SKYVIEW PARTNERS, INC

What Authorities Do Programmers Really Need?

by Carol Woodbury

Laws and regulations require that users be given only enough authority to do their jobs, and auditors require IT departments to reduce the number of “powerful” users on the system. As a result, the programming staff comes under scrutiny. Why? Because they have often been given lots of power—in i5/OS terms, that means programmers have been given the *ALLOBJ special authority. The question I am often asked is, “What authorities do programmers need to do their jobs?” Unfortunately, the answer is, “That depends.” This article looks at how you can determine the answer to this question for your environment and also explains what authorities programmers do *not* need.

What programmers don’t need: *ALLOBJ special authority

It may take a bit of investigation to determine exactly what authorities programmers require, but they rarely need the *ALLOBJ special authority. However, many programmers will insist that it’s a requirement to perform their job functions on both development and production systems. I can say with certainly that, unless the programmers double as security officers, they do not need *ALLOBJ on a permanent basis.

Giving *ALLOBJ authority to programmers is a bad idea because it enables them to access every object on the system. Because the actions of an *ALLOBJ user cannot be controlled, you cannot maintain proper change management controls if you have programmers with *ALLOBJ authority. Even if you have implemented change management software, programmers with *ALLOBJ authority can directly access and modify production source. Programmers with *ALLOBJ authority can easily cover up their actions by deleting objects such as joblogs and journal receivers and by clearing the history log. Finally, programmers with *ALLOBJ authority on production systems can access, modify, or delete production database programs and files; view or download private data; or run encryption routines to decrypt data. All of these actions bring the integrity and availability of your data into question, which is why programmers’ authorities are under scrutiny and regulatory compliance requirements are being implemented.

What programmers do require

Now that we know what programmers don’t need, we must determine what programmers do require. To begin, look at the tasks programmers perform. If their sole responsibility is architecting, coding, and testing applications (which tends to be the case in larger organizations), the requirements are fairly straightforward. The development environment and change management software you’re using will dictate the actual implementation. However, on a development system, generally programmers need to be able to create, compile, test, and debug their programs. They also need to be able to view (but not update) various levels of source that are “above” the development level. This implies that the source files are set to *PUBLIC authority *USE or *PUBLIC authority *EXCLUDE, with the programmers’ group having *USE authority to the files.

When programmers need access to production systems

Any auditor will tell you that programmers should not have access to production systems—period. Although preventing them from gaining access may be a great goal, it’s not a reality for most shops. In certain situations, programmers need access to production systems. The key is to not give them too much authority and to monitor their activity when they are accessing the production system. Let’s look at two situations in which access to a production system is required for a programmer.

Situation 1
A problem arises on a production system but the programmer cannot recreate it on a test or development system. In this case, the programmer has to debug the problem on the production system. To do so, the programmer needs *CHANGE authority to the program as well as *USE authority to the Start Debug (STRDBG) command. You may not want to give the programmer *CHANGE authority to the production programs, so instead, give the programmer the *SERVICE special authority. With *SERVICE, the programmer needs only the *USE authority to debug the program. Many programs are already configured as *PUBLIC *USE, which makes this an easier option to implement. Before implementing this technique, however, you will want to understand and accept the dangers associated with the *SERVICE special authority. (These issues are documented in the iSeries Security Reference book, found at http://publib.boulder.ibm.com/infocenter/iseries/v5r4/topic/rzahg/rzahgsecref.htm.) Fortunately, the *SERVICE special authority is not as dangerous as it once was. In the past, the *SERVICE special authority automatically granted users the ability to access powerful service tools through the Start Service Tools (STRSST) command. But now, to access service tools, you must also have a service tools ID. If you avoid giving programmers a service tools ID, the dangers associated with *SERVICE are reduced.

Situation 2
In most organizations, data in a production file may eventually need to be altered. The reason could be that a maintenance program doesn't exist to maintain production files when new values need to be added, incorrect data was entered by end users and insufficient parameter checking allowed the value to be entered into the database, or program logic caused an incorrect calculation. Whatever the situation—the data is not correct or a file needs to be updated—the problem is reported and programmers need to resolve the issue. Some would add *ALLOBJ to a programmer's profile, use a tool that swaps to an *ALLOBJ user, or provide the programmer with the user ID and password of a profile that has *ALLOBJ. I don't care for any of these solutions because there's no i5/OS requirement that the programmer have *ALLOBJ to update the file—the programmer just needs sufficient authority.

You have several options for providing programmers with sufficient authority. One technique for handling this problem is to have the programmer use a profile whose group profile is the one that owns the application. That way, the programmer has sufficient authority to work with the application programs and files but does not have access to everything on the system. Although this is a powerful profile, it is not as powerful as a profile with the *ALLOBJ special authority. This profile should not be shared (multiple programmers should not be using this profile) because you want the journal entries to reflect which programmer performed the action—otherwise, accountability is lost. Nor should the programmer be allowed to sign on with this profile all of the time. The profile should be either a one-time-use profile (created for this purpose and then deleted when the task is complete) or enabled for a particular programmer to use in a specific emergency situation.

Another option is to give only sufficient authority to the programmer to modify the specific file. If you have application files secured with an authorization list, this becomes very easy. Just add the programmer to the authorization list (granting them *CHANGE authority to the list) and they have only enough authority to modify the files. This is less dangerous than temporarily placing them in the group that owns the application because their access rights are limited to only files and with only *CHANGE authority rather than *ALL. Again, the programmer should not have this authority permanently – only when an update is required.

Whatever method you choose, you should always have documentation to back up the use of the profile. Also, command auditing should be enabled so that all commands entered are audited. To enable this type of auditing, make sure the QAUDCTL system value specifies *AUDLVL and then run the following command for each profile:

```
CHGUSRAUD USRPRF(POWER_PGMR) AUDLVL(*CMD)
```

In extremely rare occasions, a programmer will need the *ALLOBJ special authority. As with the profile just described, a process should be put around the use of this profile so that it is enabled for sign-on only after management approvals have been granted. In addition, make sure that auditing is turned on for this profile.

I recommend that every month or so, you examine the issues that necessitated these powerful profiles. Look for cases in which the programmer simply didn't have authority to a particular command or task. Then look for ways to enable this authority without giving it to the programmer (such as through a program the programmer calls that adopts authority).

QPGMR as a group profile

An easy way to grant all programmers the same authorities and provide them with the same special authorities is to put all programmers in a group and place the authority at the group level rather than the individual level.

Some organizations choose to make the IBM profile QPGMR the programmers' group profile, but I prefer not to do that for three reasons. First, doing so may provide authority to more commands and objects than you would like. Second, many vendors have chosen to have QPGMR be the profile that owns their application. Therefore, when you assign the programmers to QPGMR, you are giving them ownership (that is, *ALL) rights to these applications, which is not at all appropriate. Finally, because QPGMR has *SAVSYS and *JOBCTL special authorities, making QPGMR the programmers' group profile means the programmers have these special authorities. To perform development tasks, programmers do not need any special authorities.

I prefer to create a separate profile that is assigned as the programmers' group. The profile doesn't own any applications, has no special authorities, and is not an IBM profile. It is likely, however, that the programmers will need to be authorized to some of the commands to which QPGMR is authorized. Running the following command produces a list of commands to which QPGMR is authorized:

```
DSPUSRPRF QPGMR *CMDAUT
```

After examining this list, you may wish to grant *USE authority to the programmers' group profile for the commands the programmers need to run. Examples include the STRDBG and Trace Job (TRCJOB) commands.

When programmers perform administrative tasks

When a programmer also performs administrative tasks or rotates on-call duties with administrators (as is often the case in small to medium-sized companies), determining the exact authorities required takes a bit more investigation. The easiest way to determine the authorities required is to make a list of the tasks the programmers perform. For example, if they are on call, they may also start and stop printers, end jobs, stop and start subsystems, or start and stop TCP/IP servers.

Once you've listed the tasks or commands the programmers run, you can look up the required authorities in the iSeries Security Reference book, Appendix D: Authority Required for Objects Used by Commands, found at http://publib.boulder.ibm.com/infocenter/iseries/v5r4/topic/books/sc415302.pdf. Appendix D lists the authority for the objects the commands use as well as the special authorities required to run the commands. For example, to end jobs (other than their own), programmers will need the *JOBCTL special authority.

In small System i shops, I have seen programmers also act as back-up administrators. In those cases, programmers may need authority to more commands and third-party system management products. They may also need to be given additional special authorities (including *IOSYSCFG) to create network devices and interfaces, configure TCP/IP or *SAVSYS to perform saves, or restore libraries from back-up tapes. They may even need *SECADM to create or re-enable user profiles if user profile administration tools are not in place.

More tips for avoiding *ALLOBJ

Programmers on call often need the *ALLOBJ special authority if they have to debug failed jobs running under an *ALLOBJ user. (*ALLOBJ is required to view the joblog of an *ALLOBJ user.) However, configuring one of the Host applications in Application Administration or running the Change Function Usage (CHGFCNUSG) command allows you to authorize users with the *JOBCTL special authority to view the joblogs of an *ALLOBJ user. (For more information about Host applications in Application Administration, see the article “Alternatives to special authorities” in the September/October 2005 issue of *iSeries 400 Experts Journal*, found at http://www.iseries400experts.com/aej/article.jsp?article_id=43808&volume_id=5921 .) For example, running the following command authorizes all users in the PGMR_GROUP to view the joblog of *ALLOBJ users:

```
CHGFCNUSG FCNID(QIBM_ACCESS_ALLOBJ_JOBLOG) USER(PGMR_GROUP) USAGE(*ALLOWED)
```

Other information required to debug a failed job often comes from user profiles. Since user profiles default to *PUBLIC *EXCLUDE, programmers who do not have *ALLOBJ cannot see any profile attributes. If this is information your programmers need, you have some options. One option is to write a CL program that prompts the Display User Profile (DSPUSRPRF) command. Configure the program to adopt its owner’s authority and make sure the owner has authority to all profiles on the system. A second option is to give the programmers’ group profile *READ authority to the profiles whose attributes you want them to view. Granting them *READ authority allows them to display the user profile attributes. However, giving them more authority (such as *USE) will allow them to submit jobs as any profile on the system, which is not appropriate and should be avoided.

Summary

The purpose of this article is not to suggest that you revoke programmers’ authority so that they can no longer perform their jobs (although, undoubtedly, that’s what some programmers will claim!). Rather, the article explains that because users with powerful authorities—namely *ALLOBJ special authority—are increasingly coming under scrutiny from auditors, more laws and regulations are demanding that excess authorities be removed. In light of these demands, you now have a method to determine what authorities your programmers require so that when *ALLOBJ is removed, they can continue to perform their job functions.

***Carol Woodbury** is President and co-founder of SkyView Partners, Inc., a firm specializing in security policy and compliance management software as well as security consulting and remediation services. Carol has over 16 years in the security industry, 10 of those working for IBM's Enterprise Server Group as the AS/400 Security Architect and Chief Engineering Manager of Security Technology. Carol's second book, Experts' Guide to OS/400 and i5/OS Security, is available at www.amazon.com .*

Carol can be reached at carol.woodbury@skyviewpartners.com