

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

APIs by Example: User Function Registration APIs, Part 2

[System iNetwork Programming Tips Newsletter](#)

[Carsten Flensburg](#)

Carsten Flensburg

Thu, 08/25/2005 (All day)

This week I will continue coverage of the User Function Registration APIs. You'll get more information about the APIs and two new commands that you can call from your CL programs or the command line to add and retrieve User Function usage for a user profile. Also included are some comments and insight into the design and concepts of the User Function Registration APIs provided by Patrick Botz, who is a Senior Technical Staff Member with the eServer Security Architecture & Consulting team of IBM's Client Technology Center in Rochester.

In part one of this article, I was very concerned about pointing out that User Function APIs as such would not offer any protection against access to resources outside of the application(s) implementing the User Function check. This is, of course, correct and worth considering. Admittedly, I didn't go into detail about how the User Function APIs will provide an effective authorization facility even when taking this caution into account.

This prompted the following response from Patrick Botz, who emphasized that the User Function APIs were not designed to protect access to unprotected objects, but were rather created to provide the equivalent of Application Defined Special Authority/Privileges. Patrick illustrated this point with the following example:

"Consider a large application. All users must be allowed to use the application programs. However, some programs in the application provide multiple application functions. For example, a program provides the option to create a report of customer credit card balances with and without credit card numbers.

All authorized application users should be allowed to see the report without the credit card numbers, but only a few are allowed to see the reports with the credit card numbers.

No user is authorized to the database files that contain the information. The program adopts enough authority to access the database files. The application programmer can choose to create an application specific function/special authority by using the function usage APIs.

The programmer can design the application to check if the requested user is allowed access to the "CREDIT CARD NUMBER" function. Only if she or he does will the application produce the report with the sensitive data. A user can only see the sensitive data if they:

1. Are authorized to the application; and
2. Are allowed CREDIT CARD NUMBER functional usage (i.e. have CREDIT CARD NUMBER special authority).

This is essentially the same as existing commands and APIs that require access to the resource and *JOBCTL. In other words, the function acts the same way as a special authority."

Thanks to Patrick Botz for taking the time to provide background and perspective on the User Function APIs. In the next API by Example column, I will present a sample application based on the methodology described above.

Continuing my ongoing effort to create a set of CL commands that will present a green screen interface to the User Function APIs, I have now arrived at the challenge of changing and checking a user profile's User Function access. (Note that V5R3 also brought a set of CL commands to accomplish the change usage part; run the command GO CMDFCNUSG to examine the OS offering.)

To set a user profile's usage setting, I have written the Change User Function Usage (CHGUSRFCNU) command. Here's how the command prompt looks:

Change User Function Usage (CHGUSRFCNU)			
Type choices, press Enter.			
Function ID		
User profile	Name, *	CURRENT
Usage	*ALLOWED	*ALLOWED, *DENIED, *NONE

The command has a prompt override program (POP) that, once the function ID and user profile has been entered and Enter pressed, retrieves and displays the user's current usage. If no current usage is registered, the command prompt displays the usage opposite to the function ID's default usage.

As mentioned last week, you can also access and maintain the User Functions using the Operations Navigator GUI-interface. See part one of this article for instructions on how to accomplish that.

To retrieve a user profile's usage indicator from CL-program, the Retrieve Function Usage (RTVFCNUSG) command can be used. Here's the command prompt:

Retrieve Function Usage (RTVFCNUSG)			
Type choices, press Enter.			
Function ID		
User profile	Name, *	CURRENT
CL var for USGIND	(1) . .	Character value	

The Check Function Usage algorithm performed by the system when determining whether a user profile has usage access to a User Function is described in exact details here:

<http://as40obks.rochester.ibm.com/iserics/v5r2/ic2924/info/apis/QSYCKUFU.htm>

As with most retrieve commands, it can't be run from the command line, so I've written a small sample CL program to give an example of how to use the RTVFCNUSG command.

I have also included a service program that offers a slightly simplified interface to the Check Function Usage API. You can, of course, also just copy the code directly into your own program, if you prefer.

The CHGUSRFCNU command includes the following sources:

CBX141 -- Command processing program.
CBX141X -- Command definition source member.
CBX141O -- Command prompt override program.
CBX141H -- Command help text panel group.

The RTVFCNUSG command includes the following sources:

CBX142 -- Command processing program.
CBX142X -- Command definition source member.
CBX142H -- Command help text panel group.
CBX142T -- Sample program to test command.

And finally the Check Function Usage service program:

CBX142S -- Service program.

Compilation instructions are found in the source headers.

This article demonstrates the following APIs:

Change Function Usage Information (QsyChangeFunctionUsageInfo) API:
<http://as400bks.rochester.ibm.com/iserics/v5r2/ic2924/info/apis/qsychnfui.htm>

Retrieve Function Usage Information (QsyRetrieveFunctionUsageInfo) API:
<http://as400bks.rochester.ibm.com/iserics/v5r2/ic2924/info/apis/qsyrtnfui.htm>

Check Function Usage (QsyCheckUserFunctionUsage) API:
<http://as400bks.rochester.ibm.com/iserics/v5r2/ic2924/info/apis/QSYCKUFU.htm>

Normal End (CEETREC) API:
<http://as400bks.rochester.ibm.com/iserics/v5r2/ic2924/info/apis/CEETREC.htm>

Send Program Message (QMHSNDPM) API:
<http://publib.boulder.ibm.com/iserics/v5r2/ic2924/info/apis/QMHSNDPM.htm>

You can retrieve the source code for this API example from the following link:
http://www.pentontech.com/IBMContent/Documents/article/51418_34_UserFuncReg2.zip

Source URL: <http://iprodeveloper.com/rpg-programming/apis-example-user-function-registration-apis-part-2>