

2012

IBM Power Systems Technical University

featuring IBM PureSystems

Enabling the infrastructure for smarter computing

Security Considerations for IBM i Application Development

Jeffrey Uehling
IBM i Security Development
uehling@us.ibm.com



Security for Application Development - Outline

- **Security Documentation**
 - Security related suggestions for application development documentation

- **Security Considerations**
 - Object ownership and public authority
 - Private authority and Authorization lists

- **System integrity Considerations**
 - Object domain, program state and security levels
 - Trojan horse considerations
 - Parameter validation, User Domain Objects

- **Application Security Techniques**
 - Adopted authority
 - Profile Swap

Application Documentation

- When designing a new application, design documentation is a requirement in order to provide the necessary security information to a system administrator
 - What objects in the application contain “sensitive” data (these objects may need extra attention in order to **secure the data**)
 - What objects in the application contain “sensitive” data (the system admin may want to “**AUDIT**” access to the objects)
 - What user profiles are used to **own** the application (this may be needed for audit purposes)
 - Are there authorization lists used to provide security for the data objects. The system admin will need this information in order to add/remove users for application access

Application Documentation continued...

- Application documentation is necessary if, prior to application install/restore, objects need to be created by the system administrator
 - Does an “application” user profile(s) need to be created in order to establish correct ownership of objects?
 - Does an “application” authorization list(s) need to be created in order to secure application objects?
- An entry point into the application should be defined and documented. This entry point could be a CL Command or a Program that is called to “start” the application
 - This entry point can then be used to provide access to the application by granting users authority to “run” the application. An authorization list would be a good mechanism to secure the entry point into the application.

Security Overview & Considerations

Security Overview

- In this section of the presentation, we will discuss:
 1. Object Ownership
 2. Public Authority
 3. Private Authority
 4. Authorization lists

Object Ownership Overview

Object Ownership

- Every object (*FILE, *PGM, *DIR, *STMF, etc) on the system has an owner
- Ownership assigned when the object is initially created
 - By default, the process user profile owns the object
 - The owner is granted ***ALL** authority to the created object
 - Options exist to assign ownership to the primary group profile of the process user profile (not discussed in this presentation)
 - Ownership can be changed via CHGOBJOWN or CHGOWN

Object ownership continued...

- Signon and create an object. Process user owns the object.
- CRTPF FILE(PAYLIB/PAYROLL) RCDLEN(80) AUT(*EXCLUDE)
- DSPOBJAUT OBJ(PAYLIB/PAYROLL) OBJTYPE(*FILE)

```
Session A - [24 x 80]
File Edit View Communication Actions Window Help
Display Object Authority
Object . . . . . : PAYROLL      Owner . . . . . : UEHLING
Library . . . . . : PAYLIB     Primary group . . . . . : *NONE
Object type . . . . . : *FILE   ASP device . . . . . : *SYSBAS
Object secured by authorization list . . . . . : *NONE

User      Group      Object
*PUBLIC   *EXCLUDE  Authority
UEHLING   *ALL

Bottom

Press Enter to continue.
F3=Exit  F11=Display detail object authorities  F12=Cancel  F17=Top
F18=Bottom
(C) COPYRIGHT IBM CORP. 1980, 2009.
Mâ a 01/001
I902 - Session successfully started
```

Object owner

Owner authority

Object Ownership Considerations

- What user profile will own the application objects?
 - Creating an “application” user profile is recommended to own all application objects
 - A product “build” procedure should exist to set object ownership during application development & build
 - The “application” user profile should be created during application install via the pre-install exit program of RSTLICPGM (or created by the system administrator prior to restore of the application on the system)
 - CRTUSRPRF USRPRF(PAYROLL) PWD(*NONE) SPCAUT(*NONE)
 - This provides support to “assign ownership” during the application restore process

Object Ownership Considerations continued...

- What user profile should own objects created (by the application) when the application is run by an end user?
 - Most likely **NOT** the user running the application...
 - If the process user owns the objects, they will, by default, have ***ALL** authority to the object which is a security exposure as they can change all data or delete the object
 - Use the CHGOBJOWN or CHGOWN interface, within your application, to change object ownership at run-time to the application user profile

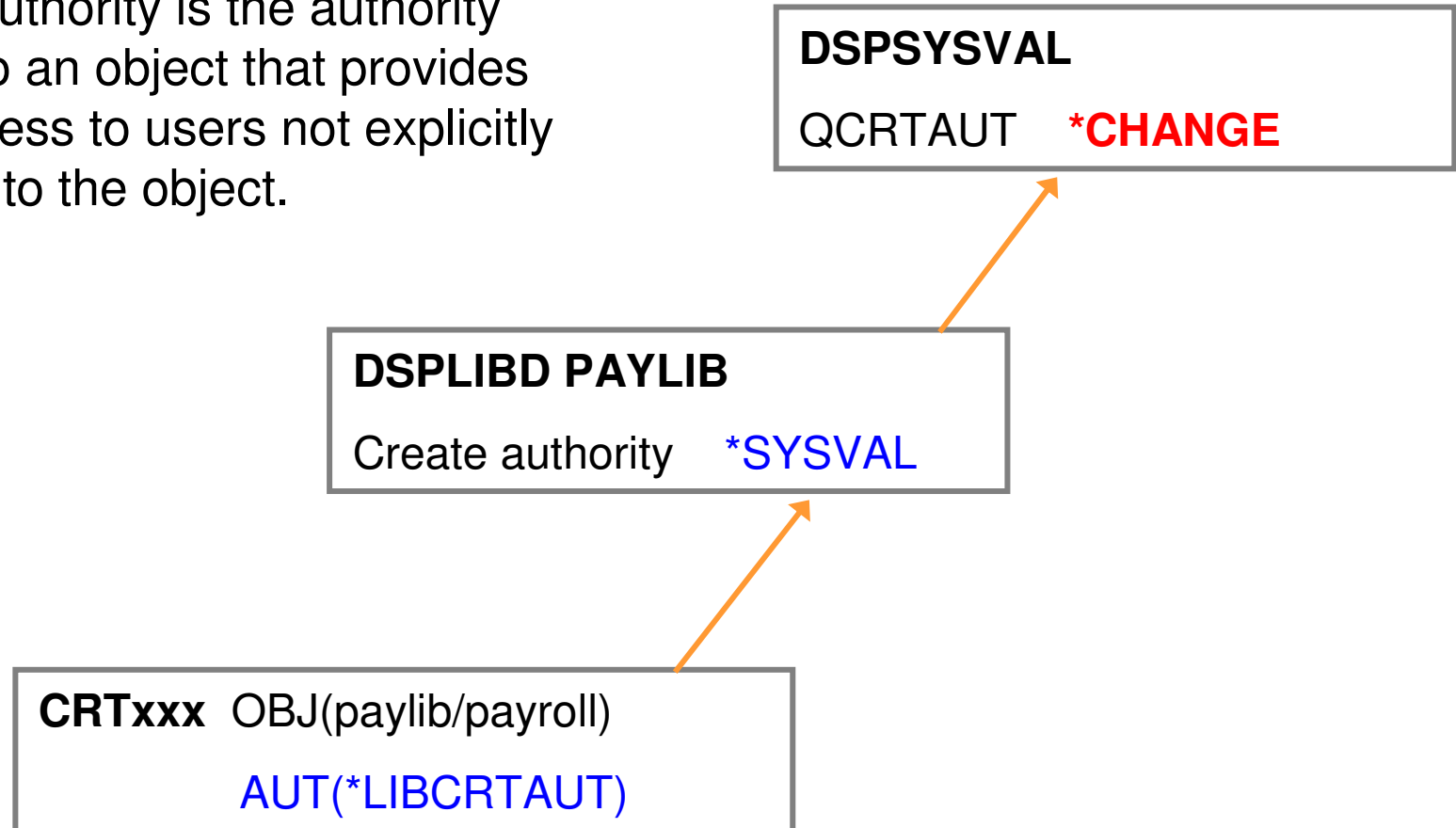
Public Authority Overview

Object Public Authority

- Every object (*FILE, *PGM, *DIR, *STMF, etc) on the system has a Public Authority Value
- Public Authority is assigned when the object is initially created
 - Public authority is set to the value specified on the AUT parm of the create interface (CL Command or API, e.g. CRTPF)
 - For IFS, inherited from the directory (default)
 - Public Authority can be changed via GRTOBJAUT or CHGAUT

Where does *PUBLIC authority come from?

*PUBLIC authority is the authority assigned to an object that provides default access to users not explicitly authorized to the object.



NOTE: The system default, because of history, is *CHANGE

“Short-cut” Authorities

Authority is required to access every object on the system

	*OBJOPR	*OBJMGT	*OBJEXIST	*OBJALTER	*OBJREF	*READ	*ADD	*UPD	*DLT	*EXECUTE
*ALL	X	X	X	X	X	X	X	X	X	X
*CHANGE	X					X	X	X	X	X
*USE	X					X				X
*EXCLUDE										

Object Authorities

*OBJOPR – Object Operational	Look at the description of an object and use the object as determined by the data authorities the user has.
*OBJMGT – Object Management	Move or rename an object or add members to database files. Superset of *OBJALTER and *OBJREF
*OBJEXIST – Object Existence	Change ownership and delete the object, free storage for the object, perform save and restore operations
*OBJALTER – Object Alter	Add, clear, initialize and reorganize members of database files, after and add attributes of database files, add and remove triggers, change attributes of SQL packages
*OBJREF – Object Reference	Specify database file as the parent in a referential constraint
*AUTLMGT – Authorization List Management	Add and remove users and their authorities from an authorization list.

Data Authorities - Defined

*READ	Display the contents of an object, such as viewing the records in a file
*ADD	Add entries to an object, such as adding messages to a message queue, or records to a file
*UPD (Update)	Change entries in an object, such as changing records in a file
*DLT (Delete)	Remove entries from an object, such as removing messages from a message queue or deleting records from a file
*EXECUTE	Run a program or search a library or directory

Public Authority

- CRTPF FILE(PAYLIB/PAYROLL) RCDLEN(80) **AUT(*EXCLUDE)**
- DSPOBJAUT OBJ(PAYLIB/PAYROLL) OBJTYPE(*FILE)

```
Session A - [24 x 80]
File Edit View Communication Actions Window Help
Display Object Authority
Object . . . . . : PAYROLL      Owner . . . . . : UEHLING
Library . . . . . : PAYLIB      Primary group . . . . . : *NONE
Object type . . . . . : *FILE      ASP device . . . . . : *SYSBAS
Object secured by authorization list . . . . . : *NONE

User      Group      Object
*PUBLIC   UEHLING   Authority
*EXCLUDE *ALL

Bottom

Press Enter to continue.

F3=Exit  F11=Display detail object authorities  F12=Cancel  F17=Top
F18=Bottom
(C) COPYRIGHT IBM CORP. 1980, 2009.
Mâ a 01/001
I902 - Session successfully started
```

Public authority

Public Authority Considerations

- What public authority setting should be set for the application objects?
 - A product “build” procedure should exist to set public authority during application development (or ensure public is set via the CRTxxx interface)
 - Set *EXCLUDE public authority for “sensitive” data objects and *USE or *CHANGE for objects used during run-time that do not contain sensitive data.
 - If adopted authority or profile swap is used within the application then all objects can have PUBLIC(*EXCLUDE), which provides great security! This is discussed later in this presentation.
 - **NOTE: *USE or *CHANGE on a DB2 or IFS File will expose data. *USE allows the data to be “read” or FTP download from the server**
 - **Never set *ALL** public authority as this allows any user the ability to delete or change ownership of the object

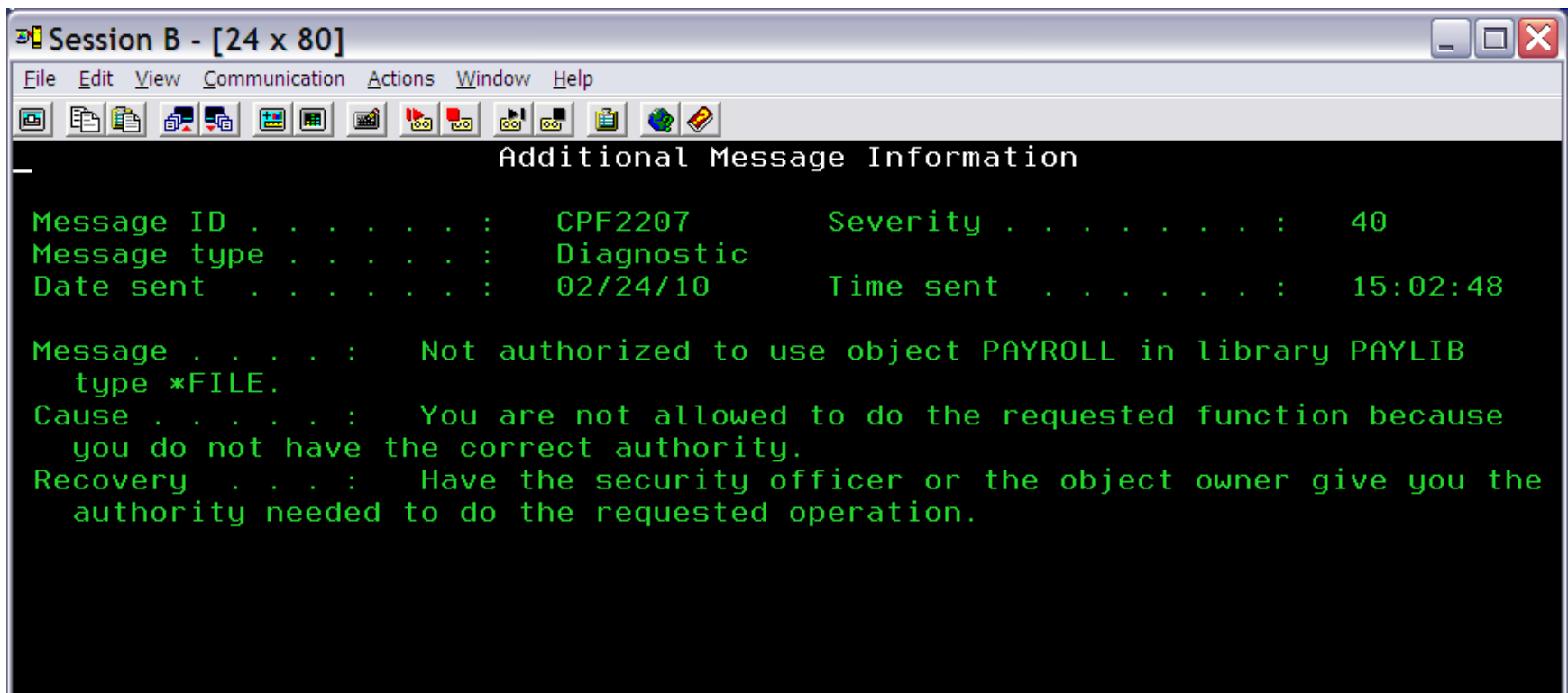
Public Authority Considerations continued...

What public authority value should be used for objects created when the application is run by an end user?

- Set *EXCLUDE public authority for “sensitive” data objects and *USE or *CHANGE for objects used during run-time that do not contain sensitive data
 - If adopted authority or profile swap is used within the application then all objects can have PUBLIC(*EXCLUDE), and provides great security!. This is discussed later in this presentation.
 - **NOTE: *USE or *CHANGE on a DB2 or IFS File will expose data. *USE allows the data to be “read” or FTP download from the server**
- **Never set *ALL** public authority as this allows any user the ability to delete or change ownership of the object
- Set public authority for the object via the AUT parm of the CRTXXX interface at application run-time... or....
- Use the GRTOBJAUT or CHGAUT interface, within your application, to change public authority at run-time

Authority problems at application run-time

- Authority problems can occur at application run-time when the user running the application is a low power user and tight security is applied to the objects
 - Techniques, which can be utilized by the application, exist to avoid these authority problems and will be discussed in this presentation
 1. Authorization lists used to secure the application objects
 2. Adopted user profile authority
 3. Profile Swapping



The screenshot shows a terminal window titled "Session B - [24 x 80]" with a menu bar (File, Edit, View, Communication, Actions, Window, Help) and a toolbar. The terminal output displays "Additional Message Information" followed by a diagnostic message (CPF2207) with a severity of 40. The message text is: "Message : Not authorized to use object PAYROLL in library PAYLIB type *FILE. Cause : You are not allowed to do the requested function because you do not have the correct authority. Recovery : Have the security officer or the object owner give you the authority needed to do the requested operation."

```
Additional Message Information

Message ID . . . . . : CPF2207      Severity . . . . . : 40
Message type . . . . . : Diagnostic
Date sent . . . . . : 02/24/10     Time sent . . . . . : 15:02:48

Message . . . . . : Not authorized to use object PAYROLL in library PAYLIB
type *FILE.
Cause . . . . . : You are not allowed to do the requested function because
you do not have the correct authority.
Recovery . . . . . : Have the security officer or the object owner give you the
authority needed to do the requested operation.
```

Private Authority

- GRTOBJAUT OBJ(PAYLIB/PAYROLL) OBJTYPE(*FILE) **USER(FRED) AUT(*USE)**
- DSPOBJAUT OBJ(PAYLIB/PAYROLL) OBJTYPE(*FILE)

```
Session A - [24 x 80]
File Edit View Communication Actions Window Help
Display Object Authority
Object . . . . . : PAYROLL      Owner . . . . . : UEHLING
Library . . . . . : PAYLIB      Primary group . . . . . : *NONE
Object type . . . . . : *FILE      ASP device . . . . . : *SYSBAS
Object secured by authorization list . . . . . : *NONE

User      Group      Object
Authority
*PUBLIC
UEHLING
FRED      *EXCLUDE
          *ALL
          *USE ← Private authority

Bottom
Press Enter to continue.
F3=Exit  F11=Display detail object authorities  F12=Cancel  F17=Top
F18=Bottom
(C) COPYRIGHT IBM CORP. 1980, 2009.
Mâ a 01/001
I902 - Session successfully started
```

Private authority

Private Authority Considerations

- Private authority to application objects should be avoided or minimally used
 - Private authorities increase the maintenance overhead for the system administrator
 - Adding new application users is problematic
 - Private authorities can cause system performance problems at both run-time and system save
 - Private authorities complicate a security audit

Private Authority Considerations continued...

- Private authorities, if necessary, should be implemented via an authorization list (next topic)
- Group user profiles provide an alternative to granting private authorities to individual users (but groups are usually not controlled by the application development environment)
- Private authorities will provide object access, outside of an application run-time environment, to perform operations such as a “Query” or “FTP” so at times are necessary

Authorization lists (Create the *AUTL and add users)

- CRTAUTL AUTL(PAYLIST) AUT(*EXCLUDE)
- EDTAUTL AUTL(PAYLIST)
- Add the users and authority via F6

```

Session A - [24 x 80]
File Edit View Communication Actions Window Help
Edit Authorization List

Object . . . . . : PAYLIST          Owner . . . . . : PAYROLL
Library . . . . . : QSYS             Primary group . . . . . : *NONE

Type changes to current authorities, press Enter.

User      Object Authority List
*PUBLIC  *EXCLUDE  _
PAYROLL  *ALL      X
JMU      *CHANGE  _
FRED     *USE     _

F3=Exit  F5=Refresh  F6=Add new users
F11=Display detail object authorities  F12=Cancel  F24=More keys
Object authorities changed.

Bottom

MA a 10/014
I902 - Session successfully started
    
```

Authorized Users

Authorization lists continued (Secure the objects)...

- GRTOBJAUT OBJ(PAYLIB/PAYROLL) OBJTYPE(*FILE) **AUTL(PAYLIST)**
 - GRTOBJAUT OBJ(PAYLIB/PAYFILE2) OBJTYPE(*FILE) **AUTL(PAYLIST)**
- or
- CRTPF FILE(PAYLIB/PAYROLL) RCDLEN(80) **AUT(PAYLIST)**

```

Session A - [24 x 80]
File Edit View Communication Actions Window Help
Display Object Authority

Object . . . . . : PAYROLL      Owner . . . . . : UEHLING
Library . . . . . : PAYLIB      Primary group . . . . . : *NONE
Object type . . . . . : *FILE      ASP device . . . . . : *SYSBAS

Object secured by authorization list . . . . . : PAYLIST

User      Group      Object
*PUBLIC
UEHLING
FRED      Authority
*EXCLUDE
*ALL
*USE

Press Enter to continue.

F3=Exit  F11=Display detail ob
F14=Display authorization list
(C) COPYRIGHT IBM CORP. 1980, 20
MA a
1902 - Session successfully started
    
```

Authorization list

Display AUTL Objects

```

Session A - [24 x 80]
File Edit View Communication Actions Window Help
Display Authorization List Objects

Authorization list . . . . . : PAYLIST
Library . . . . . : QSYS
Owner . . . . . : PAYROLL
Primary group . . . . . : *NONE

Object      Library      Type      Owner      Primary      Text
PAYFILE2    PAYLIB      *FILE     PAYROLL    *NONE        Payroll File 2
PAYROLL     PAYLIB      *FILE     PAYROLL    *NONE        Payroll File
    
```


Authorization list considerations

- Authorization lists provide a great mechanism to authorize objects with the same set of users and authority
- Documentation is a requirement to let the system administrator know what authorization list(s) are used to secure the data (so they can add/remove users when necessary)
- The “application” authorization list should be created during application install via the pre-install exit program of RSTLICPGM (or created by the system administrator prior to restore of the application on the system
 - CRTAUTL AUTL(PAYLIST) AUT(*EXCLUDE)
 - This provides support to “connect” the authorization list to each secured object during the object restore process

Authorization list considerations continued...

- Authorization lists help the system admin manage authority on the system
 - Adding new application users is accomplished by adding the new user to the authorization list
- Authorization lists can cause system performance problems at both run-time and system save (private authorities are still involved)
 - Use the authorization list only when necessary on a minimal number of objects that need to be authorized for “outside the application” access
 - Authorization lists can be used to secure “sensitive” objects
 - Objects that are used at application run-time, that do not contain sensitive data, should be secured with public authority or accessed via other security techniques (described in this presentation)

Determining Object Access

How Does the System Determine Access an Object?

- When the user logs into the server and tries to access an object, the OS will make checks to determine if the user is authorized to perform the action against the object
 - Object access is defined by the object type
 - Call a *PGM
 - Run a *CMD
 - Open a *FILE
 - Etc...
 - Authority requirements differ between object types and usage
 - *USE to call a *PGM or run a *CMD
 - *USE to open a file for read and *CHANGE to open for update
- **NOTE: *ALLOBJ special authority gives the user access to all objects on the system. *ALLOBJ = Security officer!**

How Does the System Determine Access an Object?

- To access or use an object, the user must have sufficient authority to the object. The authority could come from:
 - *ALLOBJ special authority (Security officer Authority)
 - (privilege given to a user via the CRT/CHGUSRPRF command)
 - Private authority to the object
 - Authorization list authority
 - *PUBLIC authority
 - Adopted authority (discussed later)

i5/OS Authority Search Order

*ALLOBJ Private Authorization List	USER	Stops when ANY authority is found
*ALLOBJ Primary Group Private Authorization List	GROUP(S)	Repeats for each group until sufficient authority is accumulated or no more groups
Object Authorization List	*PUBLIC	Checked when no authority is found for User or Group(s)
Adopted		Only checked when authority is not sufficient

IBM i System Integrity Considerations

Authority checking and integrity - Security level 40 & 50

User written applications, running at security level 40 or 50, **MUST** use system interfaces (commands and APIs) to gain access to the objects.

- Authority checking is enforced by the system interface
 - Parameter Validation is performed
 - Object Domain checking is performed
 - Object Hardware storage protection is performed
-
- Direct access by user programs to system objects is not allowed at Security level 40 and 50 due to domain and hardware storage protection attributes of the objects. “Direct access” means addressing the object internals directly via “pointer” access (accessing the object “internals” directly via address).
 - Test your application on security level 50, QSECURITY system value = 50

Object Domain attributes - Object integrity

Every object: *CMD, *FILE, *PGM, etc. has a “domain”
Every program has a “state” (*SYSTEM or *USER)

Program state is compared against object Domain

Program run state: *SYSTEM or *USER (DSPPGM/DSPSRVPGM)
Object Domain: *SYSTEM or *USER (DSPOBJD)

Programs running *SYSTEM state can access both *USER and *SYSTEM domain.
Programs running *USER state can only access *USER domain objects.

- **Security level 30 ALLOWS access regardless of state/domain combination**
- **Security level 40 and 50 enforce domain checking**

Object Domain, Program State

```
Session A - [24 x 80]
File Edit View Communication Actions Window Help
[Icons]
Display Object Description - Full
Library 1 of 1
Object . . . . . : SIGNOFF      Attribute . . . . . :
Library . . . . . : QSYS          Owner . . . . . : QSYS
Library ASP device . . . . . : *SYSBAS      Library ASP group . . . . . : *SYSBAS
Type . . . . . : *CMD          Primary group . . . . . : *NONE

User-defined information:
Attribute . . . . . :
Text . . . . . : Sign Off

Creation information:
Creation date/time . . . . . : 02/03/09 15:35:48
Created by user . . . . . : *IBM
System created on . . . . . : 00000000
Object domain . . . . . : *SYSTEM

Press Enter to continue.
F3=Exit F12=Cancel
(C) COPYRIGHT IBM CORP.
MA a
I902 - Session successfully started
```

Object Domain

Program State

```
Session A - [24 x 80]
File Edit View Communication Actions Window Help
[Icons]
Display Program Information
Program . . . . . : QCMD      Library . . . . . : QSYS
Owner . . . . . : QSYS
Program attribute . . . . . :

Program statistics:
Number of parameters . . . . . : 0
Program size (bytes) . . . . . : 200704
Associated space size (bytes) . . . . . : 0
Static storage size (bytes) . . . . . : 0
Automatic storage size (bytes) . . . . . : 6688
Number of MI instructions . . . . . : 2732
Number of ODT entries . . . . . : 1161
Program state . . . . . : *SYSTEM
Program domain . . . . . : *USER
```

User domain objects

Creating User Domain Objects

You are not guaranteed that you will be able to create user domain user objects (***USRSPC**, ***USRIDX**, and ***USRQ**) in any library other than QTEMP

- QALWUSRDMN system value controls into which libraries/directories the objects can be created.
- QTEMP is the only library that can always contain user domain versions of these objects.

NOTE: User domain objects are a security risk because they can be directly read on any security level via pointer access.

Trojan Horse concerns

Library qualify object reference

- Program and command invocation should be library qualified rather than *LIBL
 - CALL **PAYLIB/PAYROLL** (parms)
 - **QSYS/CRTDTAARA** DTAARA(LIB/DTAARA1) TYPE(*CHAR)
- By qualifying the object reference, you prevent trojan horse attacks via the library list. For example, someone creates a program called PAYROLL and places it in a library in the library list ahead of PAYLIB.
- For objects that contain translatable text (message files, panel groups, etc) you should use *LIBL for library qualification in order to pick up different versions of the translated text.
- **NOTE: Using library qualified references can cause potential testing difficulties. A test system or partition is required to test application changes rather than just manipulating a library list to test. The security benefits outweigh the test considerations!**

Parameter Validation Considerations

Test input parameters for valid values

- Testing parameter input is a good programming practice
 - Testing ranges of valid values for numeric parms
 - Testing for valid “strings” (*GOOD vrs *GOOOD)
 - Testing for valid “syntax” of date, time, names, etc.
 - ETC...
- The lack of parameter validation could cause an application to fail and introduce a security problem
 - Abnormal application end could leave a job with libraries in the library list, open files, etc.
 - Abnormal application end could leave a job at a command line with a profile swap or adopted authority condition
 - Make sure to add appropriate Exception Handling to your code to prevent abnormal end
 - ETC...

Menu Security

- Menu Security = Display a menu with allowable options and forget about object security!
 - Still a common “UNSECURE” practice today
- Don't rely on menu security!
 - Menu Security was acceptable... 20 years ago
 - Typical Design was PUBLIC(*ALL) for object security
- Objects are exposed...
 - FTP from a client system, read/change any data file that has PUBLIC(*CHANGE or *ALL), or download any PUBLIC(*USE) file
- Secure your sensitive objects with the appropriate level of authority at the object level!

Security Implementation Options

1. **Adopted User profile Authority**
2. **Profile Swap to obtain additional Authority**

Adopted User Profile Authority

Adopted User Profile Authority

Adopted Authority implementation

- The implementation is to “Secure” all application objects so the application is in control of all read and update operations
 - Set PUBLIC authority to *EXCLUDE for **ALL** application objects
 - The system administrator can grant authority to the “entry point” of the application to provide access to authorized users
- Within the application, use program adopted authority to gain access to objects in the QSYS file system while the application is running
- To access objects in the IFS file system, Profile Swap can be used (see Profile Swap topic in this presentation)

Adopted User Profile Authority

Adopted Authority Considerations

- If all objects are PUBLIC(*EXCLUDE), an end user has NO access to any of the objects without running the application
 - This means, to run a QUERY, FTP or other interface, outside of the application, authority will be required. Private authority would need to be granted to certain objects for the list of users who need access outside the application
- Certain “network” interfaces, specifically in the DB2 area, will not work without “extra” security considerations. Interfaces like DRDA, Web Query, ODBC, JDBC, etc. cannot easily use adopted authority so private authority would be required (unless you design a procedure to call via the network interface)

Adopted User Profile Authority

Authorization scheme using adopted authority

- Create all application objects with PUBLIC(*EXCLUDE) authority
- Set the Ownership of all application objects to the application profile

```
Session A - [24 x 80]
File Edit View Communication Actions Window Help
Display Object Authority
Object . . . . . : PAYROLL      Owner . . . . . : PAYROLL
Library . . . . . : PAYLIB      Primary group . . . . . : *NONE
Object type . . . . . : *FILE      ASP device . . . . . : *SYSBAS
Object secured by authorization list . . . . . : *NONE

User      Group      Object
*PUBLIC   *PUBLIC   Authority
PAYROLL   *PUBLIC   *EXCLUDE
          *PAYROLL *ALL

Bottom

Press Enter to continue.

F3=Exit  F11=Display detail object authorities  F12=Cancel  F17=Top
F18=Bottom
(C) COPYRIGHT IBM CORP. 1980, 2009.
MA a 01/001
I902 - Session successfully started
```

Owner is the Application Profile

Public Authority set to *EXCLUDE

Adopted User Profile Authority

Adopted Authority – Program attribute

- When a program or service program adopts authority, it uses both the authority of the “process (job) user” **and** the authority of the program/srvpgm **owner**.
- Adopted authority specified when program/srvpgm is created
 - C, CL, RPG, etc. --> by specifying USRPRF(*OWNER) on the CRTxxxPGM or CRTBNDxx commands
 - Or via the CHGPGM CL command, USRPRF(*OWNER) parm

Adopted Authority

- Used to temporarily give additional authority
- When a program with USRPRF(*OWNER) runs, the authority in effect is the job user plus the owner of the program
- Both special authorities and private authorities are adopted (the program owner's groups are not included)
- Additional authority is in effect for as long as the program is in the call stack

Adopted User Profile Authority considerations

Adopted Authority considerations

- Application designer must be careful to not include adopted authority in any authority checks the application specifically makes (e.g. CHKOBJ or authority checking APIs)
- IFS interfaces do not honor adopted authority
 - Must “swap” to powerful profile to ensure a user is authorized to access an IFS object (swap discussed later...)
- Make sure command line (or exit program) **not available** to end user when adopting authority or swapped to a powerful profile. Use **LMTCPB(*YES)** user profile parm to prevent command line access with authority of the adopted/swapped profile.
 - Adopted authority and swapped user profile authority given to the command line user
 - Technique to allow a command line will be discussed in the “use adopted authority” charts

Creating a program that adopts authority

- CRTCLPGM PGM(PAYROLL/ADOPTPGM) TEXT('Program that adopts owner authority') **USRPRF(*OWNER)**
- DSPPGM PGM(PAYROLL/ADOPTPGM)

```
Session A - [24 x 80]
File Edit View Communication Actions Window Help
Display Program Information
Program . . . . . : ADOPTPGM Library . . . . . : PAYROLL
Owner . . . . . : PAYROLL
Program attribute . . . : CLP

Program creation information:
Program creation date/time . . . . . : 02/25/10 15:44:44
Type of program . . . . . : OPM
Source file . . . . . : QCLSRC
Library . . . . . : UEHLING
Source member . . . . . : ADOPTPGM
Source file change date/time . . . . . : 02/25/10 15:38:35
Observable information . . . . . : *ALL
User profile . . . . . : *OWNER
Use adopted authority . . . . . : *YES
Log commands (CL program) . . . . . : *JOB
Allow RTVCLSRC (CL program) . . . . . : *YES
Fix decimal data . . . . . : *NO

Press Enter to continue.

F3=Exit F12=Cancel
(C) COPYRIGHT IBM CORP. 1980, 2009.

Mâ a 01/001
I902 - Session successfully started
```

Program adopts the owners authority

Program adopts authority

Another attribute to be discussed... next

“Use Adopted Authority” – program attribute

“Use Adopted Authority” program attribute

- A second program attribute, “Use Adopted Authority”, can be set on an application program
 - Set via the CHGPGM CL command, USEADPAUT parm
 - Attribute is **not** available on the CRTxxxPGM commands
- This attribute causes all previously adopted authority, available to the job from previously called programs, to be dropped/ignored when this program is called
- This attribute allows an application to present a command line, perform authority checks, etc., from the program with the Use Adopted Authority = *NO attribute, without previously adopted authority being considered

Use Adopted Authority program attribute

- CRTCLPGM PGM(PAYROLL/NOADOPTAUT) TEXT('USEADPAUT example')
- CHGPGM PGM(PAYROLL/NOADOPTAUT) **USEADPAUT(*NO)**
- DSPPGM PGM(PAYROLL/NOADOPTAUT)

```
Session A - [24 x 80]
File Edit View Communication Actions Window Help
Display Program Information
Program . . . . . : NOADOPTAUT      Library . . . . . : PAYROLL
Owner . . . . . : PAYROLL
Program attribute . . . : CLP

Program creation information:
Program creation date/time . . . . . : 02/26/10  10:32:26
Type of program . . . . . : OPM
Source file . . . . . : QCLSRC
Library . . . . . : UEHLING
Source member . . . . . : NOADOPTAUT
Source file change date/time . . . . . : 02/26/10  10:31:41
Observable information . . . . . : *ALL
User profile . . . . . : *USER
Use adopted authority . . . . . : *NO
Log commands (CL program) . . . . . : *JOB
Allow RTVCLSRC (CL program) . . . . . : *YES
Fix decimal data . . . . . : *NO

More . . .

Press Enter to continue.

F3=Exit  F12=Cancel
(C) COPYRIGHT IBM CORP. 1980, 2009.

Mâ a 01/001
I902 - Session successfully started
```

Program does not adopt authority

Program does NOT use previously adopted user profile authority

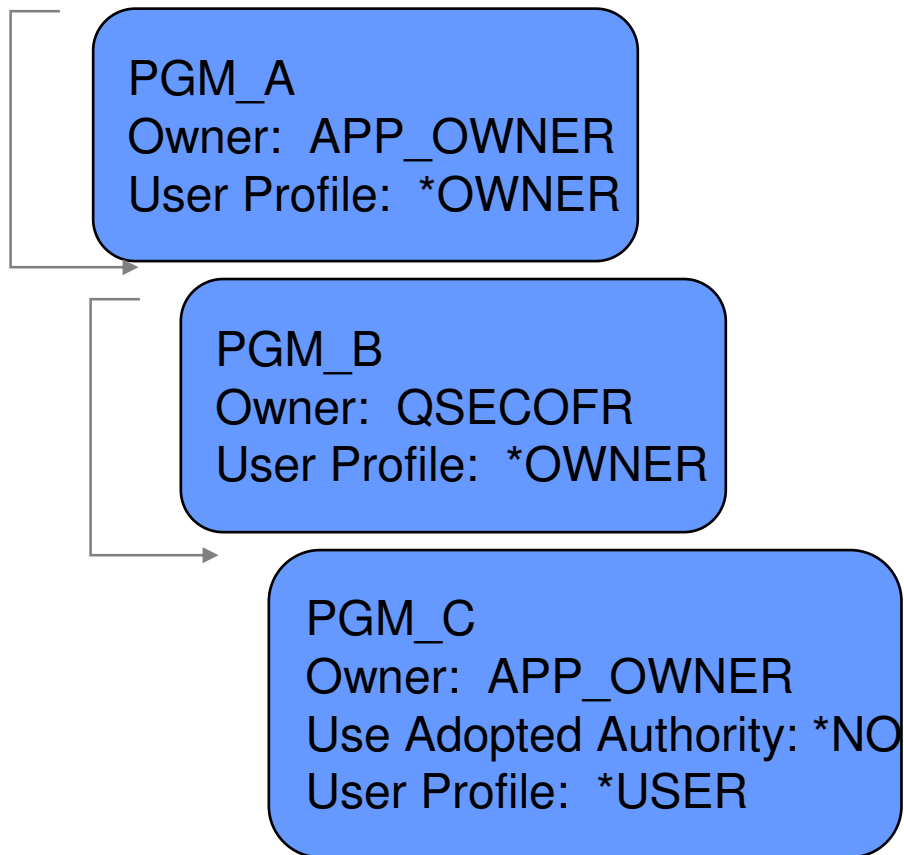
“Use Adopted Authority” – continued...

“Use Adopted Authority” program attribute

- The “Use Adopted Authority = *NO” attribute blocks all adopted authority from previously called programs in the job
- The program with the “Use Adopted Authority = *NO” attribute can itself adopt authority
 - All previously adopted authority, from previously called programs, is “not used”
 - The Owner of this program (the owner of the program with the Use Adopted Authority = *NO) can be “adopted”. This provides the ability to drop all adopted authority and include only the adopted authority from the well known “program owner”.

Adopted Authority Example

Program Call Stack



Scenario:
Need to modify a file
Requires *CHANGE authority
*PUBLIC authority of file is *EXCLUDE

Signed on User - JEFF

Users Checked

JEFF then APP_OWNER

JEFF then APP_OWNER (from
PGMA) then QSECOFR

~~PGMC = USEADPAUT(*NO)~~

Only JEFF because USEADPAUT(*NO)
and USRPRF(*USER)

Adopted Authority Summary

- Secure the application objects with PUBLIC(*EXCLUDE)
 - Provides a secure environment for all application objects
- Use program adopted authority to gain access to the application objects at application run-time
- Use the “Use Adopted Authority = *NO” attribute if a command line, an authority check, etc. is necessary
- Use an Authorization List to provide “private authority” access to a set of data files that may need to be examined “outside” of the application, via Query, FTP, etc.
- Profile “Swap” is required for accessing secure IFS objects (next topic)

Profile “Swapping”

Profile Swapping for Object Access

Profile Swapping implementation

- The implementation is to “Secure” all application objects so the application is in control of all read and update operations
 - Set PUBLIC authority to *EXCLUDE for all application objects
 - The system administrator can grant authority to the “entry point” of the application to provide access to authorized users
- Within the application, use profile swapping to gain access to objects in the IFS file system while the application is running
 - Profile swap can be used to gain access to objects in the QSYS file system as well as IFS

Profile Swapping for Object Access

Profile Swapping Considerations

- If all objects are PUBLIC(*EXCLUDE), an end user has NO access to any of the objects without running the application
 - This means, to run a QUERY, FTP or other interface, outside of the application, authority will be required. Private authority would need to be granted to certain objects for the list of users who need access outside the application
- Certain “network” interfaces, specifically in the DB2 area, will not work without “extra” security considerations. Interfaces like DRDA, Web Query, ODBC, JDBC, etc. cannot easily use adopted authority so private authority would be required (unless you design a procedure to call via the network interface)

Profile Swapping

- Used to change the job user profile running the application
 - NOTE: Profile Swap changes the jobs User Profile. All audit records sent by the system will reflect the “swapped user profile”... but the job name in the audit record doesn’t change.
- When the profile swapping interface is used to gain object access, the process user profile of the job is changed from the currently running user to the swapped user
 - Job running as user “**JEFF**” is swapped, by the swap API called by the application, to run as user “**PAYROLL**”
- All authority checks are based on the “Swapped Profile” (A profile swap will change both the process profile and group profile list)
The swapped profile is in effect until another “Swap Profile” is run or the job ends

Profile Swapping considerations

Profile Swapping considerations

- Application designer must be careful to not include swapped profile in any authority checks the application specifically makes (e.g. CHKOBJ or authority checking APIs)
- Make sure command line (or exit program) **not available** to end user when adopting authority or swapped to a powerful profile. Use the **LMTCPB(*YES)** user profile parm on the “swapped” user profile to limit command line access.
 - Adopted authority and swapped user profile authority given to the command line user
 - For profile swapping, you need to “swap back” to the previous (original) user if presenting a command line interface

Profile Swapping

Profile swap

Use of the QSYGETPH (profile handle create), QWTSETP (profile swap) and QSYRLSPH (release profile handle) APIs allow the user profile of a job to be swapped.

- Log in as user "JEFF"
- Swap to user "PAYROLL"

Swap APIs

1. QSYGETPH – Get Profile Handle
2. QWTSETP – Swap profile in the job using profile handle
3. QSYRLSPH – Release profile handle

Disclaimer

This presentation contains programming examples ("Sample Code").

IBM grants you a nonexclusive copyright license to use the Sample Code to generate similar function tailored to your own specific needs.

The Sample Code is provided by IBM for illustrative purposes only. The Sample Code has not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of the Sample Code.

The Sample Code contained herein is provided to you "AS IS" without any warranties of any kind. THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT ARE EXPRESSLY DISCLAIMED. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSIONS MAY NOT APPLY TO YOU. IN NO EVENT WILL IBM BE LIABLE TO ANY PARTY FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES FOR ANY USE OF THE SAMPLE CODE INCLUDING, WITHOUT LIMITATION, ANY LOST PROFITS, BUSINESS INTERRUPTION, LOSS OF PROGRAMS OR OTHER DATA ON YOUR INFORMATION HANDLING SYSTEM OR OTHERWISE, EVEN IF WE ARE EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Profile Swap example code

Profile Swap instructions

1. The source code in the following slide can be used to Test User Profile Swap
2. Copy the source code, next slide, into a source physical file, perhaps member PAYADP in file QCLSRC, and update the source with a test user name
3. The CL program you create needs to adopt authority of a powerful user, such as QSECOFR. The profile swap APIs (QSYGETPH) require a significant amount of authority to run. The adopt will provide authority required to swap when this program is run by a low power user
4. Create the program with PUBLIC(*EXCLUDE) authority. This program is only an example program. When using profile swap in an application, the program should **never return control** to the end user (command line) without swapping back to the original user first..
 1. Signon as user QSECOFR to create the program... or
 2. Signon as an *ALLOBJ user, create the program, and then change the owner to QSECOFR. The program doesn't have to be owned by QSECOFR, any *ALLOBJ profile will work, but QSECOFR exists on every system so it is a good choice to use
5. CRTCLPGM PGM(PAYROLL/PAYADP) USRPRF(*OWNER) AUT(*EXCLUDE)

Example CL program to Swap User Process user profile

```
/* Signon with an *ALLOBJ user, QSECOFR, to create this program. Create this program with USRPRF(*OWNER) to adopt */
/* authority required to get a profile handle for a USRPRF. NOTE: Control access to this program, PUBLIC(*EXCLUDE). */
/* NOTE: For testing purposes, use command, DSPJOB OPTION(*STSA), before and after each QWTSETP invocation to see */
/* the swap results (current user profile changes with the swap). */
```

PGM

```
DCL &HNDLCUR *CHAR 12 VALUE(' ') /* 12 character profile handle variable for the original user */
DCL &HNDL *CHAR 12 VALUE(' ') /* 12 character profile handle variable for the new user */
```

```
/* Call QSYGETPH to get a profile handle for the current job user profile. */
```

```
CALL QSYS/QSYGETPH (*CURRENT' *NOPWDCHK' &HNDLCUR)
```

```
/* Call QSYGETPH to get a profile handle for the profile to swap to. NOTE: Change XXX to the user who you want to swap to. */
```

```
CALL QSYS/QSYGETPH ('XXX' *NOPWDCHK' &HNDL)
```

```
/* DSPJOB OPTION(*STSA) - for testing, display the job, current user profile */
```

```
/* Call QWTSETP to swap to the new profile. Job will run under user profile XXX after the swap */
```

```
CALL QSYS/QWTSETP &HNDL
```

```
/* DSPJOB OPTION(*STSA) - for testing, display the job, current user profile */
```

```
/* add application logic here */
```

```
/* CLEANUP processing */
```

```
/* Call QWTSETP to swap back to the original user profile. */
```

```
CALL QSYS/QWTSETP &HNDLCUR
```

```
/* DSPJOB OPTION(*STSA) - for testing, display the job, current user profile */
```

```
/* Call QSYRLSPH to release the two profile handles */
```

```
CALL QSYS/QSYRLSPH &HNDLCUR
```

```
CALL QSYS/QSYRLSPH &HNDL
```

```
ENDPGM
```

Profile Swap Cleanup and release of profile handle

Profile swap (Cleanup)

Use of the QSYGETPH and QWTSETP APIs allow the user profile of a job to be swapped.

- Log in as user "JEFF"
- Swap to user "PAYROLL"
- The job is now running under user "PAYROLL". If the application fails, the job continues to run under "PAYROLL"
- A Scope Message provides the ability to cleanup or swap back to original user.
 - NOTE: Scope handling programs can be used to cleanup anything within the application, such as profile swapping, libraries in the library list, open files, etc.
- Code example follows...

Disclaimer

This presentation contains programming examples ("Sample Code").

IBM grants you a nonexclusive copyright license to use the Sample Code to generate similar function tailored to your own specific needs.

The Sample Code is provided by IBM for illustrative purposes only. The Sample Code has not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of the Sample Code.

The Sample Code contained herein is provided to you "AS IS" without any warranties of any kind. THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT ARE EXPRESSLY DISCLAIMED. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSIONS MAY NOT APPLY TO YOU. IN NO EVENT WILL IBM BE LIABLE TO ANY PARTY FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES FOR ANY USE OF THE SAMPLE CODE INCLUDING, WITHOUT LIMITATION, ANY LOST PROFITS, BUSINESS INTERRUPTION, LOSS OF PROGRAMS OR OTHER DATA ON YOUR INFORMATION HANDLING SYSTEM OR OTHERWISE, EVEN IF WE ARE EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Scope Message program

```
/* Signon with an *ALLOBJ user to create this program. Create this program with USRPRF(*OWNER) in order to adopt */
/* authority required to get a profile handle for a USRPRF. NOTE: Control access to this program, PUBLIC(*EXCLUDE). */
```

PGM

```
DCL &ERRCDE *CHAR 8 VALUE(X'0000000000000000')
```

```
DCL &MSGKEY *CHAR 4 VALUE(X'00000000')
```

```
DCL &HNDLCUR *CHAR 12 VALUE(' ')
```

```
DCL &HNDL *CHAR 12 VALUE(' ')
```

```
/* Call QSYGETPH to get a profile handle for the current user. */
```

```
CALL QSYS/QSYGETPH (*CURRENT' *NOPWDCHK' &HNDLCUR)
```

```
/* The following API will send a scope message that causes program SCOPEPGM in library QGPL to be called when */
/* this program ends either normally or abnormally. */
```

```
CALL QSYS/QMHSNDSM +
(*CSE ' /* Scope type */ +
'SCOPEPGM QGPL ' /* Scope program name */ +
&HNDLCUR /* Scope data */ +
X'0000000C' /* Scope data length - 12 */ +
&MSGKEY /* Message key */ +
&ERRCDE) /* Error code */
```

```
/* Call QSYGETPH to get a profile handle for a user. NOTE: Change XXX to the user who you want to swap to */
```

```
CALL QSYS/QSYGETPH ('XXX' *NOPWDCHK' &HNDL)
```

```
/* Call QWTSETP to swap to the profile. */
```

```
CALL QSYS/QWTSETP &HNDL
```

```
/* Normal application logic */
```

```
ENDPGM
```


Scope Handling program

```
PGM (&DATA)    /* SCOPEPGM */
```

```

/*****/
/* This program is called when the invocation that ran the    */
/* QMHSNDSM API returns either normally or abnormally.      */
/*****/

```

```

DCL &DATA *CHAR 12 /* Data received as input when this scope    */
                  /* handling program is called. This data    */
                  /* is variable length and is declared and    */
                  /* set by the program that issues the      */
                  /* QMHSNDSM API.                            */
                  /* For this test program, pass the 12      */
                  /* byte profile handle of the original      */
                  /* user obtained via *CURRENT on QSYGETPH.   */

```

```
/* Program logic to cleanup. */
```

```

/* Call QWTSETP to swap back to the original profile. */
CALL QSYS/QWTSETP &DATA

```

```

/* Call QSYSRSLPH to release the profile handle.      */
CALL QSYS/QSYRSLPH &DATA

```

```
ENDPGM
```

Profile Swapping Summary

- Secure the application objects with PUBLIC(*EXCLUDE)
 - Provides a secure environment for all application objects
- Use profile swapping to gain access to the application objects at application run-time
- Make sure you swap back to the original user if presenting a command line or performing an authority check, etc.
- Use an Authorization List to provide “private authority” access to a set of data files that may need to be examined “outside” of the application, via Query, FTP, etc.
- Profile “Swap” is required for accessing secure IFS objects

Summary

- Secure objects with PUBLIC(*EXCLUDE) authority
- Use Adopted Authority and/or Profile Swapping to gain object access
- Private Authorities and Authorization Lists can be used to provide object authority outside of the application
- Always take care when presenting the user a command line when using either adopted user profile authority or profile swapping

IBM Systems Lab Services and Training

Helping you gain the IBM Systems skills
needed for smarter computing

- Comprehensive education, training and service offerings
- Expert instructors and consultants, world-class content and skills
- Multiple delivery options for training and services
- Conferences explore emerging trends and product strategies

www.ibm.com/training

Special Programs:

- IBM Systems 'Guaranteed to Run' Classes -- ***Make your education plans for classes with confidence!***
- Instructor-led online (ILO) training ***The classroom comes to you.***
- Customized, private training
- Lab-based services assisting in high tech solutions

Special notices

This document was developed for IBM offerings in the United States as of the date of publication. IBM may not make these offerings available in other countries, and the information is subject to change without notice. Consult your local IBM business contact for information on the IBM offerings available in your area.

Information in this document concerning non-IBM products was obtained from the suppliers of these products or other public sources. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. Send license inquires, in writing, to IBM Director of Licensing, IBM Corporation, New Castle Drive, Armonk, NY 10504-1785 USA.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

The information contained in this document has not been submitted to any formal IBM test and is provided "AS IS" with no warranties or guarantees either expressed or implied.

All examples cited or described in this document are presented as illustrations of the manner in which some IBM products can be used and the results that may be achieved. Actual environmental costs and performance characteristics will vary depending on individual client configurations and conditions.

IBM Global Financing offerings are provided through IBM Credit Corporation in the United States and other IBM subsidiaries and divisions worldwide to qualified commercial and government clients. Rates are based on a client's credit rating, financing terms, offering type, equipment type and options, and may vary by country. Other restrictions may apply. Rates and offerings are subject to change, extension or withdrawal without notice.

IBM is not responsible for printing errors in this document that result in pricing or information inaccuracies.

All prices shown are IBM's United States suggested list prices and are subject to change without notice; reseller prices may vary.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

Any performance data contained in this document was determined in a controlled environment. Actual results may vary significantly and are dependent on many factors including system hardware configuration and software design and configuration. Some measurements quoted in this document may have been made on development-level systems. There is no guarantee these measurements will be the same on generally-available systems. Some measurements quoted in this document may have been estimated through extrapolation. Users of this document should verify the applicable data for their specific environment.

Revised September 26, 2006

Special notices (cont.)

IBM, the IBM logo, ibm.com AIX, AIX (logo), AIX 5L, AIX 6 (logo), AS/400, BladeCenter, Blue Gene, ClusterProven, DB2, ESCON, i5/OS, i5/OS (logo), IBM Business Partner (logo), IntelliStation, LoadLeveler, Lotus, Lotus Notes, Notes, Operating System/400, OS/400, PartnerLink, PartnerWorld, PowerPC, pSeries, Rational, RISC System/6000, RS/6000, THINK, Tivoli, Tivoli (logo), Tivoli Management Environment, WebSphere, xSeries, z/OS, zSeries, Active Memory, Balanced Warehouse, CacheFlow, Cool Blue, IBM Systems Director VMControl, pureScale, TurboCore, Chiphopper, Cloudscape, DB2 Universal Database, DS4000, DS6000, DS8000, EnergyScale, Enterprise Workload Manager, General Parallel File System, , GPFS, HACMP, HACMP/6000, HASM, IBM Systems Director Active Energy Manager, iSeries, Micro-Partitioning, POWER, PowerExecutive, PowerVM, PowerVM (logo), PowerHA, Power Architecture, Power Everywhere, Power Family, POWER Hypervisor, Power Systems, Power Systems (logo), Power Systems Software, Power Systems Software (logo), POWER2, POWER3, POWER4, POWER4+, POWER5, POWER5+, POWER6, POWER6+, POWER7, System i, System p, System p5, System Storage, System z, TME 10, Workload Partitions Manager and X-Architecture are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries.

A full list of U.S. trademarks owned by IBM may be found at: <http://www.ibm.com/legal/copytrade.shtml>.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

AltiVec is a trademark of Freescale Semiconductor, Inc.

AMD Opteron is a trademark of Advanced Micro Devices, Inc.

InfiniBand, InfiniBand Trade Association and the InfiniBand design marks are trademarks and/or service marks of the InfiniBand Trade Association.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries or both.

Microsoft, Windows and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries or both.

NetBench is a registered trademark of Ziff Davis Media in the United States, other countries or both.

SPECint, SPECfp, SPECjbb, SPECweb, SPECjAppServer, SPEC OMP, SPECviewperf, SPECcapc, SPECchpc, SPECjvm, SPECmail, SPECimap and SPECsfs are trademarks of the Standard Performance Evaluation Corp (SPEC).

The Power Architecture and Power.org wordmarks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

TPC-C and TPC-H are trademarks of the Transaction Performance Processing Council (TPPC).

UNIX is a registered trademark of The Open Group in the United States, other countries or both.

Other company, product and service names may be trademarks or service marks of others.

Revised December 2, 2010