topic/cl/addtcpsvr.htm>

Change TCP/IP Server command:

<http://publib.boulder.ibm.com/infocenter/iseres/v5r4/topic/cl/chgtcpsvr.htm>

Remove TCP/IP Server command:

<http://publib.boulder.ibm.com/infocenter/iseres/v5r4/topic/cl/rmvtcpsvr.htm>

Start TCP/IP Server command:

<http://publib.boulder.ibm.com/infocenter/iseres/v5r4/topic/cl/strtcpvr.htm>

End TCP/IP Server command:

<http://publib.boulder.ibm.com/infocenter/iseres/v5r4/topic/cl/endtcpsvr.htm>

Previously published related articles:

Make QSYGETPH API Calls V5R3 Compatible (Scott Klement):

<http://systeminetwork.com/article/make-qsygetph-api-calls-v5r3-compatible>

System Request: Who Is the User? V5R4 Makes a Change (Dan Riehl):

<http://systeminetwork.com/article/system-request-who-user-v5r4-makes-change>

APIs by Example: Swapping User Profiles:

<http://systeminetwork.com/article/apis-example-swapping-user-profiles>

APIs by Example: Override Group Profile:

<http://systeminetwork.com/node/60978>

APIs By Example: Profile Authorization Management:

<http://systeminetwork.com/node/61008>

APIs by Example: Profile Tokens:

<http://systeminetwork.com/node/61340>

This article demonstrates the following Security and Job APIs:

Get Profile Handle (QsyGetProfileHandle) API:

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

Get Profile Handle No Password (QsyGetProfileHandleNoPwd) API:

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

Set Profile Handle (QsySetToProfileHandle) API:

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

Release Profile Handle (QsyReleaseProfileHandle) API:

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

Change Job (QWTCHGJB) API:

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

You can retrieve the source code for this API example from:

http://www.pentontech.com/IBMContent/Documents/article/57692_799_OvrUsrPrf.zip.

Source URL: <http://iprodeveloper.com/rpg-programming/apis-example-use-security-and-job-apis-user-defined-servers>