

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

Keeping System Resource Security Under Control

[System iNEWS Magazine](#)

[Carsten Flensburg](#)

Carsten Flensburg

Mon, 11/21/2011 (All day)

[Click here](#) to download the code bundle.

To report code errors, email SystemiNetwork.com

Effectively securing your system's resources in terms of applications and business data takes a lot of effort. It also requires a well-

devised strategy as well as comprehensive planning and implementation measures. As in many of life's circumstances, good intentions will take you only part of the way. Upon successful completion of all planned activities, you'll need to ensure that all vital security policies and precautions are continuously enforced as the daily business once again takes over the agenda.

If not, your next internal or external system security audit most certainly will provide you with a wake-up call, making my aforementioned statement brilliantly clear to you. Your auditors will listen with great interest as you explain the basic principles of the Application-Only resource security model that constitutes the backbone of your system security policy (for a list of Application-Only resources, see Find Out More, below). The fact that only the application programs possess adopted authority to access your data files, and that all user profiles have access to these programs only through group profile membership, might initially impress your auditors, depending on their knowledge of the IBM i security mechanisms.

After all, the presented security model is flexible and, to a great extent, exploits the advanced built-in security features of the IBM i. The excitement is bound to quickly fade, however, when the audit reveals that the payroll file, following a recent change that added two new fields, is now owned by a developer user profile, and the file's public authority is defined as *CHANGE. The given example is, of course, quite obvious. But other more subtle failures to comply with the security scheme in place might be more difficult to discover, while in turn causing a similar serious impact on your system's resource protection.

Even if the security exposure remains undiscovered at first, having to explain the failure to comply with your signed-off security policy to your security auditors might in its own right spell disaster. Needless to say, that part of implementing your security model would in fact include well-defined processes that handle moving application objects into production and apply the necessary object ownership, private authorities, authorization list, program authority adoption attributes, and whatever else is required. But experience tells me that regardless of the effort put into ensuring that nothing goes wrong, eventually something will.

To help me verify and manage object authority and security attribute settings on demand, and to avoid potentially unpleasant conversations with my security auditors, I've designed and created a couple of object authority CL commands. One command helps me compare the security-related attributes of two specific objects and reports found discrepancies; the other performs the same

comparison for a list of objects. Here, I explain the two commands and how you can take advantage of them.

The CMPOBJAUT Command

Managing object authority requires control of a number of object attributes, such as object ownership, object private authorities, object primary group, and object authorization list, to mention the most common properties involved. In addition, database file objects support field-level authority, and program objects define how the program interacts with adopted authority. As for the latter, programs can both inherit adopted authority from previous call levels and adopt their own owner's authority, depending on the program's USEADPAUT and USRPRF attribute setting, respectively.

Ensuring that two objects have corresponding authority attributes therefore implies comparing the two objects' various security-related attributes. To make this operation as simple and efficient as possible, I've created the Compare Object Authority (CMPOBJAUT) command (for a list of the source files, see "Creating the Compare and Work with Object Authority Commands," below). Let me introduce you to the CMPOBJAUT command prompt in [Figure 1](#) and define the required input parameters.

You specify the qualified name and type of object for which to compare authority as well as the reference object's qualified name and object type. CMPOBJAUT then compares all relevant object authority attributes of the specified object against the corresponding attributes of the reference object. Here's an example of how the CMPOBJAUT command would look if you wanted to compare program AAA in library XXX against program BBB in library ZZZ:

```
CMPOBJAUT OBJ (XXX/AAA)
          OBJTYPE (*PGM)
          REFOBJ (ZZZ/BBB)
          REFOBJTYPE (*OBJTYPE)
          OUTPUT (*)
```

You can direct the command's output to either display on screen or print with your job's spooled output. [Figure 2](#) shows the detailed outcome of the requested comparison on a display panel. As you can see, the two objects involved in the object authority comparison, as well as their authority-related attributes and any object-type-specific attributes, display side by side. The compare status defines the first found difference, and a plus (+) sign at the end of the status line indicates the discovery of more differences. Pressing F7 presents the complete list of the differences that CMPOBJAUT found.

Both objects' private authorities are listed at the bottom of the panel, and all private authorities assigned to the object in question that are not matched on the reference object are highlighted. Note that the security-related commands executed by the panel's function keys are all targeted against the specified object, letting you immediately change the relevant authority attributes of the object. Using F5, you can repeat and update the authority comparison directly from the display panel following a successful change.

In [Figure 2](#), I've included both sets of available function keys to show all the functions accessible from the display panel. To toggle between the two sets, press F24. The accompanying cursor-sensitive help text explains in detail both the command and display panel.

The WRKOBJAUT Command

The Work with Object Authority (WRKOBJAUT) command lets you indicate a range of objects defined by either a generic object name or the special value *ALL for objects located in the specified library for comparison against either a specific reference object or the corresponding object in a stated reference library. You have the option of further narrowing the selected object range by criteria such as object attribute, user-defined attribute, creator-user profile, owning-user profile, as well as object create, change, and last-used date ranges.

You can also indicate one or more object comparison statuses to include only objects failing the object authority attribute comparison for a number of predefined causes, in case you're investigating a specific authority issue. Here's a list of single-value object statuses and their explanations:

- ***ALL** includes all objects in the list, irrespective of the outcome of the object authority information comparison.
- ***OBJFAIL** includes only objects failing one or more of the object authority information comparison tests.
- ***OBJPASS** includes only objects passing all object authority information comparison tests.
- ***OBJFAILX** includes only objects failing one or more of the object authority information comparison tests, excluding the reference object existence test. This implies that only objects identified by name and type that exist in both libraries are listed.

In addition, you can select from 12 different special values to further narrow the resulting object list. Please refer to WRKOBJAUT's help text for all the details.

[Figure 3](#) shows the WRKOBJAUT command's prompt panel. Here, you can specify a specific object, a generic object name or the special value *ALL, a library name, and the object type or the special value *ALL to include all object types in the list. The latter is supported only if you state a Reference library. You must indicate a single object type if you choose to specify a reference object instead. The associated cursor-sensitive help text panel group fully documents the WRKOBJAUT command and all its parameters. To demonstrate the output of the WRKOBJAUT command, I executed the following command on my system:

```
WRKOBJAUT OBJ(TESTLIB/C*)  
          OBJTYPE(*ALL)  
          REFLIB(PRODLIB)  
          STATUS(*OBJFAILX)
```

Because of the number of objects meeting the selection criteria, the command ran for a little while and then produced the output list panel in [Figure 4](#). The object list shows how many statuses the object authority attribute comparison process uncovered and the status text for the first one found. In the event that the comparison process finds more than one status, use option 8=Compare to evoke the CMPOBJAUT command to identify all object statuses. When the Compare Object Authority panel displays, you simply press F7 to see the complete status list for the selected object.

The list panel also includes list options that let you immediately execute various object- and authority-related commands against the selected objects to further investigate or correct the current object authority status. The list panel offers an alternative view as well, which includes information about the object owner, primary group, authorization list, and public authority. As always, the accompanying help text explains in detail the list panel, the list panel options, and the list panel function keys.

If you prefer to run WRKOBJAUT in batch due to the performance impact of investigating a large number of objects, you can direct the command output to either a printed list or an output file. Both these options also serve as a facility to document the current resource security status on your system regularly or upon request.

I wish you the best of luck in tracking down any object authority issues—before they get to you!

Carsten Flensburg is a **System iNEWS** senior technical editor and has been an IBM i programmer since 1992. His focus areas are modular system design, API programming, system integration, and communication. Carsten currently works as an IBM i application development manager for a European vacation rental company called Novasol, which is a part of the U.S.-based Wyndham Worldwide Corporation.

Creating the Compare and Work with Object Authority Commands

The following sources are involved in creating the CMPOBJAUT and WRKOBJAUT commands:

```
CBX807L  -- RPGLE  -- Retrieve Command Parameter Value List
CBX807   -- RPGLE  -- Compare Object Authority - CPP
CBX807V  -- RPGLE  -- Compare Object Authority - VCP
CBX807C  -- RPGLE  -- Compare Object Authority - Choice Program
CBX807E  -- RPGLE  -- Compare Object Authority - UIM Exit Program
CBX807H  -- PNLGRP -- Compare Object Authority - Help
CBX807P  -- PNLGRP -- Compare Object Authority - Panel Group
CBX807X  -- CMD    -- Compare Object Authority

CBX808   -- RPGLE  -- Work with Object Authority - CPP
CBX808C  -- RPGLE  -- Work with Object Authority - Choice Program
CBX808H  -- PNLGRP -- Work with Object Authority - Help
CBX808P  -- PNLGRP -- Work with Object Authority - Panel Group
CBX808V  -- RPGLE  -- Work with Object Authority - VCP
CBX808X  -- CMD    -- Work with Object Authority

CBX807M  -- CLP    -- Compare Object Authority - Build command
CBX808M  -- CLP    -- Work with Object Authority - Build command

SQLCLI_H -- RPGLE  -- SQL CLI API Header File
```

To create all objects discussed in this article, compile and run the CBX807 and CBX808M CL programs, following the instructions in the source header. You'll also find compilation instructions in the respective source headers. For the WRKOBJAUT command processing program CBX808 to compile, you'll need to download and copy the SQLCLI_H member included with the above source package to a QRPGLSRC source file in your job's library list. Thanks to Scott Klement, who wrote the SQLCLI_H include member.

Find Out More

IBM Security documentation:

[Planning and setting up system security](#)

[Planning object authority](#)

[Planning resource security](#)

Application-only security model:

[Application-Only Access: A Resource Security Strategy, Part 1](#)

[Application-Only Access: A Resource Security Strategy, Part 2](#)

[Wayne O. Evans: Application Only Access presentation](#)

Source URL: <http://iprodeveloper.com/rpg-programming/keeping-system-resource-security-under-control>