RACF Program Control Demystified!

Bruce R. Wells
brwells@us.ibm.com

Trademarks

- See url http://www.ibm.com/legal/copytrade.shtml for a list of trademarks.

Agenda

- What does “Program Control” mean, anyway?
- Basic program protection - Controlling the ability to run a program
- Clean and dirty environments – why, when, and how?
- Limiting access to a data set only when running a certain program
- Limiting access to a socket only when running a certain program
- UNIX Servers and Daemons
- Program control in enhanced mode
- Extra credit

Program control in three bullets

- A program is a resource like any other
- When certain sensitive resources are accessed, there must not be any untrusted programs in the environment. The PROGRAM class is a white-list of trusted programs.
- Program control in enhanced mode ensures that trusted programs are invoked in the **way** they were intended to be invoked

What does “Program Control” mean, anyway?

- It means any protection activated by SETROPTS WHEN(PROGRAM)
- Which activates the use of profiles in the PROGRAM class
- The PROGRAM class is unique in that you don't use
  - SETROPTS CLASSACT(PROGRAM)
  - SETROPTS RACLIST(PROGRAM)
  - SETROPTS GENERIC(PROGRAM)

  Even though it has analogous behaviors

- Program signing and verification is not *completely* activated by SETROPTS WHEN(PROGRAM), and is not discussed in this presentation
  - Though it does depend on SETROPTS WHEN(PROGRAM)

Basic program protection

- Like in any other class, a profile protects an object of a certain type
- For the PROGRAM class, the object is a program (duh!)
- You can think of MVS Contents Supervision as the 'resource manager'
- The profile name identifies the program name. The member list identifies the library (data set) or libraries containing the program
- READ access allows execution of that program
- You can specify audit options and UACC(READD) to log the use of programs even if you don't care who runs them

Basic program protection …

- RACF recommends protecting a couple of its own programs (I'll tell you why when we've seen slide 22)

```
SETROPTS WHEN(PROGRAM)

RDEFINE PROGRAM ICHDSM00 ADDMEM('SYS1.LINKLIB'//NOPADCHK) UACC(NONE)
PERMIT ICHDSM00 CLASS(PROGRAM) ID(authorized IDs) ACCESS(READ)

RDEFINE PROGRAM IRRDPI00 ADDMEM('SYS1.LINKLIB'//NOPADCHK) UACC(NONE)
PERMIT IRRDPI00 CLASS(PROGRAM) ID(authorized IDs) ACCESS(READ)

SETROPTS WHEN(PROGRAM) REFRESH
```

Basic program protection … auditing

- RACF does not create “command-style” SMF Type 80 records for many of the “list” commands (as it does for ALTUSER, for example)
- But you could create an audit trail using the PROGRAM class:

```
RDEFINE PROGRAM LISTUSER ADDMEM('SYS1.LINKLIB'//NOPADCHK) UACC(READ)
                         AUDIT(ALL(READ))

RDEFINE PROGRAM LISTGRP  ADDMEM('SYS1.LINKLIB'//NOPADCHK) UACC(READ)
                         AUDIT(ALL(READ))

RDEFINE PROGRAM RLIST    ADDMEM('SYS1.LINKLIB'//NOPADCHK) UACC(READ)
                         AUDIT(ALL(READ))

RDEFINE PROGRAM LISTDSD  ADDMEM('SYS1.LINKLIB'//NOPADCHK) UACC(READ)
                         AUDIT(ALL(READ))
```

Basic program protection … One twist: a program alias must be protected separately

- Each RACF command has an alias, so you would need to add the following to the example on the previous slide

```
RDEFINE PROGRAM LU ADDMEM('SYS1.LINKLIB'//NOPADCHK) UACC(READ)
                   AUDIT(ALL(READ))

RDEFINE PROGRAM LG ADDMEM('SYS1.LINKLIB'//NOPADCHK) UACC(READ)
                   AUDIT(ALL(READ))

RDEFINE PROGRAM RL ADDMEM('SYS1.LINKLIB'//NOPADCHK) UACC(READ)
                   AUDIT(ALL(READ))

RDEFINE PROGRAM LD ADDMEM('SYS1.LINKLIB'//NOPADCHK) UACC(READ)
                   AUDIT(ALL(READ))
```

Basic program protection … the member string

- The member string contains three qualifiers, separated by “/”

```
'SYS1.LINKLIB'//NOPADCHK
```

1) The name of the library (data set) that contains the program, fully qualified and enclosed in single quotes
2) The volume on which the data set resides. Can be:
   - A specific volume name (e.g. MYVOL1)
   - Blank (as in this example) to indicate *any* volume
   - '******' (including quotes) to indicate the system IPL volume
3) NO|PADCHK designates behavior relating to **P**rogram **A**ccess to **D**ata **S**ets...more later

- Multiple member strings can be specified to protect the same program when executed from different data sets or volumes

Basic program protection … protecting many programs with one profile

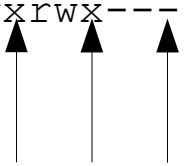
- While not technically a generic profile, an asterisk can be specified at the **end** of the program name as a wild-card
  - For example, ABC* protects programs named ABCD and ABCXYZ
- A more specific profile works as you would expect
  - ABCX* would protect ABCXYZ while ABC* protects ABCD
- This is just how the PROGRAM class works. Do not specify SETROPTS GENERIC(PROGRAM) or SETROPTS GENCMD(PROGRAM)

```
ICH14075I SETROPTS GENERIC had no effect on class PROGRAM.
```

Basic program protection: Contrast with UNIX file execution and auditing

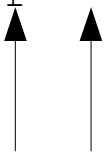
- The ability to run a UNIX file is determined by its permission bits (and optional acl)

```
$ ls -l
-rwxrwx---    1 BRWELLS   DEPTD60    106496 Mar  9  2007 irrhfsu
```



- The ability to audit UNIX file execution is determined by its audit bits

```
$ ls -W
-rwxrwx---  fff---  1 BRWELLS   DEPTD60    106496 Mar  9  2007 irrhfsu
```



- None of this depends on the PROGRAM class and is not considered “program control”

Recap

- A program is just another z/OS resource that can be protected, with the following twists:
  - The class is activated using SETROPTS WHEN(PROGRAM) and refreshed using SETROPTS WHEN(PROGRAM) REFRESH
  - The profile member list is used to identify the library(s) and volume(s) that contain the program
  - Profile names have generic tendencies, but aren't really generic
  - Alias program names must also be defined

So why have I been PROGRAMmed to cringe at the sound of “program control”?

- So far, we have not mentioned the words “clean” and “dirty”
- Let's crank it up to 11 ...

What is a safe environment?

- Insert clever house/safe/guests analogy here ...

Controlled environments ...

- Terminology:
  - An environment in which only “trusted” programs have been loaded is:
    - Clean
    - Safe
    - Controlled
  - As opposed to:
    - Dirty
    - Unsafe
    - Uncontrolled
- A “trusted” program in this sense means one that has been defined to the RACF PROGRAM class
- A clean environment is required when certain sensitive resources exist within an environment, or when certain powerful privileges are available to programs running within the environment

What is this “sensitive stuff” of which you speak?

- Open data sets protected with a WHEN(PROGRAM(x)) conditional access entry
- Open sockets protected with a WHEN(PROGRAM(x)) conditional access entry
- Programs containing sensitive algorithms or data that are protected with EXECUTE access
- Authority to switch identities via BPX.DAEMON and BPX.SERVER

- If none of these things exist within a given address space, then a controlled environment is ***not*** required

PADS: Program Access to Data Sets - Preview

- Fancy term for putting a conditional access list entry on a DATASET profile that specifies a specific program
- Where the program mediates the user's access for a specific function

```
PERMIT XYZCORP.PAYROLL.DATA CLASS(DATASET) ID(HRFOLKS)
       ACCESS(UPDATE) WHEN(PROGRAM(HRAPPL))
```

- You are telling RACF: “I only want the users in the HR group to be able to update the payroll data set when they are running the HRAPPL program.”
- However, while MVS architecture protects address spaces from each other, it does not protect resources within an address space from other tasks/programs within that same address space

IBM z/OS System Integrity Statement

- z/OS “System Integrity” is defined as
  - the inability of any program not authorized by a mechanism under the installation’s control to:
    - circumvent or disable store or fetch protection
    - access a resource protected by the z/OS Security Server (RACF®) [to which the user has not been permitted]
    - obtain control in an authorized state:
      - ✔ in supervisor state
      - ✔ with a protection key less than eight (8)
      - ✔ Authorized Program Facility (APF) authorized.

Controlled environments ...

- RACF requires a controlled environment in order to use one of the sensitive functions
- The PROGRAM class can be thought of as a “white list” of programs
- But it sure would be difficult to define profiles for every program I trust, especially when I'm not particularly concerned with restricting access to them or auditing them
- Can I white-list entire libraries of programs?

- Yes! Introducing the ** profile

Controlled environments ...

- ** is where you define entire trusted **libraries** (in the member list)
- These could contain:
  - LINKLIB programs
  - System functions
  - IBM or other vendor application software
  - Programs that your developers have written, reviewed, and installed under a change control procedure
- Generally speaking, uncontrolled programs are user-written programs not subject to your change control procedures
- The challenge is in knowing what libraries to add ...

Controlled environments ...

- RACF documentation gets you started (reminder: slide 7!)

```
RDEFINE PROGRAM ** ADDMEM('SYS1.LINKLIB'//NOPADCHK) UACC(READ)
RALTER  PROGRAM ** ADDMEM('SYS1.MIGLIB'//NOPADCHK)
RALTER  PROGRAM ** ADDMEM('SYS1.CMDLIB'//NOPADCHK)
RALTER  PROGRAM ** ADDMEM('cee.version.SCEERUN'//NOPADCHK)
RALTER  PROGRAM ** ADDMEM('TCPIP.SEZALOAD'//NOPADCHK
                          'TCPIP.SEZATCP'//NOPADCHK
                          'db2.DSNLOAD'//NOPADCHK
                          'db2.DSNEXIT'//NOPADCHK
                          'ftp.userexits'//NOPADCHK)
```

- Software documentation might identify application-specific libraries which should be added. Or, it might **not.** ☹

Controlled environments ...

- You will make mistakes
- Best to test WHEN(PROGRAM) thoroughly, and continuously, before moving to production!

What could possibly go wrong?

- If somebody renames a library (e.g. changes a data set version qualifier when upgrading) without changing PROGRAM **, the library is no longer under program control
- If somebody moves a library to another volume, and the PROGRAM member string explicitly specifies the old volume, the library/program is no longer under program control
- Some previously unused function/feature of a software products gets used for the first time, and requires the use of a program not identified to program control, the environment now becomes dirty

- Moral: Your change control procedures must involve your system programmers and security administrators, who must be aware of basic program control concepts

Controlled environments ...

- Messages you will know and love

- ICH417I THE ENVIRONMENT IS NOT CONTROLLED. CONDITIONAL ACCESS LIST BYPASSED FOR DATA SET *dsname*
- ICH418I CONDITIONAL ACCESS LIST FOR DATA SET *dsname* DID NOT GRANT AUTHORITY TO PROGRAM(S): *program-name1*, *program-name2*, *program-name3* ...
- ICH419I THE ENVIRONMENT IS NOT CONTROLLED. ATTEMPT TO LOAD PROGRAM *program-name* FROM LIBRARY *dsname* FAILED.
- ICH420I PROGRAM *program-name*, FROM {[LIBRARY *dsname*] | LPA | JPA | IDENTIFY } CAUSED THE ENVIRONMENT TO BECOME UNCONTROLLED.
- ICH421I REASON FOR UNCONTROLLED ENVIRONMENT IS NOT KNOWN.
- ICH422I THE ENVIRONMENT CANNOT BECOME UNCONTROLLED.
- ICH423I RACF EXECUTE-CONTROLLED PROGRAMS ARE ACTIVE: *program-name1*,*program-name2*, *program-name3* …
- ICH424I DATA SETS OPENED USING RACF WHEN(PROGRAM(...)) ARE STILL OPEN: *dsname1*, *dsname2*, *dsname3* …

- Etc. Note there may also be MVS (CSV...) and UNIX (BPX...) messages

PADS: Program Access to Data Sets

- Fancy term for putting a conditional access list entry on a DATASET profile that specifies a specific program
- Where the program mediates the user's access for a specific function

```
PERMIT XYZCORP.PAYROLL.DATA CLASS(DATASET) ID(HRFOLKS)
       ACCESS(UPDATE) WHEN(PROGRAM(HRAPPL))
```

- Requires HRAPPL to be under program control
  - Separate WHEN clause required for an alias, if one exists (which we have already said must also be under program control)
- And since **any other program in the environment also shares the requested access**, all of these programs must also be permitted with a WHEN entry. Unless:
  - the program is defined with the 3rd qualifier of the member string as "NOPADCHK"
  - which is why you almost exclusively see NOPADCHK in the wild

PADS: Program Access to Data Sets ...

- Let's say there is some internal structure of the HRAPPL application that you'd rather not have to know. All you know is that you run it via a batch job:

```
//HRJOB   EXEC PGM=HRAPPL
```

- But in reality, HRAPPL calls an internal program named HRPROG2 which calls another named HRPROG3 which actually opens the PADS-protected data set named XYZCORP.PAYROLL.DATA.
- Which program(s) do you specify on WHEN(PROGRAM(x))?
- Answer in the olden days: HRPROG3
  - But how on earth would you know to do that?
- Answer in modern times: Any one of them
  - But HRAPPL would certainly make the most sense

Program Access to SERVAUTH

- The SERVAUTH class is used to define a “security zone” which generally protects an IP address, or range of IP addresses
- Similar to PADS, any program in the address space would be able to use an open socket
- A network admin typically uses a program like ping or traceroute to perform network diagnostics, but that doesn't mean you want them reading all the traffic or injecting messages into it
- So, like PADS, you can conditionally allow access based on the running program
- In fact, the only real difference from PADS is that there isn't a NO|PADCHK-like qualifier, and so all programs in the environment would need a WHEN clause

A word about TSO/E and ISPF

- In an interactive environment where lots of unpredictable programs might be running at any given time, wouldn't it be nearly impossible to meaningfully maintain a controlled environment?
  - Answer: **Yes!** Programs requiring a clean environment are best implemented via batch or UNIX
- But I know that ping and traceroute have TSO command versions, and the RACF documentation uses them as examples, so what's up with that?
  - TSO has the capability of setting up a parallel safe environment
  - You can use this yourself when invoking a program:
    - **`TSOEXEC`** `CALL 'LIBRARY.NAME(PROG1)'`
    - There is a corresponding IKJEFTSR service
  - But when a command/program is listed in the AUTHCMD/AUTHPGM list in IKJTSOxx, TSO handles it for you.
    - z/OS Communication Server documents to put PING and TRACERTE in the AUTHCMD list

Building an addition

- Make house/safe/guests analogy completely absurd here ...

UNIX Servers and Daemons

- What might be “sensitive” in a UNIX context?
  - When running a server or daemon at the “z/OS UNIX level” of security, you have defined BPX.SERVER and/or BPX.DAEMON in the FACILITY class to control server and daemon functions
  - You have permitted with READ access those (non-human) user IDs under whose authority the application will run
  - The application user now has the powerful ability to switch its identity to that of any z/OS UNIX user
  - Which means that any program running in that address space has this ability

Anything sound familiar?

Enhanced mode program control

- So you have identified all your trusted programs and life is good.
- However, will the trusted programs always be invoked the way they were intended to be invoked?
  - HRAPPL calls HRPROG2 calls HRPROG3 which opens a PADS-protected data set
  - The user is supposed to invoke HRAPPL
  - But what if the user invokes HRPROG3 directly?
    - Was it written to expect this?
    - Was it written to protect against ill effects in this case?
    - Do you want to have to know?
- Better to simply require that the environment be “predictable”, or “as intended”

Enhanced mode program control ...

- Identify each program intended to be the main (top-level) program running in sensitive environments. You must use a “discrete” PROGRAM profile

```
RDEFINE PROGRAM HRAPPL APPLDATA('MAIN')
                ADDMEM('XYZCORP.LINKLIB'//NOPADCHK)
```

- Activate enhanced program control. You'll want to start in warning mode.

```
RDEFINE FACILITY IRR.PGMSECURITY APPLDATA('ENHANCED-WARNING')
SETROPTS WHEN(PROGRAM) REFRESH
```

- Verify

```
SETROPTS LIST
ATTRIBUTES = INITSTATS WHEN(PROGRAM -- ENHANCED WARNING) ...
```

- Now, the sensitive functions will not be allowed unless running in a MAIN environment (well, once you're beyond warning mode, anyway)

Enhanced mode program control ...

- Messages you will know and love

- ICH426I NON-MAIN PROGRAM IS IN CONTROL. CONDITIONAL ACCESS LIST BYPASSED FOR DATA SET dsname
- ICH427I NON-MAIN PROGRAM IS IN CONTROL. TEMPORARY USE OF CONDITIONAL ACCESS LIST ALLOWED FOR DATA SET dsname
- ICH428I PROGRAM *program-name* FROM { *library-name* | LPA | JPA | IDENTIFY} ESTABLISHED THE CURRENT EXECUTION ENVIRONMENT
- ICH429I NON-MAIN PROGRAM IS IN CONTROL. ATTEMPT TO LOAD PROGRAM *program-name* FROM LIBRARY *library-name* FAILED.
- ICH430I NON-MAIN PROGRAM IS IN CONTROL. TEMPORARY USE OF PROGRAM *program-name* FROM LIBRARY *library-name* ALLOWED.

Enhanced program control in UNIX …

- Must be enabled by defining the discrete BPX.MAINCHECK profile in the FACILITY class
- Any UNIX programs acting as MAIN must be moved into an MVS library, if they aren't already in one
  - See backup slides for gory details

Good sources of information:

- RACF Security Administrator's Guide
  - Chapter: Protecting programs

- UNIX System Services Planning manual
  - Chapter: Establishing UNIX security
    - Topic: Setting up for daemons
    - Topic: Preparing security for servers
    - Topic: Using enhanced program security

- Example: Communication Server addresses program control
  - IP Configuration Guide
    - IP configuration overview

      Topic: Program control

- Presentation: The World of Program Control and PADS

  ftp://public.dhe.ibm.com/eserver/zseries/zos/racf/pdf/r05_program_control.pdf

Congratulations, **You're the expert now!**

Backup slides

- These didn't make the cut, but might still have useful information

The easy stuff … restricting program execution by system

- PROGRAM is the only class for which you can specify a system ID in a conditional access list entry

```
PERMIT MYPROG CLASS(PROGRAM) ID(SOMEGRP) ACCESS(READ)
                                          WHEN(SYSID(SYSA))
```

- Where “SYSA” is the SID value specified in the active SMFPRM*xx* member of SYS1.PARMLIB.

Controlled environments

- A *controlled* (or *clean*) environment is one in which only programs defined in the PROGRAM class have run.
- Where:
  - An *environment* means:
    - A TSO session
    - A job step in a batch job
    - A UNIX address space (more on UNIX later)
  - A *program* is
    - A load module residing in a a partitioned data set (PDS)
    - A program object residing in a program library (PDS/E)
    - Not:
      - ✔ a UNIX executable file (more on UNIX later)
      - ✔ an interpreted program like PERL, java, TSO/E CLIST, or (uncompiled) Rexx
      - ✔ a program in the link pack area (LPA). Well, these **are** programs, but are automatically treated as trusted by the system

PADS: Program Access to Data Sets ...

- Use the following technique so that you need to maintain only one access list: that of the PROGRAM profile
  - Allow "**anybody**" to access the dataset while running the program

```
PERMIT XYZCORP.PAYROLL.DATA CLASS(DATASET) ID(*)
       ACCESS(UPDATE) WHEN(PROGRAM(HRAPPL))
```

  - Use a specific PROGRAM profile to limit who can run the program

```
PERMIT HRAPPL CLASS(PROGRAM) ID(HRFOLKS) ACCESS(READ)
```

PADS: Program Access to Data Sets ...

- **Tip:** When starting from scratch, to determine the name of the program to use in a WHEN(PROGRAM(*x*)) entry for a data set you wish to protect with PADS
  - Add a dummy entry

```
PERMIT XYZCORP.PAYROLL.DATA CLASS(DATASET) ID(HRFOLKS)
       ACCESS(UPDATE) WHEN(PROGRAM(FAKEPROG))
```

  - Attempt to access the data set using the *intended* program (MYPROG in this example) from a user without any current access
  - Look on the console for the resulting messages

```
ICH418I CONDITIONAL ACCESS LIST FOR DATA SET XYZCORP.PAYROLL.DATA DID
         NOT GRANT AUTHORITY TO PROGRAM(S): MYPROG
 ICH408I USER(TSOUSR5 ) GROUP(SYS1    ) NAME(####################)
   XYZCORP.PAYROLL.DATA CL(DATASET ) VOL(MYVOL1)
   INSUFFICIENT ACCESS AUTHORITY
   ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
```

  - ICH418I will **not** be issued unless there is at least one conditional PROGRAM access list entry!

Execute controlled libraries and programs

- READ access to a PROGRAM profile grants authority to execute the program, and, well, to ***read*** it!
- READ access to a library (data set) allows you to run programs from that library, and, well, to ***read*** its contents! (including copying it)
- If your program contains 'special sauce', you may want somebody to be able to execute it, but not read it
- You can grant EXECUTE access to PROGRAM and DATASET profiles to accomplish this!
- But guess what? Any program running in your environment can “see” the code in storage, and copy it somewhere
- So, you guessed it! Your environment must be clean when execute-controlled programs are active!
  - Note that there are no conditional access list entries involved here

Execute controlled libraries and programs ...

- Grant execute access to the private program library

```
PERMIT 'PRIVATE.LIBRARY' GENERIC ID(JOE) ACCESS(EXECUTE)
SETROPTS GENERIC(DATASET) REFRESH
```

- You must create a “discrete” PROGRAM profile for each program in PRIVATE.LIBRARY and grant READ or EXECUTE access

```
RDEFINE PROGRAM PLPROG1 ADDMEM('PRIVATE.LIBRARY'//NOPADCHK)
                        UACC(EXECUTE)
SETROPTS WHEN(PROGRAM) REFRESH
```

- Considerations are slightly different for LINKLIB programs, depending on how they are called

Why not just put the APF concatenation in PROGRAM **?

- After all, we trust it. And doing so might avoid some future outage.
- True, but blindly adding your APF concatenation to PROGRAM ** increases your risk surface.
  - That is, even though the software is trusted by definition, it is not necessarily required in a given clean environment (Least Privilege)
  - It might actually be dangerous in such an environment (think debugger)
  - And it provides that many more programs for somebody to try to invoke in ways they weren't intended to be invoked
- Though this can perhaps be mitigated with the use of ENHANCED mode

Additional UNIX concepts

- UNIX servers and daemons must also operate in a controlled environment
- But UNIX executable files cannot be defined to the PROGRAM class
- So we have to introduce a couple of new topics

UNIX concepts … defining program controlled executables

- There are actually three different ways

1) Turn on the program-controlled extended file attribute

2) Turn on the *sticky bit*
   - This means that the file is simply a reference to a standard MVS program in a standard MVS library located through the standard search order and defined in the PROGRAM class
   - This does limit the program name to 8 upper case characters
   - Some IBM software ships this way, but you can use the technique for your own programs

3) Create an external link from the UNIX file to an MVS program name
   - Same considerations apply

UNIX concepts … defining program controlled executables

1) Turn on the program-controlled extended file attribute

```
extattr +p /bin/wherever/hrappl
ls -E /bin/wherever/hrappl
-rwxr--r--  -ps-
```

2) Turn on the *sticky bit*

```
ls -l /bin/wherever/HRAPPL
-rwxr--r--
chmod +t /bin/wherever/HRAPPL
ls -l /bin/wherever/HRAPPL
-rwxr--r-T
```

3) External link

```
ln -e HRAPPL /bin/wherever/hrappl
ls -l /bin/wherever/hrappl
erwxr--r--
```

UNIX concepts … Tying up some loose ends

- Enough with this artificial boundary between MVS and UNIX. It doesn't really exist.
  - A UNIX program could try to
    - access a PADS-protected data set
    - call an execute-controlled MVS program
    - open a program-controlled socket
  - An MVS program could
    - Invoke a UNIX service to switch UNIX identity
- All rules apply everywhere
- But if a UNIX program wants to do one of the “MVS program-ish” things, it cannot. It must be moved into an MVS program library.
  - Exception: Note that UNIX provides basic read/execute protection with permission bits and access control lists. UNIX also provided auditing support with file-level audit settings. None of this is considered “program control”.

UNIX concepts … Tying up some loose ends

- For UNIX daemons, if you want only UNIX files to be checked for program control, and do not want programs loaded from MVS libraries to be checked, you can set up BPX.DAEMON.HFSCTL in the FACILITY class
  - Not recommended because doing this weakens some of the security provided by the BPX.DAEMON resource
  - However, if you do not already run with BPX.DAEMON but want to gain only a subset of the benefits of running with BPX.DAEMON, it does represent a step forward