A Practical Guide to IBM i Authority Collection

Steve Riedmueller
Director – IBM i Security Services
IBM CHAMPION
steve@kisco.com

Outline and Goals

Remove the "mystery" from Authority Collection

Provide commands and queries so you can "hit the ground running"

Present Authority Collection as a tool you should add to your toolbox

- Concepts and Terminology
- Using Authority Collection
- Use Cases, Investigating Data, and Problem Solving
- Finding Active Collections and General Housekeeping

kisco

What is Authority Collection?

- Added in IBM i 7.3 – allows you to collect information for a specific profile:
  - Objects accessed (including objects in the IFS)
  - Authority required for the access
  - Current authority
  - Source of current authority (group, authorization list, adopted authority, etc)
- Enhanced in IBM i 7.4 – allows you to collect information for a specific object – including objects in the IFS

kisco

Why Use Authority Collection?

To reduce access without fear of breaking something

kisco

Section: Authority Collection Concepts

Authorization Name?

What is an "authorization name"?

kisco

Authorization Name?

What is an "authorization name"?

A user profile!

kisco

Authority Check?

What is an "authority check"?

kisco

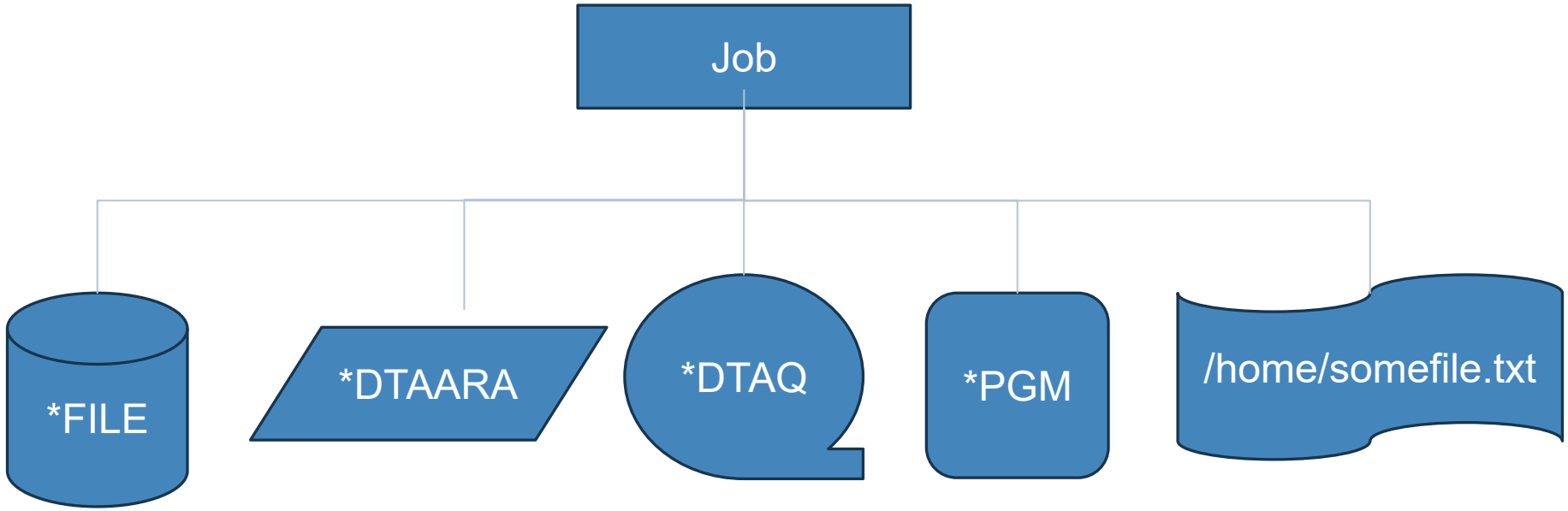
Authority Check?

What is an "authority check"?

Under-the-covers OS security validation

kisco

Authority Check?



kisco

Authority Collection Repository?

- Stores your Authority Collection data
- Not accessible directly as an "object"
- Not a traditional "log file" or table
- Collection of *unique* authority checks
- Accessible only through SQL views

Authorization Name AUTHORIZATION_NAME	Check Timestamp CHECK_TIMESTAMP	System Object Name SYSTEM_OBJECT_NAME	System Object Schema SYSTEM_OBJECT_SCHEMA	System Object Type SYSTEM_OBJECT_TYPE	Authority Check Successful AUTHORITY_CHECK_SUCCESSFUL	Check Any Authority CHECK_ANY_AUTHORITY	Detailed Required Authority DETAILED_REQUIRED_AUTHORITY
RBTINSUSER	2024-04-03 09:16:45.369791	UI06796922	RBTREPLIB	*USRIDX	1	0	*READ
RBTINSUSER	2024-04-03 09:18:08.987569	UI06799058	RBTREPLIB	*USRIDX	1	0	*OBJEXIST *OBJMGT *OBJOPR *RE
RBTINSUSER	2024-04-03 09:18:08.987579	UI06799058	RBTREPLIB	*USRIDX	1	1	*OWNER *OBJEXIST *OBJMGT *OBJ
RBTINSUSER	2024-04-03 09:18:09.027775	U07491679	RBTREPLIB	*USRSPC	1	0	*OBJEXIST *OBJMGT *OBJOPR *RE
RBTINSUSER	2024-04-03 09:18:09.027786	U07491679	RBTREPLIB	*USRSPC	1	0	*OBJOPR
RBTINSUSER	2024-04-03 09:18:09.030389	UI06799058	RBTREPLIB	*USRIDX	1	0	*READ
RBTINSUSER	2024-04-03 09:20:19.651009	UI06798951	RBTREPLIB	*USRIDX	1	0	*OBJEXIST *OBJMGT *OBJOPR *RE
RBTINSUSER	2024-04-03 09:20:19.651015	UI06798951	RBTREPLIB	*USRIDX	1	1	*OWNER *OBJEXIST *OBJMGT *OBJ
RBTINSUSER	2024-04-03 09:20:19.704113	U07491572	RBTREPLIB	*USRSPC	1	0	*OBJEXIST *OBJMGT *OBJOPR *RE

kisco

Save/Restore Considerations

Each object's authority collection setting (on/off) is stored and saved with the object, but the collected data is stored separately in the repository

- SAVSECDTA will save the fact that AC is turned on
- Authority collection repository is saved in a full system save
- Alternative: write the data out to a table!

Extract/Safeguard Authority Collection Data

Authority collection data can be “dumped” into a table/PF

```
--Write all AUTHORITY_COLLECTION_LIBRARIES entries to an outfile
CREATE TABLE SRIEDMUE.AC_OUTFILE AS
    (SELECT *
        FROM QSYS2.AUTHORITY_COLLECTION_LIBRARIES) WITH DATA;


--View the data in the outfile
SELECT * FROM SRIEDMUE.AC_OUTFILE;
```

AUTHORIZATION_NAME	CHECK_TIMESTAMP	SYSTEM_OBJECT_NAME	SYSTEM_OBJECT_SCHEMA
SRIEDMUE	2025-02-24 14:14:45.218557	AJ	SRIEDMUE
SRIEDMUE	2025-02-24 14:14:40.259577	AJ	SRIEDMUE
SRIEDMUE	2025-02-24 14:14:40.259607	AJ	SRIEDMUE

kisco

Types of Authority Collection

User-based Collection

- Records all authority checks for a specific user profile
- Can be limited to certain libraries/objects/types, but targets one user
- Intuitive one-step start/stop procedure

Object-based Collection

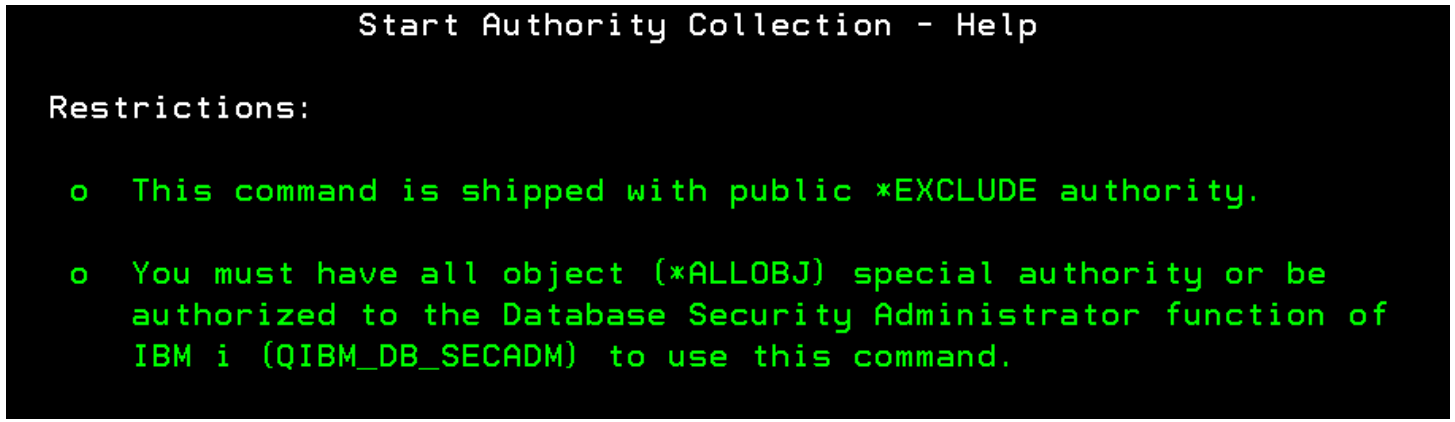
- QSYS "native" objects (programs, files, data areas, etc.)
- IFS objects (directories, stream files, etc.)
- Less intuitive two-step start/stop procedure

kisco

Ability to start/stop Authority Collection

What authority is required for turning collections on/off or deleting collection data?

*ALLOBJ special authority or function usage for QIBM_DB_SECADM



kisco

Supported Object Types

Object types that support the authority collection flag:

*CMD	*JRNRCV	*SQLUDT
*DTAARA	*LIB	*SQLXSR
*DTADCT	*OUTQ	*SRVPGM
*DTAQ	*PGM	*USRIDX
*FILE	*QMFORM	*USRQ
*JOBD	*QMQRY	*USRSPC
*JOBQ	*QRYDFN	IFS paths/files
*JRN	*SQLPKG	

kisco

Section: Using Authority Collection

Starting Collection and Viewing Data

Authority Collection – More considerations

- You must have either *ALLOBJ special authority or be authorized to the Database Security Administrator function (QIBM_DB_SECADM) to start the collection. You can administer this function via Application Administration (which is available as part of Navigator for i) or the Work with Function Usage (WRKFCNUSG) command.
- Limit User Function (Application Administration) settings are not recorded.
- For those features where authority to an object plus some special authority is required, the special authority requirement is not recorded.
- To display whether a collection is active and/or an authority collection repository exists for a user, run the DSPUSRPRF (Display User Profile) command and scroll to the end of the display. Use DSPOBJD to see when the collection is active for an object (IBM i 7.4.)
- While a user's collection setting is saved when running the SAVSECDTA (Save Security Data) command, the actual collection data is not. Likewise, saving an object does not save the collection (IBM i 7.4)
- If an authority collection exists and the profile or object is deleted, its authority collection is also deleted.
- If you specify to collect authority for all objects in all libraries, some objects, such as operating system programs are omitted from the collection; however, objects, such as IBM-supplied commands will be included in the collection data
- See Chapter 10 of the *IBM i Security Reference* manual for more details.

kisco

Start Authority Collection for a user profile

Start collection for user SRIED, but only for one library:

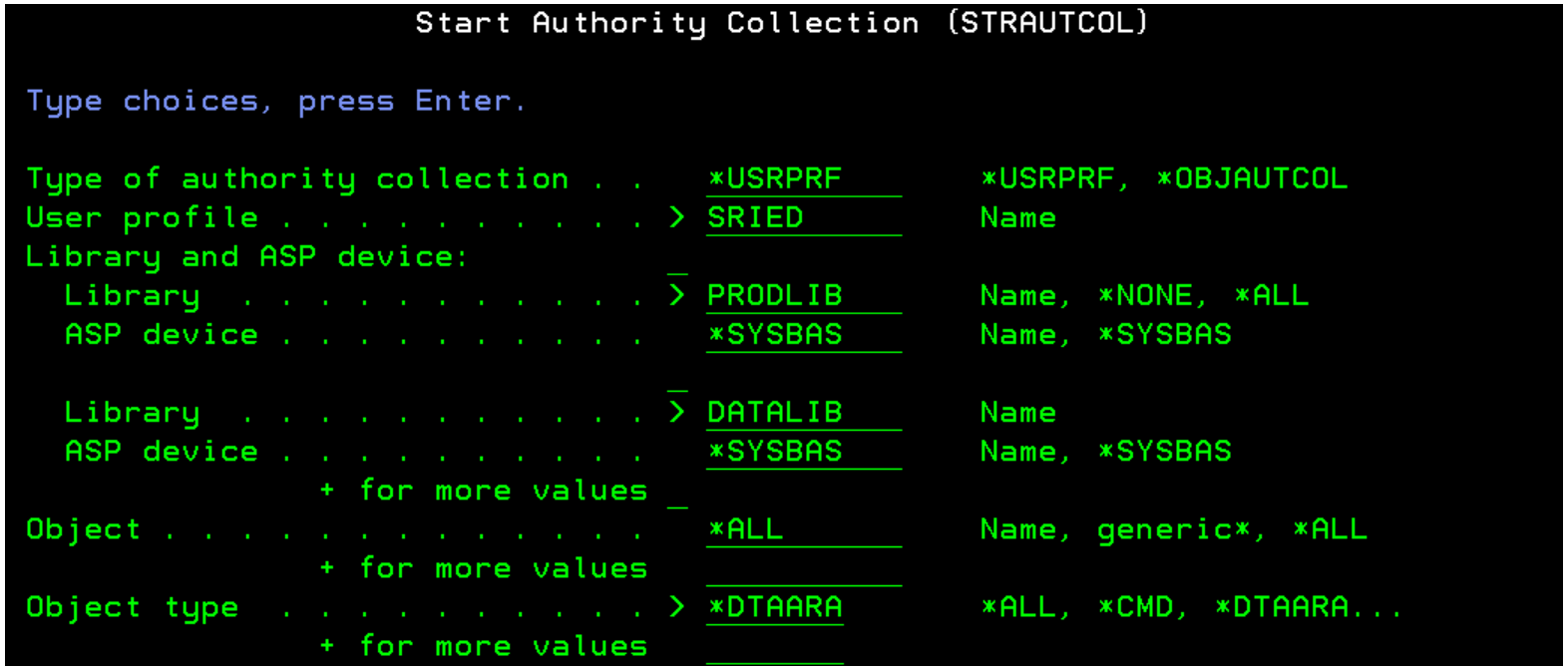
```
STRAUTCOL USRPRF(SRIED) LIBINF((PRODLIB))
```

kisco

Start Authority Collection for a user profile

Start collection for user SRIED, but only for *DTAARA in two libraries:

```
STRAUTCOL USRPRF(SRIED)
          LIBINF((PRODLIB) (DATALIB))
          OBJTYPE(*DTAARA)
```



Start Authority Collection for a user profile

Start collection for user SRIED, but only for the IFS:

```
STRAUTCOL USRPRF(SRIED)
          INCFSOBJ(*ALL)
```

Start Authority Collection for a user profile

Start a full collection for user SRIED (all libraries and IFS):

```
STRAUTCOL USRPRF(SRIED)
          LIBINF(*ALL)
          INCFSOBJ(*ALL)
```

kisco

Start Authority Collection for a "native" object

1. Set the flag "on" for the object:

```
CHGAUTCOL OBJ('/QSYS.LIB/SRIED.LIB/QCLSRC.FILE')
          AUTCOLVAL(*OBJINF)
```

2. Start the object-based collection function globally (no harm if already active)

```
STRAUTCOL TYPE(*OBJAUTCOL)
```

kisco

Start Authority Collection for an entire library

1. Set the flag "on" for all objects in a library:

```
CHGAUTCOL OBJ('/QSYS.LIB/SRIED.LIB')
          AUTCOLVAL(*OBJINF)
          SUBTREE(*ALL)
```

2. Start the object-based collection function globally (no harm if already active)

```
STRAUTCOL TYPE(*OBJAUTCOL)
```

kisco

Start Authority Collection for IFS files

1. Set the flag "on" for an IFS directory and all objects therein:

```
CHGAUTCOL OBJ('/home/sried/')
          AUTCOLVAL(*OBJINF)
          SUBTREE(*ALL)
```

2. Start the object-based collection function globally (no harm if already active)

```
STRAUTCOL TYPE(*OBJAUTCOL)
```

kisco

How to view Authority Collection data

User-based collections:

AUTHORITY_COLLECTION

Object-based collections:

AUTHORITY_COLLECTION_OBJECT
AUTHORITY_COLLECTION_LIBRARIES
AUTHORITY_COLLECTION_FSOBJ
AUTHORITY_COLLECTION_DLO
AUTHORITY_COLLECTION_IFS

Or use Navigator!

Note: QSYS2.AUTHORITY_COLLECTION_OBJECT and QSYS2.AUTHORITY_COLLECTION_LIBRARIES return the same results. However, QSYS2.AUTHORITY_COLLECTION_OBJECT will perform better when the number of entries in the authority collection is large and you are looking for a specific object or objects in a specific library.
QSYS2.AUTHORITY_COLLECTION_LIBRARIES will perform better when the number of entries in the authority collection is small or you are looking for all or most objects in the authority collection.

kisco

Which fields to select?

Explanation of all the fields is available in IBM doc:
https://www.ibm.com/docs/en/i/7.6.0?topic=collection-authority-views

REQUIRED_AUTHORITY CURRENT_AUTHORITY CURRENT_ADOPTED_AUTHORITY	DETAILED_REQUIRED_AUTHORITY DETAILED_CURRENT_AUTHORITY DETAILED_CURRENT_ADOPTED_AUTHORITY
*USE	*OBJOPR, *READ, *EXECUTE
*CHANGE	*OBJOPR, *READ, *ADD, *DLT, *UPD, *EXECUTE
*ALL	*OBJEXIST, *OBJMGT, *OBJOPR, *OBJALTER, *OBJREF, *READ, *ADD, *DLT, *UPD, *EXECUTE
*EXCLUDE	-
-	any other combination of authorities (aka "USER DEF")

kisco

Which fields to select?

AUTHORITY_SOURCE Identifies the source of the user's authority (or lack thereof)

Potential values:

- **USER *ALLOBJ** – The user profile has *ALLOBJ special authority
- **USER OWNERSHIP** – The user owns the object
- **USER PRIVATE** – The user has private authority to the object
- **AUTHORIZATION LIST OWNERSHIP** – The user is the owner of the authorization list which secures the object
- **AUTHORIZATION LIST PRIVATE** – The user has private authority via the authorization list which secures the object
- **GROUP *ALLOBJ** – The user's group profile has *ALLOBJ
- **GROUP OWNERSHIP** – The user's group profile is the owner of the object

kisco

Which fields to select?

AUTHORITY_SOURCE - continued

Potential values:

- **GROUP PRIVATE** – The user's group has private authority to the object
- **PRIMARY GROUP** – The user profile (or one of its groups) is the "primary group" on the object
- **AUTHORIZATION LIST GROUP OWNERSHIP** – The user's group is the owner of the auth list securing the object
- **AUTHORIZATION LIST PRIMARY GROUP** – The user's group is the "primary group" of the auth list securing the object
- **AUTHORIZATION LIST GROUP PRIVATE** – The user's group has private authority in the auth list securing the object
- **AUTHORIZATION LIST PUBLIC** – The user has authority via the public authority in the auth list securing the object
- **PUBLIC** – The user has authority via the object's public authority

kisco

Which fields to select?

ADOPT_AUTHORITY_USED Identifies whether adopted authority was used (1) or not (0)

ADOPTED_AUTHORITY_SOURCE Identifies the source of the user's authority (or lack thereof)

Potential values:

- **ADOPTED *ALLOBJ** – The user adopted *ALLOBJ from the adopting program owner
- **ADOPTED OWNERSHIP** – The user adopted ownership from the program owner
- **ADOPTED PRIMARY GROUP** – The user adopted primary group authority from the program owner
- **ADOPTED PRIVATE** – The user adopted private authority from the program owner
- **ADOPTED AUTHORIZATION LIST OWNERSHIP** – The user adopted authority of the auth list owner from the program
- **ADOPTED AUTHORIZATION LIST PRIMARY GROUP** – The user adopted authority of the auth list's primary group
- **ADOPTED AUTHORIZATION LIST PRIVATE** – The user adopted authority of a user with private authority in the AUTL

kisco

Which fields to select?

CHECK_ANY_AUTHORITY Identifies whether this was a check for "any" authority or not (1 or 0).

This is a less-specific authority check – the user only needs to have any one of the "required" authorities.

System Object Schema	System Object Name	System Object Type	Check Any Authority	Authority Check Successful	Detailed Required Authority
SRSAMPLE	PROJECT	*FILE	1	1	*OWNER *OBJEXIST *OBJMGT *OBJALTER *OBJ...
SRSAMPLE	SALARY	*FILE	1	1	*OBJMGT
SRSAMPLE	PROJECT	*FILE	1	1	*READ *ADD *DLT *UPD
STEVER	TESTFILE	*FILE	1	1	*READ *ADD *DLT *UPD
STEVER	TESTFILE	*FILE	1	1	*OWNER *OBJEXIST *OBJMGT *OBJALTER *OBJ...
STEVER	TESTFILE	*FILE	1	1	*OWNER *OBJEXIST *OBJMGT *OBJALTER *OBJ...

kisco

Which records to omit?

Omit entries related to operating system programs that adopt authority:

```
SELECT *
    FROM QSYS2.AUTHORITY_COLLECTION
    WHERE AUTHORIZATION_NAME = 'SRIED'
          AND (ADOPTING_PROGRAM_SCHEMA IS NULL
              OR ADOPTING_PROGRAM_SCHEMA <> 'QSYS')
    ORDER BY CHECK_TIMESTAMP;
```

kisco

Section: Use Cases and Problem Solving

Use Cases and Problem Solving

- Authority reduction (remove *ALLOBJ or powerful groups from a user profile)
- Lock down an application library, or an individual file
- Lock down an IFS path
- Who is using a Netserver fileshare?
- Who is using an object in a particular library?
- Who is using a system command like ENDSBS?
- Research usage of the Netserver "guest user"
- Research authority failure (AF) entries in the audit journal
- Investigate FTP processes (QTFTPxxxxx)
- What process/user is accessing a file in the IFS root?

kisco

Section: Use Cases and Problem Solving

Authority Reduction

kisco

Authority Reduction

Users with too much object authority, or even *ALLOBJ

1. Enable collection for the user profile
2. Allow the data to collect
3. Review the collection
4. Grant the required object authority
5. Revoke the excess authority

kisco

Authority Reduction – find high-powered profiles

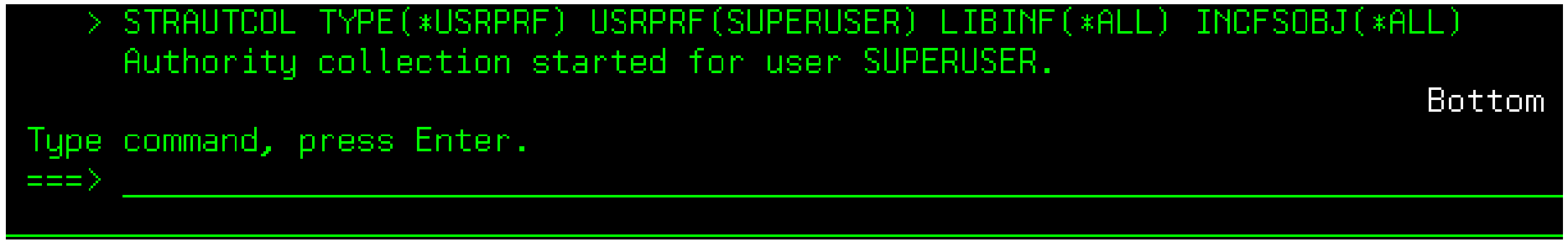
```
--Profiles with *ALLOBJ special authority
SELECT AUTHORIZATION_NAME, STATUS, SPECIAL_AUTHORITIES,
       GROUP_PROFILE_NAME, SUPPLEMENTAL_GROUP_LIST, TEXT_DESCRIPTION
    FROM QSYS2.USER_INFO_BASIC
    WHERE SPECIAL_AUTHORITIES LIKE '%*ALLOBJ%';

--Profiles that have *ALLOBJ or are members of a group having *ALLOBJ
SELECT AUTHORIZATION_NAME, STATUS, SPECIAL_AUTHORITIES,
       GROUP_PROFILE_NAME, SUPPLEMENTAL_GROUP_LIST, TEXT_DESCRIPTION
  FROM QSYS2.USER_INFO_BASIC
  WHERE SPECIAL_AUTHORITIES LIKE '%*ALLOBJ%' OR AUTHORIZATION_NAME IN
    (SELECT USER_PROFILE_NAME
       FROM QSYS2.GROUP_PROFILE_ENTRIES WHERE GROUP_PROFILE_NAME IN
         (SELECT AUTHORIZATION_NAME
            FROM QSYS2.USER_INFO_BASIC
            WHERE SPECIAL_AUTHORITIES LIKE '%*ALLOBJ%'
            AND GROUP_ID_NUMBER <> '0'));
```

kisco

Authority Reduction – start authority collection

Start authority collection for the target user profile:

```
STRAUTCOL TYPE(*USRPRF)
          USRPRF(SUPERUSER)
          LIBINF(*ALL)
          INCFSOBJ(*ALL)
```

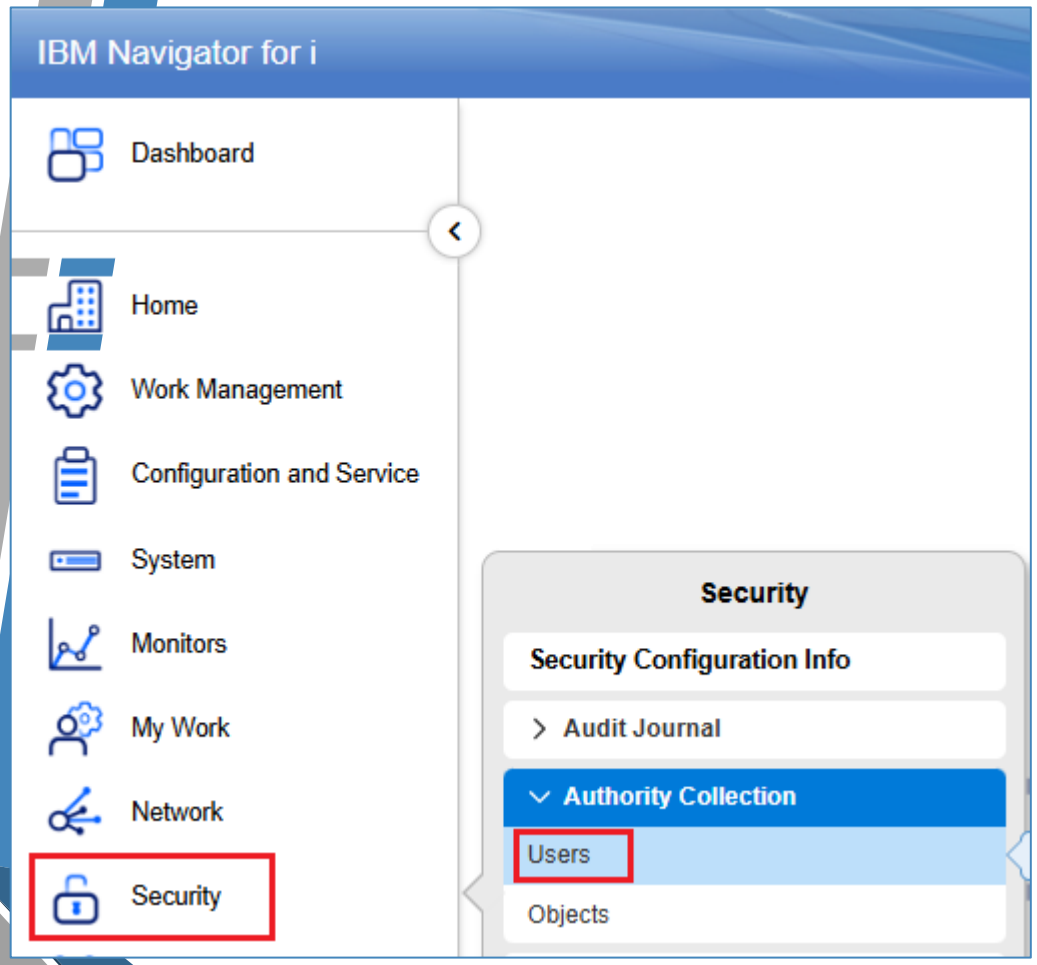


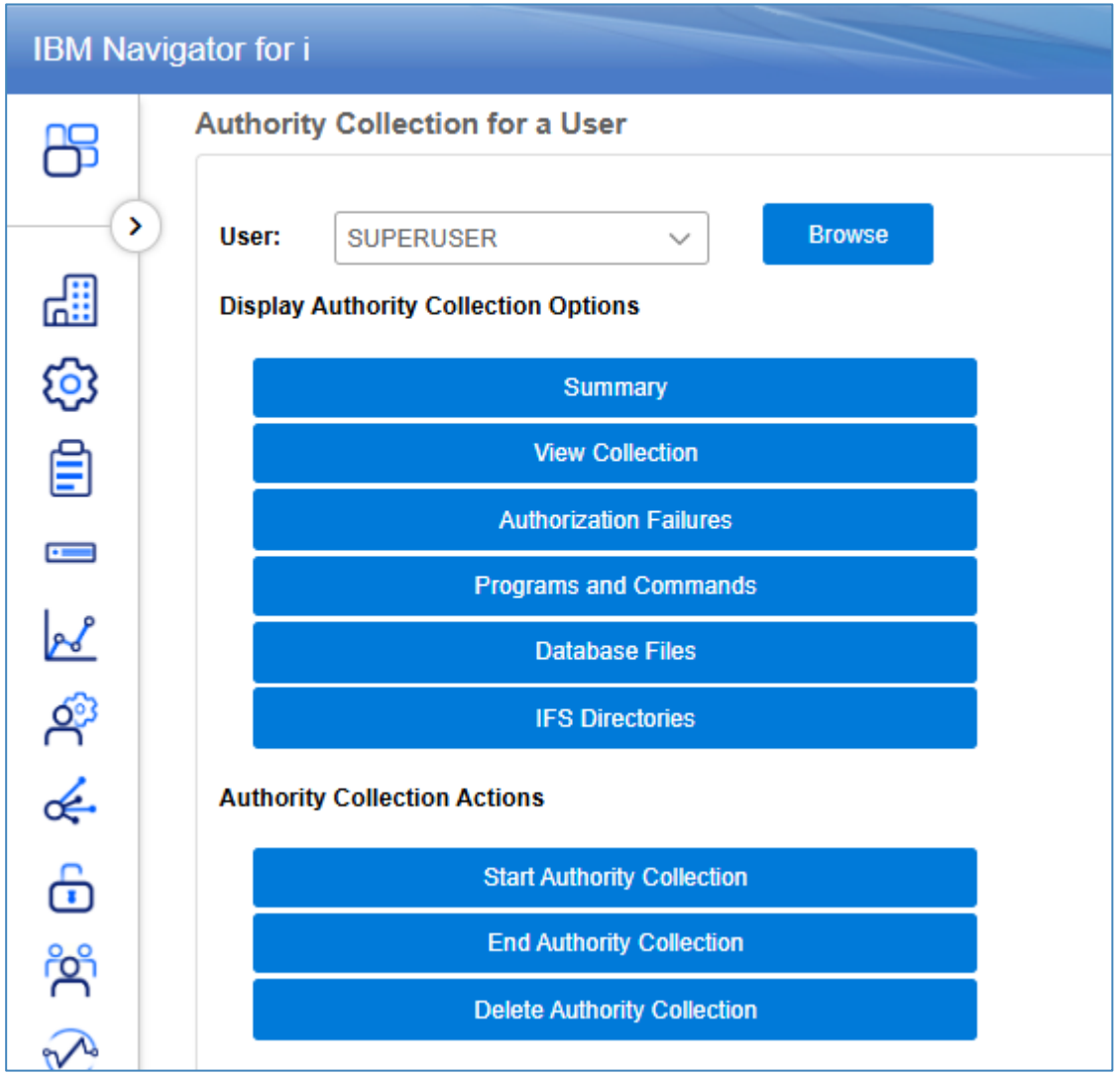
Authority Reduction – analyze the data

```
--List the required authority for all objects that the user touched
SELECT DISTINCT SYSTEM_OBJECT_SCHEMA, SYSTEM_OBJECT_NAME,
                SYSTEM_OBJECT_TYPE, CHECK_ANY_AUTHORITY,
                AUTHORITY_CHECK_SUCCESSFUL, DETAILED_REQUIRED_AUTHORITY,
                DETAILED_CURRENT_AUTHORITY, AUTHORITY_SOURCE
    FROM QSYS2.AUTHORITY_COLLECTION
    WHERE AUTHORIZATION_NAME = 'SUPERUSER'
          AND (ADOPTING_PROGRAM_SCHEMA IS NULL
              OR ADOPTING_PROGRAM_SCHEMA <> 'QSYS')
    ORDER BY SYSTEM_OBJECT_SCHEMA, SYSTEM_OBJECT_NAME;
```

kisco

Authority Reduction – analyze the data





kisco

Authority Reduction – analyze the data

Note: A single authority check does not tell the whole story.

For the whole process to work, the user needs to have the full *cumulative* list of "required authorities" for the objects.

System Object Schema	System Object Name	System Object Type	Check Any Authority	Authority Check Successful	Detailed Required Authority	Detailed Current Authority	Authority Source
QSYS	SRSAMPLE	*LIB	0	1	*EXECUTE	*OBJEXIST *OBJM...	USER *ALLOBJ
SRSAMPLE	CREATEEMP	*PGM	0	1	*EXECUTE	*OBJEXIST *OBJM...	USER *ALLOBJ
SRSAMPLE	DELETEEMP	*PGM	0	1	*EXECUTE	*OBJEXIST *OBJM...	USER *ALLOBJ
SRSAMPLE	DEPARTMENT	*FILE	0	1	*READ	*OBJEXIST *OBJM...	USER *ALLOBJ
SRSAMPLE	DEPARTMENT	*FILE	0	1	*OBJOPR	*OBJEXIST *OBJM...	USER *ALLOBJ
SRSAMPLE	EMPLOYEE	*FILE	0	1	*ADD	*OBJEXIST *OBJM...	USER *ALLOBJ
SRSAMPLE	EMPLOYEE	*FILE	0	1	*READ	*OBJEXIST *OBJM...	USER *ALLOBJ
SRSAMPLE	EMPLOYEE	*FILE	0	1	*READ *DLT	*OBJEXIST *OBJM...	USER *ALLOBJ
SRSAMPLE	EMPLOYEE	*FILE	0	1	*OBJOPR	*OBJEXIST *OBJM...	USER *ALLOBJ
SRSAMPLE	VIEWDEPT	*PGM	0	1	*EXECUTE	*OBJEXIST *OBJM...	USER *ALLOBJ
SRSAMPLE	VIEWEMP	*PGM	0	1	*EXECUTE	*OBJEXIST *OBJM...	USER *ALLOBJ

kisco

Authority Reduction – grant authority

Grant the required object authority via authorization list, group, or private authority

```
GRTOBJAUT OBJ(SRSAMPLE)    OBJTYPE(*LIB)  USER(SUPERUSER) AUT(*EXECUTE)
GRTOBJAUT OBJ(CREATEEMP)   OBJTYPE(*PGM)  USER(SUPERUSER) AUT(*EXECUTE)
GRTOBJAUT OBJ(DELETEEMP)   OBJTYPE(*PGM)  USER(SUPERUSER) AUT(*EXECUTE)
GRTOBJAUT OBJ(DEPARTMENT) OBJTYPE(*FILE) USER(SUPERUSER) AUT(*OBJOPR *READ)
GRTOBJAUT OBJ(EMPLOYEE)    OBJTYPE(*FILE) USER(SUPERUSER) AUT(*OBJOPR *READ *ADD *DLT)
GRTOBJAUT OBJ(VIEWDEPT)    OBJTYPE(*PGM)  USER(SUPERUSER) AUT(*EXECUTE)
GRTOBJAUT OBJ(VIEWEMP)     OBJTYPE(*PGM)  USER(SUPERUSER) AUT(*EXECUTE)
```

kisco

Authority Reduction – revoke *ALLOBJ

Revoke *ALLOBJ special authority or remove the group membership.

```
CHGUSRPRF USRPRF(SUPERUSER) SPCAUT(*NONE)
CHGUSRPRF USRPRF(SUPERUSER) GRPPRF(*NONE)
```

kisco

Authority Reduction – review the results

```
--List the required authority for all objects that the user touched
SELECT DISTINCT SYSTEM_OBJECT_SCHEMA, SYSTEM_OBJECT_NAME,
                SYSTEM_OBJECT_TYPE, CHECK_ANY_AUTHORITY,
                AUTHORITY_CHECK_SUCCESSFUL, DETAILED_REQUIRED_AUTHORITY,
                DETAILED_CURRENT_AUTHORITY, AUTHORITY_SOURCE
    FROM QSYS2.AUTHORITY_COLLECTION
    WHERE AUTHORIZATION_NAME = 'SUPERUSER'
          AND (ADOPTING_PROGRAM_SCHEMA IS NULL
              OR ADOPTING_PROGRAM_SCHEMA <> 'QSYS')
          AND CHECK_TIMESTAMP > '2024-04-01 14:00:00'
    ORDER BY SYSTEM_OBJECT_SCHEMA, SYSTEM_OBJECT_NAME;
```

kisco

Authority Reduction – review the results

System Object Schema	System Object Name	System Object Type	Check Any Authority	Authority Check Successful	Detailed Required Authority	Detailed Current Authority	Authority Source
QSYS	SRSAMPLE	*LIB	0	1	*EXECUTE	*EXECUTE	USER PRIVATE
SRSAMPLE	CREATEEMP	*PGM	0	1	*EXECUTE	*EXECUTE	USER PRIVATE
SRSAMPLE	DELETEEMP	*PGM	0	1	*EXECUTE	*EXECUTE	USER PRIVATE
SRSAMPLE	DEPARTMENT	*FILE	0	1	*READ	*OBJOPR *READ	USER PRIVATE
SRSAMPLE	DEPARTMENT	*FILE	0	1	*OBJOPR	*OBJOPR *READ	USER PRIVATE
SRSAMPLE	EMPLOYEE	*FILE	0	1	*ADD	*OBJOPR *READ *ADD *DLT	USER PRIVATE
SRSAMPLE	EMPLOYEE	*FILE	0	1	*READ	*OBJOPR *READ *ADD *DLT	USER PRIVATE
SRSAMPLE	EMPLOYEE	*FILE	0	1	*READ *DLT	*OBJOPR *READ *ADD *DLT	USER PRIVATE
SRSAMPLE	EMPLOYEE	*FILE	0	1	*OBJOPR	*OBJOPR *READ *ADD *DLT	USER PRIVATE
SRSAMPLE	VIEWDEPT	*PGM	0	1	*EXECUTE	*EXECUTE	USER PRIVATE
SRSAMPLE	VIEWEMP	*PGM	0	1	*EXECUTE	*EXECUTE	USER PRIVATE

kisco

Section: Use Cases and Problem Solving

Locking Down a Library or File

Lock down a Library or File – start AC

Start authority collection for the file:

```
CHGAUTCOL OBJ('/QSYS.LIB/SRSAMPLE.LIB/SALARY.FILE')
          AUTCOLVAL(*OBJINF)

STRAUTCOL TYPE(*OBJAUTCOL)
```

kisco

Lock down a Library or File – analyze the data

```
SELECT DISTINCT AUTHORIZATION_NAME, SYSTEM_OBJECT_SCHEMA, SYSTEM_OBJECT_NAME,
                SYSTEM_OBJECT_TYPE, CHECK_ANY_AUTHORITY,
                AUTHORITY_CHECK_SUCCESSFUL, DETAILED_REQUIRED_AUTHORITY,
                DETAILED_CURRENT_AUTHORITY, AUTHORITY_SOURCE
    FROM QSYS2.AUTHORITY_COLLECTION_LIBRARIES
    WHERE SYSTEM_OBJECT_SCHEMA = 'SRSAMPLE'
          AND SYSTEM_OBJECT_NAME = 'SALARY'
          AND (ADOPTING_PROGRAM_SCHEMA IS NULL
              OR ADOPTING_PROGRAM_SCHEMA <> 'QSYS');
```

Authorization Name	System Object Schema	System Object Name	System Object Type	Check Any Authority	Authority Check Successful	Detailed Required Authority	Detailed Current Authority	Authority Source
SUPERUSER	SRSAMPLE	SALARY	*FILE	0	1	*READ	*OBJOPR *READ *EXECUTE	USER PRIVATE
SRIEDMUE	SRSAMPLE	SALARY	*FILE	1	1	*OBJEXIST *OBJMGT...	*OBJEXIST *OBJMGT *O...	USER *ALLOBJ
QSECOFR	SRSAMPLE	SALARY	*FILE	1	1	*OWNER *OBJEXIST ...	*OBJEXIST *OBJMGT *O...	USER *ALLOBJ
SUPERUSER	SRSAMPLE	SALARY	*FILE	0	1	*OBJOPR	*OBJOPR *READ *EXECUTE	USER PRIVATE
QSECOFR	SRSAMPLE	SALARY	*FILE	0	1	*OBJOPR	*OBJEXIST *OBJMGT *O...	USER *ALLOBJ

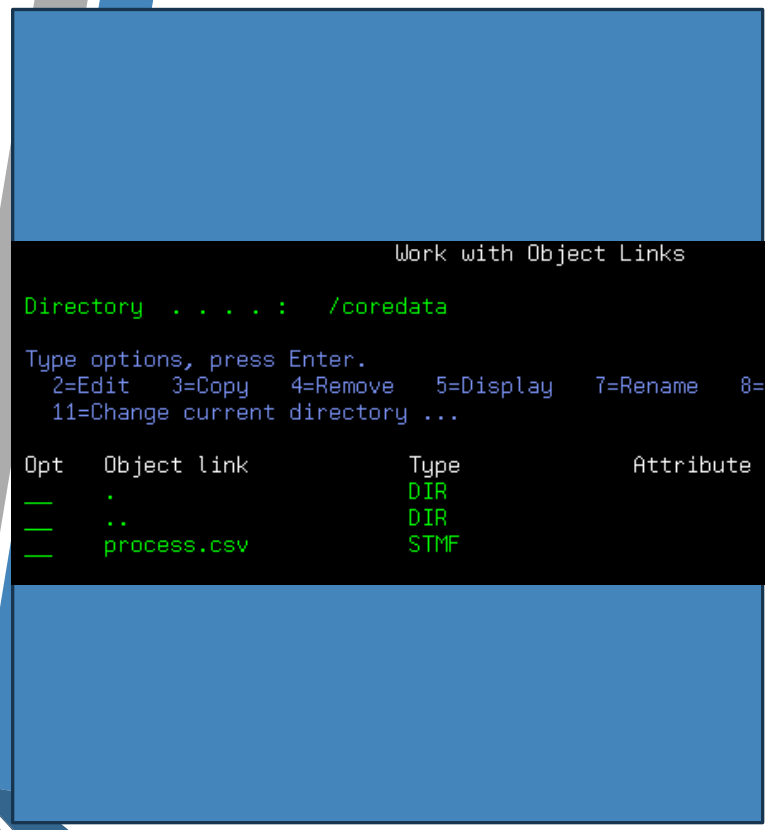
kisco

Section: Use Cases and Problem Solving

Locking Down an IFS Path

kisco

Lock down an IFS path – start authority collection




```
CHGAUTCOL
      OBJ('/coredata/')
      AUTCOLVAL(*OBJINF)
      SUBTREE(*ALL)

STRAUTCOL
      TYPE(*OBJAUTCOL)
```

kisco

Lock down an IFS path – analyze the data

```
SELECT DISTINCT AUTHORIZATION_NAME, PATH_NAME, CHECK_ANY_AUTHORITY,
                AUTHORITY_CHECK_SUCCESSFUL, DETAILED_REQUIRED_AUTHORITY,
                DETAILED_CURRENT_AUTHORITY, AUTHORITY_SOURCE
    FROM QSYS2.AUTHORITY_COLLECTION_FSOBJ
    WHERE PATH_NAME LIKE '/coredata%'
    ORDER BY AUTHORIZATION_NAME, PATH_NAME;
```

Authorization Name	Path Name	Check Any Authority	Authority Check Successful	Detailed Required Authority	Detailed Current Authority	Authority Source
QSECOFR	/coredata	0	1	*OBJOPR *READ	*OWNER *OBJEXI...	USER *ALLOBJ
QSECOFR	/coredata	0	1	*OBJOPR *EXECUTE	*OWNER *OBJEXI...	USER *ALLOBJ
QSECOFR	/coredata/process.csv	0	1	*OBJOPR *READ	*OWNER *OBJEXI...	USER *ALLOBJ
SUPERUSER	/coredata	0	1	*OBJOPR *EXECUTE	*OBJEXIST *OBJ...	PUBLIC
SUPERUSER	/coredata	0	1	*OBJOPR *READ	*OBJEXIST *OBJ...	PUBLIC
SUPERUSER	/coredata/process.csv	0	1	*OBJOPR *ADD *DLT *UPD	*OBJEXIST *OBJ...	PUBLIC
SUPERUSER	/coredata/process.csv	0	1	*OBJOPR *READ	*OBJEXIST *OBJ...	PUBLIC

kisco

AUTHORITY_COLLECTION_FSOBJ vs IFS

- New service – AUTHORITY_COLLECTION_IFS
- Similar to _FSOBJ but simpler/clearer

AUTHORITY_COLLECTION_FSOBJ:

USER_NAME	CHECK_TIMESTAMP	PATH_NAME	DETAILED_REQUIRED_AUTHORITY	DETAILED_CURRENT_AUTHORITY
TIMMR	2025-05-07 08:59:31.375538	/home/testnav/dir090	*OBJOPR *EXECUTE	*OBJEXIST *OBJMGT *OBJALTER *OBJREF *OBJ
TIMMR	2025-05-07 08:59:31.375548	/home/testnav/dir090	*OBJOPR *READ	*OBJEXIST *OBJMGT *OBJALTER *OBJREF *OBJ
TIMMR	2025-05-07 08:59:29.036798	/home/testnav/dir010	*OBJOPR *EXECUTE	*OBJEXIST *OBJMGT *OBJALTER *OBJREF *OBJ
TIMMR	2025-05-07 08:59:29.036809	/home/testnav/dir010	*OBJOPR *READ	*OBJEXIST *OBJMGT *OBJALTER *OBJREF *OBJ

AUTHORITY_COLLECTION_IFS:

USER_NAME	CHECK_TIMESTAMP	PATH_NAME	DETAILED_REQUIRED_AUTHORITY	DETAILED_CURRENT_AUTHORITY
TIMMR	2025-05-07 08:59:31.375538	/home/testnav/dir090	*X	*OBJEXIST *OBJMGT *OBJALTER *OBJREF *R *W *X
TIMMR	2025-05-07 08:59:31.375548	/home/testnav/dir090	*R	*OBJEXIST *OBJMGT *OBJALTER *OBJREF *R *W *X
TIMMR	2025-05-07 08:59:29.036798	/home/testnav/dir010	*X	*OBJEXIST *OBJMGT *OBJALTER *OBJREF *R *W *X
TIMMR	2025-05-07 08:59:29.036809	/home/testnav/dir010	*R	*OBJEXIST *OBJMGT *OBJALTER *OBJREF *R *W *X

kisco

Lock down an IFS path – secure the path

Grant the required private authority for SUPERUSER:

```
CHGAUT OBJ('/coredata') USER(SUPERUSER) DTAAUT(*RX)
CHGAUT OBJ('/coredata/process.csv') USER(SUPERUSER) DTAAUT(*RW)
```

Lock out *PUBLIC for the whole subtree:

```
CHGAUT OBJ('/coredata') USER(*PUBLIC) DTAAUT(*EXCLUDE)
       OBJAUT(*NONE) SUBTREE(*ALL)
```

kisco

Lock down an IFS path – review the results

```
SELECT DISTINCT AUTHORIZATION_NAME, PATH_NAME, CHECK_ANY_AUTHORITY,
                AUTHORITY_CHECK_SUCCESSFUL, DETAILED_REQUIRED_AUTHORITY,
                DETAILED_CURRENT_AUTHORITY, AUTHORITY_SOURCE
    FROM QSYS2.AUTHORITY_COLLECTION_FSOBJ
    WHERE PATH_NAME LIKE '/coredata%'
          AND CHECK_TIMESTAMP > '2024-02-05 19:00:00'
    ORDER BY AUTHORIZATION_NAME, PATH_NAME;
```

Authorization Name	Path Name	Authority Check Successful	Detailed Required Authority	Detailed Current Authority	Authority Source
SUPERUSER	/coredata	1	*OBJOPR *EXECUTE	*OBJOPR *READ *EXECUTE	USER PRIVATE
SUPERUSER	/coredata/process.csv	1	*OBJOPR *READ	*OBJOPR *READ *ADD *DLT *UPD	USER PRIVATE
SUPERUSER	/coredata/process.csv	1	*OBJOPR *ADD *DLT *UPD	*OBJOPR *READ *ADD *DLT *UPD	USER PRIVATE

kisco

New IFS View in IBM i 7.6 and 7.5 TR9!!!

```
142 -- New IFS view provided in 7.6 and 7.5 TR 9 that eliminates unnecessary records and presents authorities in IFS terminology
143 --
144 SELECT *
145    FROM QSYS2.AUTHORITY_COLLECTION_ifs
146    WHERE upper(path_name) = '/PAYROLL_UPLOAD';
147 stop;
148
```

Authorization Name AUTHORIZATION_NAME	Check Timestamp CHECK_TIMESTAMP	Path Name PATH_NAME	System Object Type SYSTEM_OBJECT_TYPE	Detailed Required Authority DETAILED_REQUIRED_AUTHORITY	Detailed Current Authority DETAILED_CURRENT_AUTHORITY
DEVELOPER	2024-03-28 18:09:10.622228	/payroll_upload	*DIR	*R	*EXCLUDE
DAWNM	2024-01-17 10:18:55.562835	/payroll_upload	*DIR	*OBJMGT	*OBJEXIST *OBJMGT *OBJALTER *OBJREF *R *W *X
LDB	2023-08-12 09:14:33.333951	/payroll_upload	*DIR	*X	*OBJEXIST *OBJMGT *OBJALTER *OBJREF *R *W *X
LDB	2025-03-02 12:21:19.383052	/payroll_upload	*DIR	*R	*OBJEXIST *OBJMGT *OBJALTER *OBJREF *R *W *X
LDB	2025-03-02 12:15:41.817435	/payroll_upload	*DIR	*OBJMGT	*OBJEXIST *OBJMGT *OBJALTER *OBJREF *R *W *X

kisco

Section: Use Cases and Problem Solving

Who is using a Netserver fileshare?

kisco

Who is using a Netserver fileshare?

- Enable authority collection on the shared directory:
  - `CHGAUTCOL OBJ('/home') AUTCOLVAL(*OBJINF) SUBTREE(*ALL)`
- Review AC data for that path (Netserver access handled by QZLSFILE% jobs)

```
SELECT CHECK_TIMESTAMP, AUTHORIZATION_NAME, JOB_NAME, PATH_NAME
    FROM QSYS2.AUTHORITY_COLLECTION_FSOBJ
    WHERE JOB_NAME LIKE 'QZLSFILE%'
        AND UPPER(PATH_NAME) LIKE '/HOME/%';
```

CHECK_TIMESTAMP	AUTHORIZATION_NAME	JOB_NAME	PATH_NAME
2024-08-30 09:47:16.031475	STEVE	QZLSFILET	/home/tlk/applyPatch
2024-08-30 09:47:16.032840	STEVE	QZLSFILET	/home/tlk/mg.guix.jar
2024-08-30 09:47:21.242320	STEVE	QZLSFILET	/home/tlk/WEB-INF
2024-08-30 09:47:21.243178	STEVE	QZLSFILET	/home/tlk/WEB-INF/lib
2024-08-30 09:47:33.666634	STEVE	QZLSFILET	/home/tlk/WEB-INF/lib

kisco

Section: Use Cases and Problem Solving

Who is using an object in a particular library?

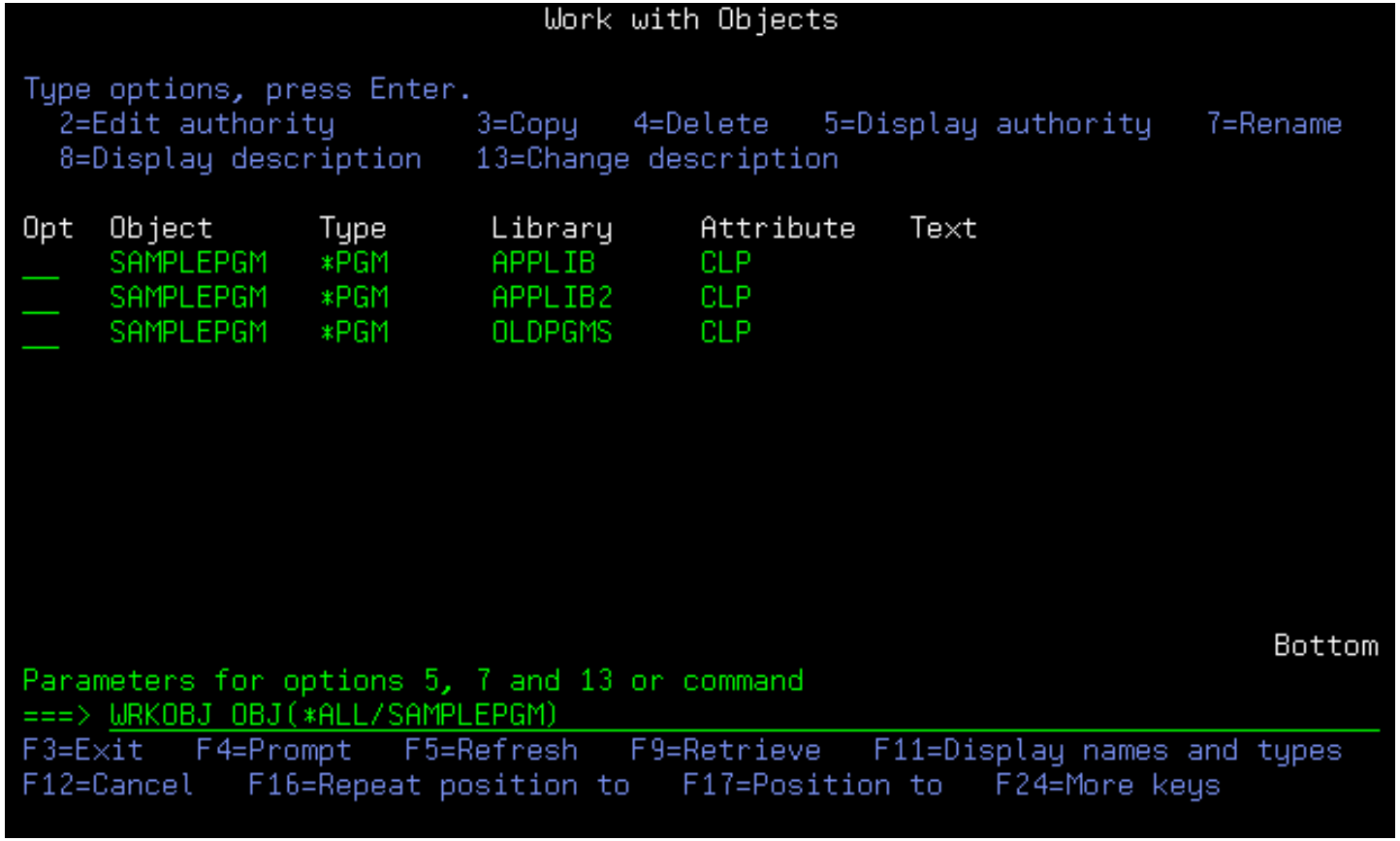
Who is using an object in a particular library?

Multiple copies of a program or file in various libraries

- Separate subsidiary data libraries
- Different versions of programs
- Old copies of objects

The object authority may not even be of interest, but we can still leverage authority collection!

Who is using an object in a particular library?



Who is using an object in a particular library?

Turn on authority collection for all 3 copies of the program:

```
CHGAUTCOL OBJ('/qsys.lib/applib.lib/samplepgm.pgm') AUTCOLVAL(*OBJINF)
CHGAUTCOL OBJ('/qsys.lib/applib2.lib/samplepgm.pgm') AUTCOLVAL(*OBJINF)
CHGAUTCOL OBJ('/qsys.lib/oldpgms.lib/samplepgm.pgm') AUTCOLVAL(*OBJINF)
```

kisco

Who is using an object in a particular library?

Check the collected data after some time:

```
SELECT DISTINCT AUTHORIZATION_NAME, SYSTEM_OBJECT_SCHEMA, JOB_NAME,
                JOB_USER, JOB_NUMBER, "CURRENT_USER"
    FROM QSYS2.AUTHORITY_COLLECTION_LIBRARIES
    WHERE SYSTEM_OBJECT_NAME = 'SAMPLEPGM';
```

Authorization Name	System Object Schema	Job Name	Job User	Job Number	Current User
SRIEDMUE	APPLIB2	QTFTP00237	QTCP	105631	SRIEDMUE
QSYSOPR	OLDPGMS	END_OF_DAY	QSYSOPR	119287	QSYSOPR
SUPERUSER	APPLIB	QPADEVD2	SUPERUSER	119168	SUPERUSER

kisco

Section: Use Cases and Problem Solving

Who is using a system command like ENDSBS?

kisco

Who is using a system command like ENDSBS?

Turn on authority collection for the command:

```
CHGAUTCOL OBJ('/qsys.lib/endsbs.cmd') AUTCOLVAL(*OBJINF)
STRAUTCOL TYPE(*OBJAUTCOL)
```

Check the data after some time:

```
SELECT DISTINCT AUTHORIZATION_NAME, SYSTEM_OBJECT_SCHEMA, JOB_NAME,
                JOB_USER, JOB_NUMBER, "CURRENT_USER"
    FROM QSYS2.AUTHORITY_COLLECTION_LIBRARIES
    WHERE SYSTEM_OBJECT_NAME = 'SAMPLEPGM';
```

AUTHORIZATION_NAME	System Object Name	Check Any Authority	Authority Check Successful	Authority Source	Job Name
QSECOFR	ENDSBS	0	1	USER *ALLOBJ	QPADEV005H
SYSOWNER	ENDSBS	1	1	USER *ALLOBJ	DETAIL
SYSOWNER	ENDSBS	0	1	USER *ALLOBJ	CLEANJSMVS
SYSOWNER	ENDSBS	0	1	USER *ALLOBJ	DETAIL
SYSOWNER	ENDSBS	0	1	USER *ALLOBJ	CLEANJSHYB
BACKUP	ENDSBS	0	1	GROUP PRIVATE	TOTSYSBUG1

kisco

Section: Use Cases and Problem Solving

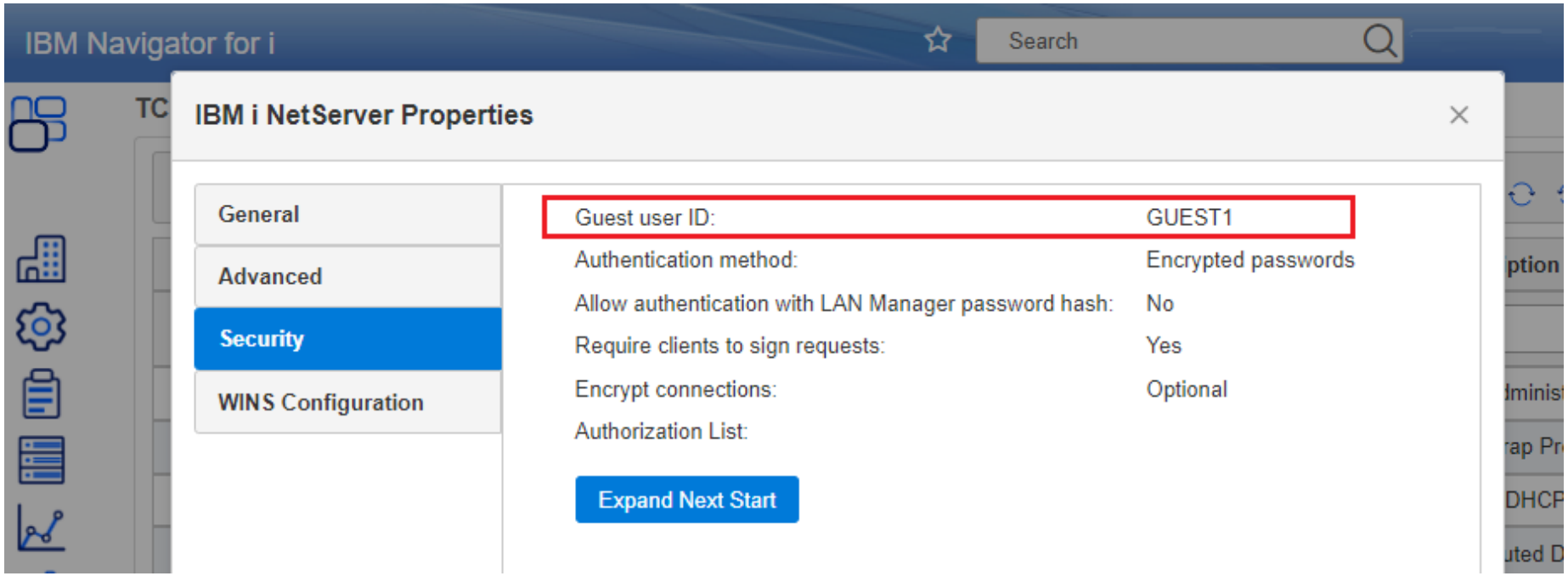
Research Netserver "guest" usage

kisco

Research Netserver “guest” usage

Check for a Netserver guest user ID using NewNav:

Network > Servers > TCP/IP Servers > Netserver > Properties > Security



Section: Use Cases and Problem Solving

Research QAUDJRN authority failures – native QSYS objects

Research AF entries in the audit journal - QSYS

```
SELECT USER_NAME, QUALIFIED_JOB_NAME, VIOLATION_TYPE_DETAIL,
       OBJECT_LIBRARY, OBJECT_NAME, OBJECT_TYPE
   FROM TABLE (
          SYSTOOLS.AUDIT_JOURNAL_AF() );
```

USER_NAME	QUALIFIED_JOB_NAME	VIOLATION_TYPE_DETAIL	OBJECT_LIBRARY	OBJECT_NAME	OBJECT_TYPE
PROFILEX	317331/PROFILEX/CSASN01CLX	Not authorized to object	EXTSAVF	SAVEQTEMP	*FILE
PROFILEX	317331/PROFILEX/CSASN01CLX	Not authorized to object	EXTSAVF	SAVEQTEMP	*FILE
PROFILEX	317331/PROFILEX/CSASN01CLX	Not authorized to object	EXTSAVF	SAVEQTEMP	*FILE
PROFILEX	317331/PROFILEX/CSASN01CLX	Not authorized to object	EXTSAVF	SAVEQTEMP	*FILE
PROFILEX	317331/PROFILEX/CSASN01CLX	Not authorized to object	EXTSAVF	SAVEQTEMP	*FILE
PROFILEX	317331/PROFILEX/CSASN01CLX	Not authorized to object	EXTSAVF	SAVEQTEMP	*FILE
PROFILEX	317331/PROFILEX/CSASN01CLX	Not authorized to object	EXTSAVF	SAVEQTEMP	*FILE

Research AF entries in the audit journal - QSYS

Turn on authority collection for this file:

```
CHGAUTCOL OBJ('/qsys.lib/extsavf.lib/saveqtemp.file') AUTCOLVAL(*OBJINF)
STRAUTCOL TYPE(*OBJAUTCOL)
```

Review the authority collection entries:

```
SELECT AUTHORIZATION_NAME, SYSTEM_OBJECT_NAME, SYSTEM_OBJECT_SCHEMA,
       SYSTEM_OBJECT_TYPE, AUTHORITY_CHECK_SUCCESSFUL,
       DETAILED_REQUIRED_AUTHORITY, DETAILED_CURRENT_AUTHORITY
   FROM QSYS2.AUTHORITY_COLLECTION_LIBRARIES
   WHERE SYSTEM_OBJECT_NAME='SAVEQTEMP'
   ORDER BY CHECK_TIMESTAMP;
```

AUTHORIZATION_NAME	System Object Name	System Object Schema	System Object Type	Authority Check Successful	Detailed Required Authority	Detailed Current Authority
PROFILEX	SAVEQTEMP	EXTSAVF	*FILE	0	*OBJOPR	*EXCLUDE
PROFILEX	SAVEQTEMP	EXTSAVF	*FILE	0	*OBJOPR *READ *ADD *EXECUTE	*EXCLUDE

kisco

Research AF entries in the audit journal - QSYS

Grant the required authority to the user profile:

```
GRTOBJAUT OBJ(EXTSAVF/SAVEQTEMP)
          OBJTYPE(*FILE)
          USER(PROFILEX)
          AUT(*OBJOPR *READ *ADD *EXECUTE)
```

kisco

Section: Use Cases and Problem Solving

Research QAUDJRN authority failures – IFS files

kisco

Research AF entries in the audit journal - IFS

```
SELECT USER_NAME,
       QUALIFIED_JOB_NAME,
       VIOLATION_TYPE_DETAIL,
       OBJECT_TYPE,
       PATH_NAME
    FROM TABLE (
            SYSTOOLS.AUDIT_JOURNAL_AF() );
```

USER_NAME	QUALIFIED_JOB_NAME	VIOLATION_TYPE_DETAIL	OBJECT_TYPE	PATH_NAME
QTMHHTTP	125510/QTMHHTTP/OLRDEV	Not authorized to object	*STMF	/web/development/olrdev/logs/access/olraccess.Q124010100
QTMHHTTP	125510/QTMHHTTP/OLRDEV	Not authorized to object	*STMF	/web/development/olrdev/logs/access/olraccess.Q124010200
QTMHHTTP	125510/QTMHHTTP/OLRDEV	Not authorized to object	*STMF	/web/development/olrdev/logs/access/olraccess.Q124010300
QTMHHTTP	125510/QTMHHTTP/OLRDEV	Not authorized to object	*STMF	/web/development/olrdev/logs/access/olraccess.Q124010400
QTMHHTTP	125510/QTMHHTTP/OLRDEV	Not authorized to object	*STMF	/web/development/olrdev/logs/access/olraccess.Q124010500
QTMHHTTP	125510/QTMHHTTP/OLRDEV	Not authorized to object	*STMF	/web/development/olrdev/logs/access/olraccess.Q124010600
QTMHHTTP	125510/QTMHHTTP/OLRDEV	Not authorized to object	*STMF	/web/development/olrdev/logs/access/olraccess.Q124010700
QTMHHTTP	125510/QTMHHTTP/OLRDEV	Not authorized to object	*STMF	/web/development/olrdev/logs/access/olraccess.Q124010800
QTMHHTTP	125510/QTMHHTTP/OLRDEV	Not authorized to object	*STMF	/web/development/olrdev/logs/access/olraccess.Q124010900

kisco

Research AF entries in the audit journal - IFS

Turn on authority collection for all files in this directory:

```
CHGAUTCOL OBJ('/web/development/olrdev/logs/access/*') AUTCOLVAL(*OBJINF)
```

kisco

Research AF entries in the audit journal - IFS

```
SELECT DISTINCT AUTHORIZATION_NAME, PATH_NAME, AUTHORITY_CHECK_SUCCESSFUL,
    DETAILED_REQUIRED_AUTHORITY, DETAILED_CURRENT_AUTHORITY, AUTHORITY_SOURCE
    FROM QSYS2.AUTHORITY_COLLECTION_FSOBJ
    WHERE PATH_NAME LIKE '/web/development/olrdev/logs/access/%'
          AND AUTHORITY_CHECK_SUCCESSFUL = 0
    ORDER BY AUTHORIZATION_NAME, PATH_NAME;
```

Authorization Name	Path Name	Authority Check Successful	Detailed Required Authority	Detailed Current Authority	Authority Source
QTMHHTTP	/web/development/olrdev/logs/access/olraccess.Q124010100	0	*OBJEXIST	*OWNER *OBJOPR *READ *ADD *DLT *UPD *EXECUTE	USER OWNERSHIP
QTMHHTTP	/web/development/olrdev/logs/access/olraccess.Q124010200	0	*OBJEXIST	*OWNER *OBJOPR *READ *ADD *DLT *UPD *EXECUTE	USER OWNERSHIP
QTMHHTTP	/web/development/olrdev/logs/access/olraccess.Q124010300	0	*OBJEXIST	*OWNER *OBJOPR *READ *ADD *DLT *UPD *EXECUTE	USER OWNERSHIP
QTMHHTTP	/web/development/olrdev/logs/access/olraccess.Q124010400	0	*OBJEXIST	*OWNER *OBJOPR *READ *ADD *DLT *UPD *EXECUTE	USER OWNERSHIP
QTMHHTTP	/web/development/olrdev/logs/access/olraccess.Q124010500	0	*OBJEXIST	*OWNER *OBJOPR *READ *ADD *DLT *UPD *EXECUTE	USER OWNERSHIP
QTMHHTTP	/web/development/olrdev/logs/access/olraccess.Q124010600	0	*OBJEXIST	*OWNER *OBJOPR *READ *ADD *DLT *UPD *EXECUTE	USER OWNERSHIP
QTMHHTTP	/web/development/olrdev/logs/access/olraccess.Q124010700	0	*OBJEXIST	*OWNER *OBJOPR *READ *ADD *DLT *UPD *EXECUTE	USER OWNERSHIP
QTMHHTTP	/web/development/olrdev/logs/access/olraccess.Q124010800	0	*OBJEXIST	*OWNER *OBJOPR *READ *ADD *DLT *UPD *EXECUTE	USER OWNERSHIP

*OBJEXIST
*OBJEXIST

Grant the required authority to the user profile:

```
CHGAUT OBJ('/web/development/olrdev/logs/access/*')
       USER(QTMHHTTP) OBJAUT(*OBJEXIST)
```

kisco

Section: Use Cases and Problem Solving

Investigate FTP process activity

kisco

Investigate FTP Processes and Activity

Turn on user-based authority collection for user profiles that are used in your FTP jobs (check the "current user" of the QTFTPxxxxx jobs)

After the FTP process runs, you can see what objects were used.

kisco

Investigate FTP Processes and Activity

Turn on authority collection for the user profile:

```
STRAUTCOL USRPRF(FTPUSER) INCFSOBJ(*ALL)
```

Check the authority collection entries for this user:

```
SELECT DISTINCT JOB_NAME, SYSTEM_OBJECT_SCHEMA, SYSTEM_OBJECT_NAME,
                SYSTEM_OBJECT_TYPE, CHECK_ANY_AUTHORITY,
                AUTHORITY_CHECK_SUCCESSFUL, REQUIRED_AUTHORITY,
                DETAILED_REQUIRED_AUTHORITY
    FROM QSYS2.AUTHORITY_COLLECTION
    WHERE AUTHORIZATION_NAME = 'FTPUSER'
          AND (ADOPTING_PROGRAM_SCHEMA IS NULL
              OR ADOPTING_PROGRAM_SCHEMA <> 'QSYS')
          AND JOB_NAME LIKE 'QTFTP%'
    ORDER BY SYSTEM_OBJECT_SCHEMA, SYSTEM_OBJECT_NAME;
```

kisco

Investigate FTP Processes and Activity

Job Name	System Object Schema	System Object Name	System Object Type	Check Any Authority	Authority Check Successful	Required Authority	Detailed Required Authority
QTFTP00117	[redacted]	[redacted]	*JOBD	1	1	*ALL	*OWNER *OBJEXIST *OBJ...
QTFTP00117			*JOBD	0	1	*USE	*OBJOPR *READ *EXECUTE
QTFTP00108			*JOBD	0	1	-	*OBJOPR
QTFTP00117			*JOBD	0	1	-	*OBJOPR
QTFTP00108			*JOBD	1	1	*ALL	*OWNER *OBJEXIST *OBJ...
QTFTP00108			*JOBD	0	1	*USE	*OBJOPR *READ *EXECUTE
QTFTP00108			*PGM	0	1	*USE	*OBJOPR *READ *EXECUTE
QTFTP00108			*PGM	0	1	-	*OBJOPR
QTFTP00108			*PGM	0	1	-	*OBJOPR
QTFTP00108			*PGM	0	1	*USE	*OBJOPR *READ *EXECUTE
QTFTP00117			*PGM	0	1	-	*OBJOPR
QTFTP00117			*PGM	0	1	*USE	*OBJOPR *READ *EXECUTE
QTFTP00108			*JOBQ	0	1	*USE	*OBJOPR *READ *EXECUTE
QTFTP00126	QGPL		*JOBD	0	1	-	*OBJOPR
QTFTP00126	QGPL		*JOBD	1	1	*ALL	*OWNER *OBJEXIST *OBJ...
QTFTP00126	QGPL		*USRSPC	1	1	*ALL	*OWNER *OBJEXIST *OBJ...
QTFTP00126	QGPL		*USRSPC	0	1	-	*OBJOPR
QTFTP00126	QSYS		*CMD	0	1	*USE	*OBJOPR *READ *EXECUTE
QTFTP00108	QSYS		*LIB	0	1	-	*EXECUTE

kisco

Section: Use Cases and Problem Solving

What process is accessing a file in the IFS root?

kisco

What process/user is accessing a file in the root?

Users and batch jobs should not be dumping files into the root of the IFS “/”

WRKLNK can tell us that the file has been accessed/changed, but not by which job or user.

Authority collection can help us figure out how the file is getting updated and used, including the job name.

Then, development can update the process to put this file in an appropriate subdirectory instead of the root!

kisco

What process/user is accessing a file in the root?

Find recently-used CSV files in the system root directory:

```
SELECT PATH_NAME, OBJECT_TYPE, CREATE_TIMESTAMP,
       ACCESS_TIMESTAMP, DATA_CHANGE_TIMESTAMP
    FROM TABLE (
        QSYS2.IFS_OBJECT_STATISTICS(
            START_PATH_NAME => '/', SUBTREE_DIRECTORIES => 'NO') )
    WHERE UPPER(PATH_NAME) LIKE '%.CSV'
    ORDER BY ACCESS_TIMESTAMP DESC;
```

PATH_NAME	OBJECT_TYPE	CREATE_TIMESTAMP	ACCESS_TIMESTAMP	DATA_CHANGE_TIMESTAMP
/downloadfile.csv	*STMF	2011-08-04 15:32:04	2024-04-07 06:26:26	2024-04-06 17:01:10

kisco

What process/user is accessing a file in the root?

Turn on authority collection for all of the files in this directory:

```
CHGAUTCOL OBJ('/downloadfile.csv') AUTCOLVAL(*OBJINF)

STRAUTCOL TYPE(*OBJAUTCOL)
```

```
Selection or command
===> CHGAUTCOL OBJ('/downloadfile.csv') AUTCOLVAL(*OBJINF)

F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel   F13=Information Assistant
F23=Set initial menu
Authority collection value changed for 1 objects, not changed for 0 objects.
```

kisco

What process/user is accessing a file in the root?

View the authority collection data:

```
SELECT DISTINCT JOB_NAME, AUTHORIZATION_NAME, PATH_NAME, CHECK_ANY_AUTHORITY,
AUTHORITY_CHECK_SUCCESSFUL, DETAILED_REQUIRED_AUTHORITY,
                DETAILED_CURRENT_AUTHORITY, AUTHORITY_SOURCE
    FROM QSYS2.AUTHORITY_COLLECTION_FSOBJ
    WHERE PATH_NAME =  '/downloadfile.csv'
    ORDER BY AUTHORIZATION_NAME, PATH_NAME;
```

JOB_NAME	JOB_USER	JOB_NUMBER	AUTHORIZATION_NAME	Path Name
OVERNIGHT	BADUSER	246538	BADUSER	/downloadfile.csv

Check Any Authority	Authority Check Successful	Detailed Required Authority	DETAILED_CURRENT_AUTHORITY	AUTHORITY_SOURCE
0	1	*READ	*OBJMGT *READ	USER PRIVATE

Exclude Authority Failures from the Results

```
132 SELECT DISTINCT authorization_name,
133             detailed_required_authority,
134             path_name,
135             current_authority,
136             authority_source,
137             authority_check_successful
138    FROM qsys2.authority_collection_fsobj
139    WHERE path_name = '/payroll_upload';
140 stop;
```

Authorization Name AUTHORIZATION_NAME	Detailed Required Authority DETAILED_REQUIRED_AUTHORITY	Path Name PATH_NAME	Current Authority CURRENT_AUTHORITY	Authority Source AUTHORITY_SOURCE	Authority Check Successful AUTHORITY_CHECK_SUCCESSFUL
JONGRKIM	*OBJMGT	/payroll_upload	*ALL	USER *ALLOBJ	1
DEVELOPER	*OBJOPR *READ	/payroll_upload	*EXCLUDE	PUBLIC	0
DAWNM	*OBJMGT	/payroll_upload	*ALL	USER *ALLOBJ	1
LDB	*OBJOPR *EXECUTE	/payroll_upload	*ALL	USER *ALLOBJ	1

```
144 SELECT DISTINCT authorization_name,
145             detailed_required_authority,
146             path_name,
147             current_authority,
148             authority_source
149    FROM qsys2.authority_collection_fsobj
150    WHERE path_name = '/payroll_upload'
151    and authority_check_successful = '1';
```

Section: Authority Collection Housekeeping

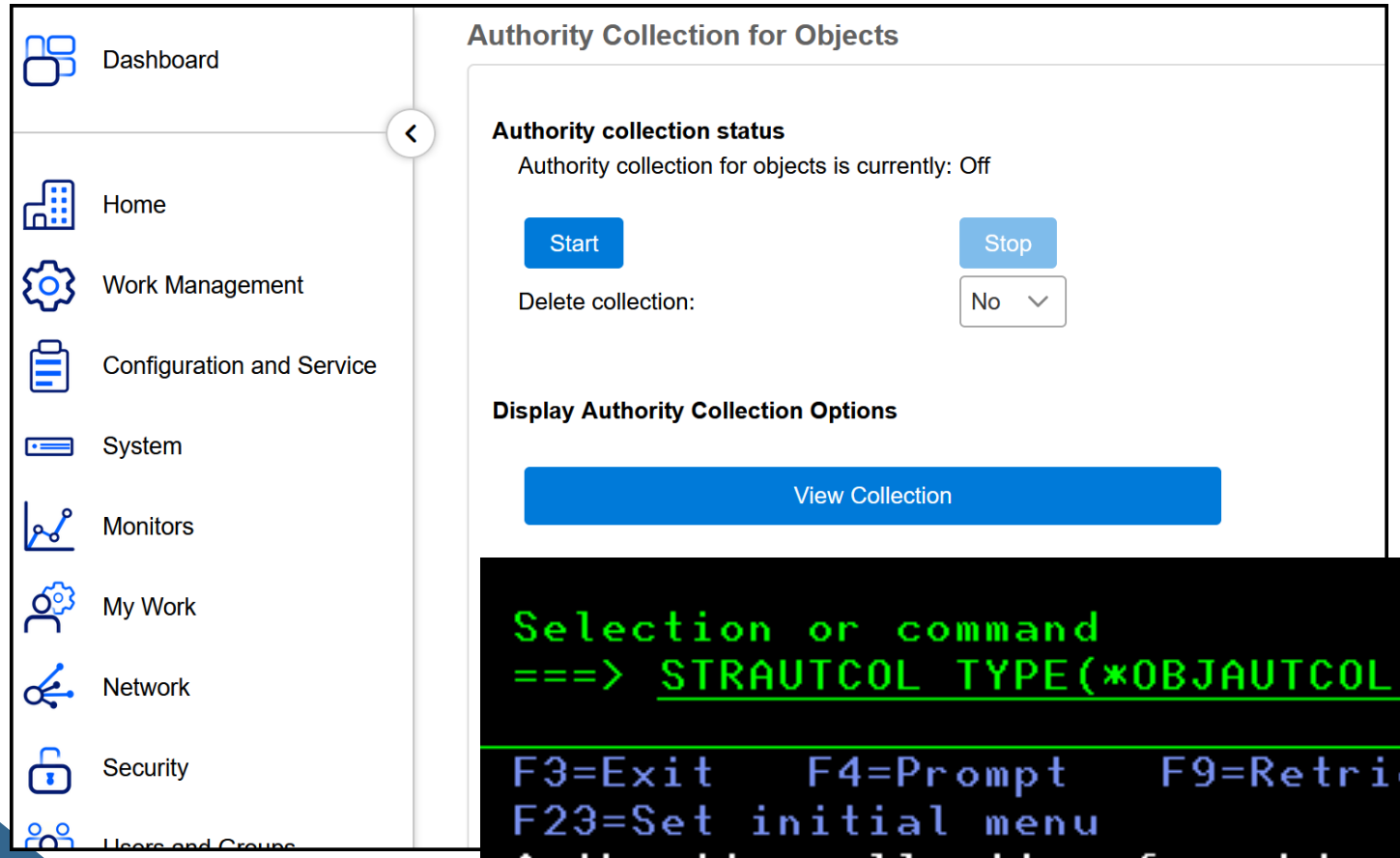
Authority Collection Housekeeping

- Authority Collection global setting
- User profiles with authority collection running
- Native QSYS objects with authority collection enabled
- IFS objects with authority collection enabled
- Alternative methods of checking for active collections
- Deleting authority collection data

kisco

Authority Collection Housekeeping

Authority Collection global setting




```
Selection or command
===> STRAUTCOL TYPE(*OBJAUTCOL) DLTCOL(*NO)

F3=Exit    F4=Prompt    F9=Retrieve    F12=Cancel    F13=In
F23=Set initial menu
Authority collection for objects already started.
```

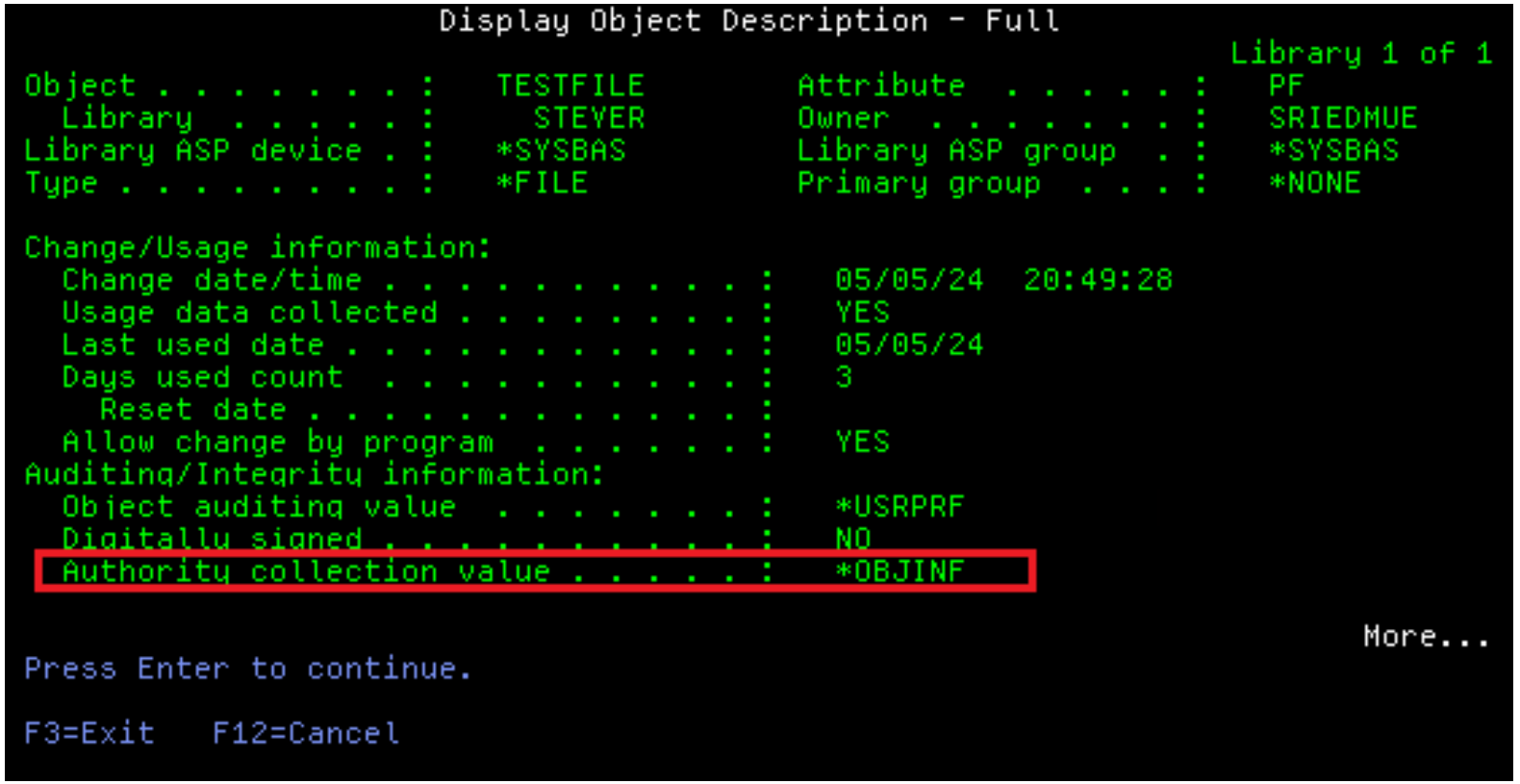
Which user profiles have collections running

Which user profiles have collections running?

```
--user profiles that have authority collection turned on
SELECT *
    FROM QSYS2.USER_INFO
    WHERE AUTHORITY_COLLECTION_ACTIVE = 'YES';


--user profiles with authority collection turned off, but a repository exists
SELECT *
    FROM QSYS2.USER_INFO
    WHERE AUTHORITY_COLLECTION_ACTIVE = 'NO'
          AND AUTHORITY_COLLECTION_REPOSITORY_EXISTS = 'YES';
```

kisco

Which "Native" objects have collection enabled



Which "Native" objects have collection enabled

```
SELECT OBJLIB, OBJNAME, OBJTYPE, OBJATTRIBUTE, AUTHORITY_COLLECTION_VALUE
    FROM TABLE (
            QSYS2.OBJECT_STATISTICS(OBJECT_SCHEMA => 'STEVER',
                                    OBJTYPELIST   => '*FILE',
                                    OBJECT_NAME   => 'TESTFILE') );
```

OBJLIB	OBJNAME	OBJTYPE	OBJATTRIBUTE	AUTHORITY_COLLECTION_VALUE
STEVER	TESTFILE	*FILE	PF	*OBJINF

kisco

Which "Native" objects have collection enabled

```
SELECT DISTINCT SYSTEM_OBJECT_SCHEMA,
                SYSTEM_OBJECT_NAME,
                SYSTEM_OBJECT_TYPE
    FROM QSYS2.AUTHORITY_COLLECTION_LIBRARIES;
```

System Object Schema	System Object Name	System Object Type
SRSAMPLE	PROJECT	*FILE
STEVER	TESTFILE	*FILE
JSMQUAERC	DC@W29	*FILE
JSMQUAERC	JSMDIRECT	*PGM
JSMQUAERC	JSMDRTDTA	*DTAARA
SRSAMPLE	SALARY	*FILE

kisco

An Alternative Method for Checking

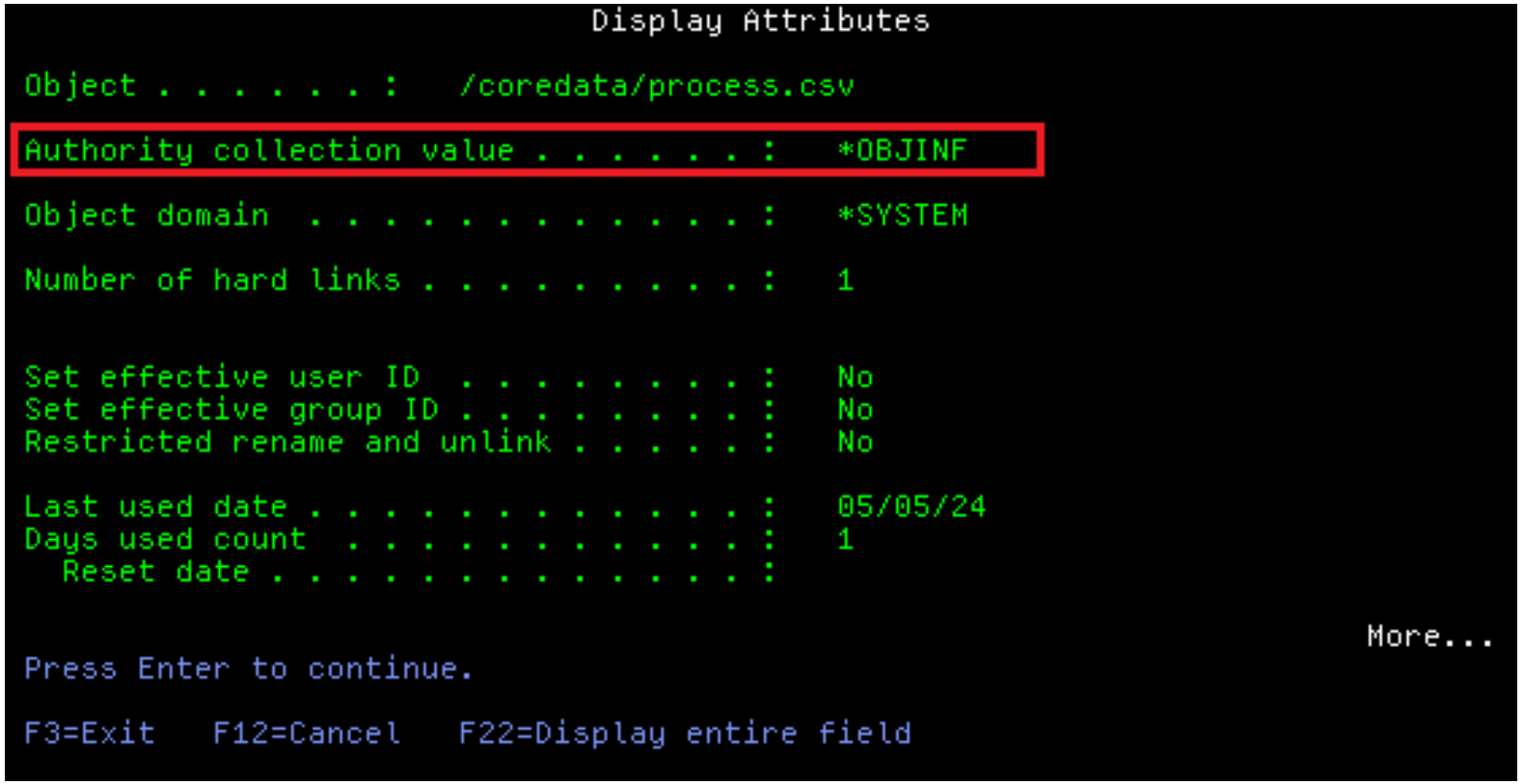
```
--objects having AC repository entries, and the current OBJAUTCOL setting
WITH OBJECTS (LIBNAME, OBJNAME, OBJTYPE) AS (
    SELECT DISTINCT SYSTEM_OBJECT_SCHEMA,
                    SYSTEM_OBJECT_NAME,
                    SYSTEM_OBJECT_TYPE
      FROM QSYS2.AUTHORITY_COLLECTION_LIBRARIES )
          SELECT OBJECTS.LIBNAME, OBJECTS.OBJNAME, OBJECTS.OBJTYPE,
                 AUTHORITY_COLLECTION_VALUE
          FROM OBJECTS, LATERAL (
             SELECT AUTHORITY_COLLECTION_VALUE FROM TABLE (
                 QSYS2.OBJECT_STATISTICS(
                     OBJECT_SCHEMA => OBJECTS.LIBNAME,
                     OBJTYPELIST   => OBJECTS.OBJTYPE,
                     OBJECT_NAME   => OBJECTS.OBJNAME)
             ) );
```

kisco

An Alternative Method for Checking

LIBNAME	OBJNAME	OBJTYPE	AUTHORITY_COLLECTION_VALUE
SRSAMPLE	PROJECT	*FILE	*OBJINF
STEVER	TESTFILE	*FILE	*OBJINF
APPLIB	DC@W29	*FILE	*NONE
APPLIB	JSMDIRECT	*PGM	*NONE
QSYS	ENDSBS	*CMD	*NONE
APPLIB	JSMDRTDTA	*DTAARA	*NONE
SRSAMPLE	SALARY	*FILE	*OBJINF

Which IFS objects have collection enabled



kisco

Which IFS objects have collection enabled

```
SELECT PATH_NAME, OBJECT_TYPE, AUTHORITY_COLLECTION_VALUE
    FROM TABLE (
            QSYS2.IFS_OBJECT_STATISTICS(
                START_PATH_NAME => '/coredata/process.csv') );
```

PATH_NAME	OBJECT_TYPE	AUTHORITY_COLLECTION_VALUE
/coredata/process.csv	*STMF	*OBJINF

kisco

An Alternative Method for Checking

```
--IFS files having AC repository entries, and the current OBJAUTCOL setting
WITH IFSFILES (PATH_NAME) AS (
    SELECT DISTINCT PATH_NAME
      FROM QSYS2.AUTHORITY_COLLECTION_FSOBJ )
    SELECT IFSFILES.PATH_NAME, AUTHORITY_COLLECTION_VALUE
      FROM IFSFILES, LATERAL (
        SELECT AUTHORITY_COLLECTION_VALUE FROM TABLE (
          QSYS2.IFS_OBJECT_STATISTICS(
            START_PATH_NAME     => IFSFILES.PATH_NAME,
            SUBTREE_DIRECTORIES => 'NO')
                        ) );
```

kisco

Deleting authority collection data

Deleting entries for a user profile:

```
ENDAUTCOL USRPRF(SRIEDMUE)
DLTAUTCOL TYPE(*USRPRF) USRPRF(SRIED)
```

Deleting entries for an entire IFS subtree:

```
ENDAUTCOL TYPE(*OBJAUTCOL)
DLTAUTCOL TYPE(*OBJ) OBJ('/home/sried/') SUBTREE(*ALL)
```

Deleting entries for a specific object:

```
ENDAUTCOL TYPE(*OBJAUTCOL)
DLTAUTCOL TYPE(*OBJ) OBJ('/QSYS.LIB/SRIED.LIB/QCLSRC.FILE')
```

kisco

What's the Dash in the Authorization_Name?

```
103 SELECT DISTINCT authorization_name,
104             system_object_name,
105             detailed_required_authority,
106             authority_source
107    FROM qsys2.authority_collection_libraries
108    WHERE system_object_name = 'ALLOBJUSRS'
109        AND adopting_program_owner IS null
110        AND check_any_authority = '0';
111 stop;
```

Authorization Name AUTHORIZATION_NAME	System Object Name SYSTEM_OBJECT_NAME	Detailed Required Authority DETAILED_REQUIRED_AUTHORITY	Authority Source AUTHORITY_SOURCE
DEVELOPER	ALLOBJUSRS	*READ	USER PRIVATE
QSECOFR	ALLOBJUSRS	*OBJOPR *READ	USER *ALLOBJ
SCOTTF	ALLOBJUSRS	*OBJOPR	USER *ALLOBJ
-	ALLOBJUSRS	*OBJOPR	GROUP PRIVATE
CWOODBURY	ALLOBJUSRS	*READ *DLT	USER *ALLOBJ
CWOODBURY	ALLOBJUSRS	*ADD	USER *ALLOBJ
CWOODBURY	ALLOBJUSRS	*OBJOPR *READ	USER *ALLOBJ
CWOODBURY	ALLOBJUSRS	*OWNER	USER *ALLOBJ

'-' for the Authorization name indicates access by a profile that's been deleted

kisco

Thank you for attending the session!

https://www.common.org/n2i/home

Contact info:

- steve@kisco.com
- Steve's "Gists" https://gist.github.com/sriedmue79

kisco

For More Information

IBM i Services

- https://www.ibm.com/support/pages/node/1119123

IBM Tutorials

- https://www.ibm.com/support/pages/ibm-i-tutorials-demos-and-sql-examples-0

IBM i Security Reference – PDF

- https://www.ibm.com/docs/en/ssw_ibm_i_76/pdf/sc415302.pdf
  - Chapter 10 – Authority Collection

IBM i Security Administration and Compliance, 3nd edition, by Carol Woodbury, 2020 available from Amazon or MCPress Bookstore

Mastering IBM i Security – A Step by Step Approach by Carol Woodbury, 2022 available from Amazon or MCPress Bookstore

Whitepaper: Securing IBM i: A Dual Responsibility

Articles by Carol Woodbury on mcpressonline.com and KiscoU