

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

APIs by Example: User Function Registration APIs, Part 1

[System iNetwork Programming Tips Newsletter](#)

[Carsten Flensburg](#)

Carsten Flensburg

Thu, 08/11/2005 (All day)

This week's installment of APIs by Example will turn the spotlight on a relatively new iSeries security concept called User Function Registration. User Functions were introduced with release V4R3 and offer a method to implement application-level security, which lets you specify the user profiles that should be allowed to access specific functions or interfaces within your application. This article describes the elements of the User Function Registration facility and also includes two new commands to help you maintain User Functions.

Application-level security in this context means that your application includes a call to the Check User Function Usage API to determine if the current user profile is allowed to access the specific function you want to protect. Contrary to general iSeries resource security, which is built into the operating system and is in effect regardless of the interface you are accessing a resource through, application security leaves it up to you to ensure that resources are not accessed through other available interfaces or applications. In other words, resources that are protected by application-level security are only secured as long as they are accessed through that specific application.

Let's say you have a customer maintenance application and you use the Function Registration to enforce which users are allowed to display customer information and which users are allowed to update customer information. This authorization scheme will work fine as long as the users are accessing the customer information by means of that customer maintenance application. But it offers no protection against users accessing the customer data through other available interfaces and applications if these do not also perform a check against that specific Function Registration.

A User Function does however offer great flexibility when it comes to function access administration. When you register or change the User Function, you can specify whether *ALLOBJ special authority should be ignored when performing the function usage check. This allows you to -- within the limits mentioned above -- actually deny security officer user profiles access to specific functions and data.

You can also specify whether the User Function default usage is to allow access or to deny access. In the first case, if most users should be allowed access, you then only register the user profiles that should be denied access. In the latter, if the majority of users should be denied access, you only register the users that should be allowed access.

As an example of User Functions deployment in the real world, the iSeries operating system utilizes User Functions to allow user profiles without *ALLOBJ special authority to look at the joblogs of user profiles that have *ALLOBJ special authority. Normally *JOBCTL special authority is required to look at another user's job log, and *ALLOBJ special authority is required to look at the job log of a user that has *ALLOBJ authority. The latter requirement can be bypassed if the user profile running the DSPJOBLOG command is registered with the QIBM_ACCESS_ALLOBJ_JOBLOG function ID.

In the above event, IBM added a check against the mentioned User Function if insufficient authority was found by the DSPJOBLOG command processing program when trying to access a job having *ALLOBJ special authority. Then if function usage access is found, access to the DSPJOBLOG function is granted. For example, it lets you give operations staff members access to privileged users' joblogs without having to grant them *ALLOBJ special authority. This is of course only one example of how to put application-level security and User Functions to work.

This week I will show you how to both register and deregister a User Function by means of two new commands that I have created for this article: The Add Function Registration (ADDFCNREG) and the RMVFCNREG (Remove Function Registration) commands. These commands also offer a starting point for those who want to create their own interfaces to the User Function Registration APIs that are the engine of the two commands.

Once you have registered a User Function, you can specify the user profiles that should be either allowed or denied access to the registered function by changing the function's usage information.

In a follow up to this article, I will provide a command that let you do that for V5R2 and earlier. If you are on V5R3, the following CL commands are available to administer and maintain User Function Registrations and their usage:

WRKFCNUSG -- Work with Function Usage

DSPFCNUSG -- Display Function Usage

CHGFCNUSG -- Change Function Usage

For earlier releases, as well as the current one, Operations Navigator offers a GUI interface to User Functions by following the route below:

- a) Right click on the connection (system) you want to work with.
- b) Select 'Application Administration'.
- c) Select 'Host Applications'.
- d) Maintain the registered User Functions.

But now it's time to have a look at the command prompts -- The ADDFCNREG command prompt has the following layout:

```

                                Add Function Registration (ADDFCNREG)
Type choices, press Enter.

Function ID . . . . .

Function type . . . . . *ADMIN      *ADMIN, *GROUP, *PRODUCT
Function category . . . . . *HOST      *HOST, *LOCNAV, *LOC...
Function group ID . . . . . *NONE
Function product ID . . . . .

*ALLOBJ special authority . . *USED      *NOTUSED, *USED
Default usage . . . . . *ALLOWED *DENIED, *ALLOWED
Function name . . . . .

Function name CCSID . . . . . *JOB      1-65535, *JOB, *HEX
Function description . . . . .

```

Function description CCSID . . .	*JOB	1-65535, *JOB, *HEX
Function name message ID:		
Message file	*BLANK	Name, *BLANK
Library		Name
Message ID		Name
Function description msg ID:		
Message file	*BLANK	Name, *BLANK
Library		Name
Message ID		Name
Allow replace	*NO	*NO, *YES

Please note that although you can specify both a message ID and a name or description for the Function name and Function description parameters respectively, only the message ID text will be used for display if a message ID parameter is specified. However, both parameters will be stored in the User Function registration. If neither is specified, the Function ID will be used as the name.

The RMVFCNREG command prompt looks like this:

```

                                Remove Function Registration (RMVFCNREG)
Type choices, press Enter.
Function ID . . . . .

```

The individual attributes are explained in detail in the help panel groups that are included for both commands.

Only function type *ADMIN registrations are used to store usage information. Function type *PRODUCT and *GROUP registrations are only used to identify and group *ADMIN registrations. Type *GROUP is only optional though. So in order to start registering *ADMIN User Functions, you need to first register at least one *PRODUCT User Function. Here's the sequence of events:

1. Register one or more type *PRODUCT User Functions.
2. Register one or more type *GROUP User Functions (optional).
3. Register one or more type *ADMIN User Functions.
4. Register user profiles' function usage for one or more *ADMIN User functions.
5. In your application(s), check the current user's function usage access against the appropriate User Function.

Items 4 and 5 will be the topic of the next installment of APIs by Example.

The ADDFCNREG command includes the following sources:

CBX1401 -- Command processing program.
 CBX1401X -- Command definition source member.
 CBX1401O -- Command prompt override program.
 CBX1401V -- Command validity checking program.
 CBX1401H -- Command help text panel group.

The RMVFCNREG command includes the following sources:

CBX1402 -- Command processing program.
CBX1402X -- Command definition source member.
CBX1402H -- Command help text panel group.

Compilation instructions are found in the source headers.

This article demonstrates the following APIs:

Register Function (QsyRegisterFunction) API:

<http://as400bks.rochester.ibm.com/iserics/v5r2/ic2924/info/apis/qsyrqfn.htm>

Retrieve Function Information (QsyRetrieveFunctionInformation) API:

<http://as400bks.rochester.ibm.com/iserics/v5r2/ic2924/info/apis/qsyrtrvi.htm>

Deregister Function (QsyDeregisterFunction) API:

<http://as400bks.rochester.ibm.com/iserics/v5r2/ic2924/info/apis/QSYDRGFN.htm>

Check User Special Authorities (QSYCUSRS) API:

<http://as400bks.rochester.ibm.com/iserics/v5r2/ic2924/info/apis/QSYCUSRS.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/51361_30_UserFunctReg.zip

Source URL: <http://iprodeveloper.com/rpg-programming/apis-example-user-function-registration-apis-part-1>