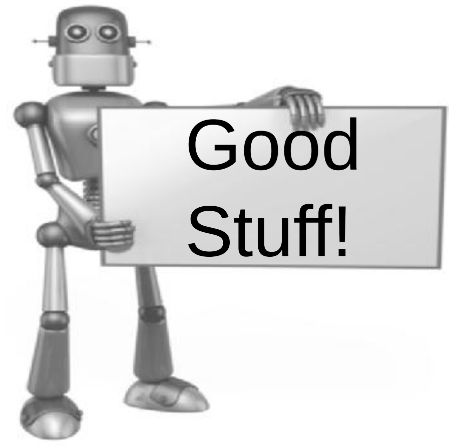
The Business, Science and Uses of ILE Service Programs



Charles Guarino
Twitter @charlieguarino
Central Park Data Systems, Inc.

About The Speaker

With an IT career spanning over 30 years, Charles Guarino has been a consultant for most of them. Since 1995 he has been founder and President of Central Park Data Systems, Inc., a New York area based IBM midrange consulting company. In addition to being a professional speaker, he is a frequent contributor of technical and strategic articles and webcasts for the IT community. He is a proud member of COMMON's Speaker Excellence Hall of Fame and also Long Island Software and Technology Network's Twenty Top Techies of 2009. Charles currently serves as a member of COMMON's Strategic Education Team (SET) and is also Immediate Past President and monthly Q&A host of LISUG, a Long Island IBM i User's Group www.lisug.org.
Charles can be reached at cguarino@centralparkdata.com.
LinkedIn - http://www.linkedin.com/in/guarinocharles
Twitter - @charlieguarino

In the beginning, your system is stable with no major incidents

ItemInq

ItemMaint

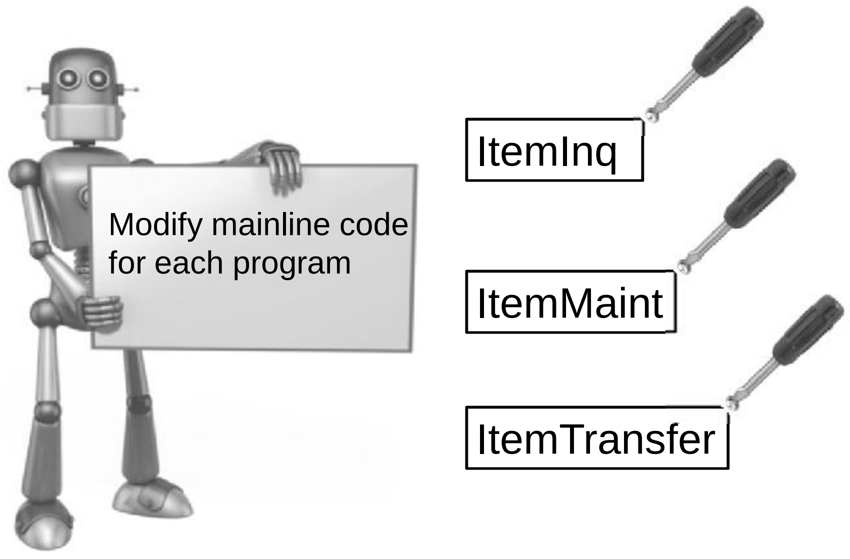
ItemTransfer

Business Disruption !!!

This merger brings new requirements to our programs

1) Unit of measure quantity conversions
2) Proprietary industry-specific conversion and product mixing routines
3) Need to make these conversions available to all programs with the exception of the proprietary conversion routines.
4) Additional requirements to be defined in the future
5) We need to expose these routines down the road to external users via a web service

Technique #1



Technique #1

Advantages

This space intentionally left blank

Disadvantages

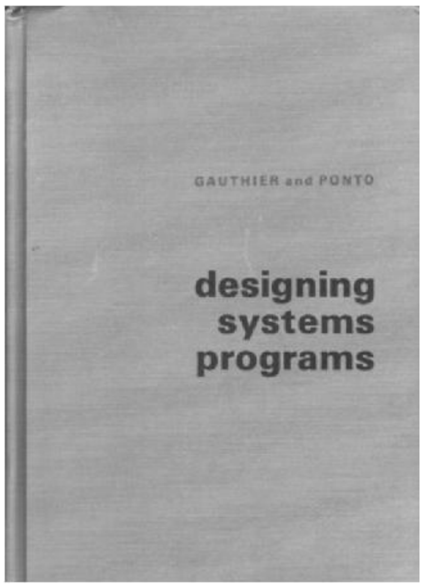
- Potential conflict with other global variables
- Add'l modifications may not be consistent across programs
- Original modification may be diluted as new functionality is added to each program
- Cannot protect certain subprocedures from being executed

Technique #2 - Modularize

mod·u·lar·ize [moj-uh-luh-rahyz]

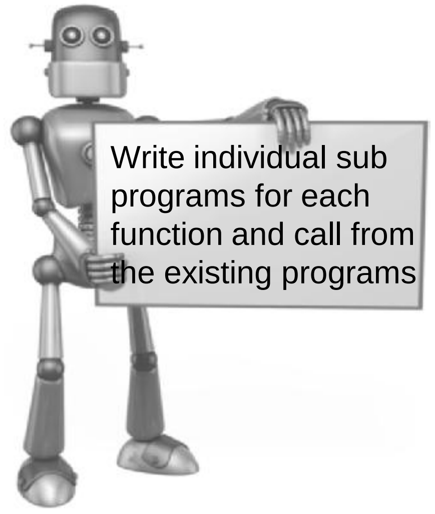
verb (used with object), mod·u·lar·ized, mod·u·lar·iz·ing. to form or organize into modules, as for flexibility.

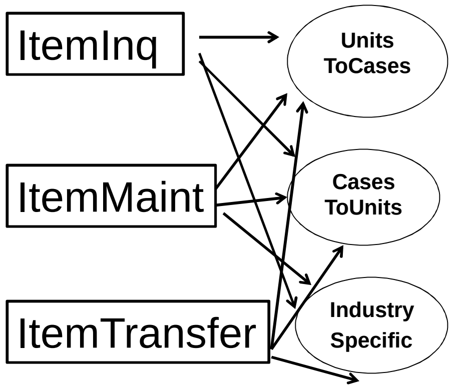
Technique #2


GAUTHIER and PONTO

designing systems programs

Possible solution #1


Write individual sub programs for each function and call from the existing programs



Possible solution #1

Advantages

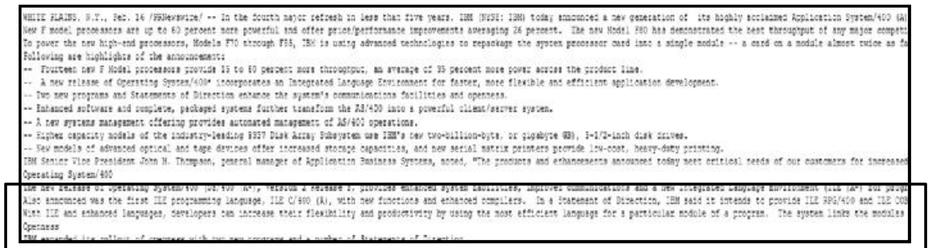
This space intentionally left blank

Disadvantages

- See disadvantages of possible solution #1
- Still a maintenance nightmare
- Potential performance issues with dynamic binding
- Can't take advantage of activation group scoping or recursive calls

February 16th, 1993 – A BIG announcement by IBM Do you remember where you were?

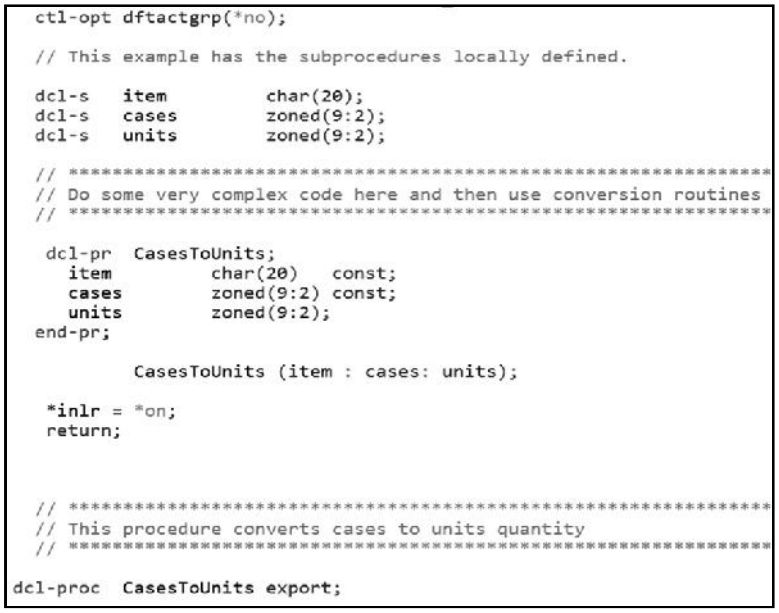
A NEW GENERATION: IBM BOOSTS AS/400 POWER UP TO 60 PERCENT, ENHANCES OPENESS, CLIENT SERVER CAPABILITIES



WHITE PLAINS, N.Y., Feb. 16 /PRNewswire/ -- In the fourth major refresh in less than five years, IBM (NYSE: IBM) today announced a new generation of its highly acclaimed Application System/400 (A
New F model processors are up to 60 percent more powerful and offer price/performance improvements averaging 26 percent. The new Model F80 has demonstrated the best throughput of any major competi
To power the new high-end processors, Models F70 through F95, IBM is using advanced technologies to repackage the system processor card into a single module -- a card on a module almost twice as fa
Following are highlights of the announcement:
-- Fourteen new F Model processors provide 15 to 60 percent more throughput, an average of 35 percent more power across the product line.
-- A new release of Operating System/400* incorporates an Integrated Language Environment for faster, more flexible and efficient application development.
-- Two new programs and Statements of Direction enhance the system's communications facilities and openness.
-- Enhanced software and complete, packaged systems further transform the AS/400 into a powerful client/server system.
-- A new systems management offering provides automated management of AS/400 operations.
-- Higher capacity models of the industry-leading 9337 Disk Array Subsystem use IBM's new two-billion-byte, or gigabyte GB), 3-1/2-inch disk drives.
-- New models of advanced optical and tape devices offer increased storage capacities, and new serial matrix printers provide low-cost, heavy-duty printing.
IBM Senior Vice President John H. Thompson, general manager of Application Business Systems, noted, "The products and enhancements announced today meet critical needs of our customers for increased
Operating System/400
The new release of Operating System/400 (OS/400 (R)*), Version 2 Release 3, provides enhanced system facilities, improved communications and a new Integrated Language Environment (ILE (R*) for progr
Also announced was the first ILE programming language, ILE C/400 (R), with new functions and enhanced compilers. In a Statement of Direction, IBM said it intends to provide ILE RPG/400 and ILE COB
With ILE and enhanced languages, developers can increase their flexibility and productivity by using the most efficient language for a particular module of a program. The system links the modules
Openness
IBM expanded its rollout of openness with two new programs and a number of Statements of Direction

"The new release of Operating System/400 Version 2 Release 3 provides enhanced system facilities, … and a new Integrated Language Environment (ILE) for programming languages.
Also announced was the first ILE programming language, ILE C/400…
In a Statement of Direction, IBM said it intends to provide ILE RPG/400 and ILE/COBOL.
With ILE and enhanced languages, developers can increase their flexibility and productivity by using the most efficient language for a particular module of a program. "

Possible solution #2 - program ITEMINQA



```
  ctl-opt dftactgrp(*no);

  // This example has the subprocedures locally defined.

  dcl-s   item         char(20);
  dcl-s   cases        zoned(9:2);
  dcl-s   units        zoned(9:2);

  // ****************************************************************
  // Do some very complex code here and then use conversion routines
  // ****************************************************************

   dcl-pr  CasesToUnits;
     item         char(20)   const;
     cases        zoned(9:2) const;
     units        zoned(9:2);
  end-pr;

           CasesToUnits (item : cases: units);

   *inlr = *on;
   return;


  // ****************************************************************
  // This procedure converts cases to units quantity
  // ****************************************************************

dcl-proc  CasesToUnits export;
```

Possible solution #2

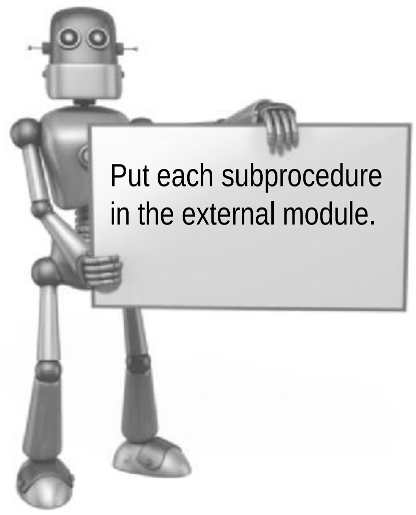
Advantages

Isolation of new program logic

Disadvantages

- See disadvantages of possible solution #1
- Still a maintenance nightmare

Possible solution #3 – Create external module



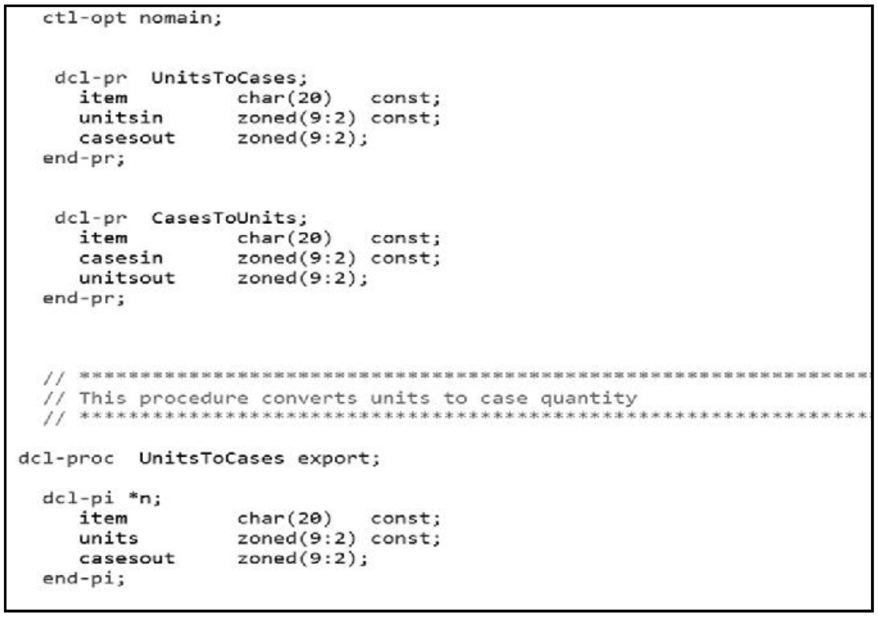
1) Units to cases conversion

2) Cases to units conversion

3) Industry specific algorithms

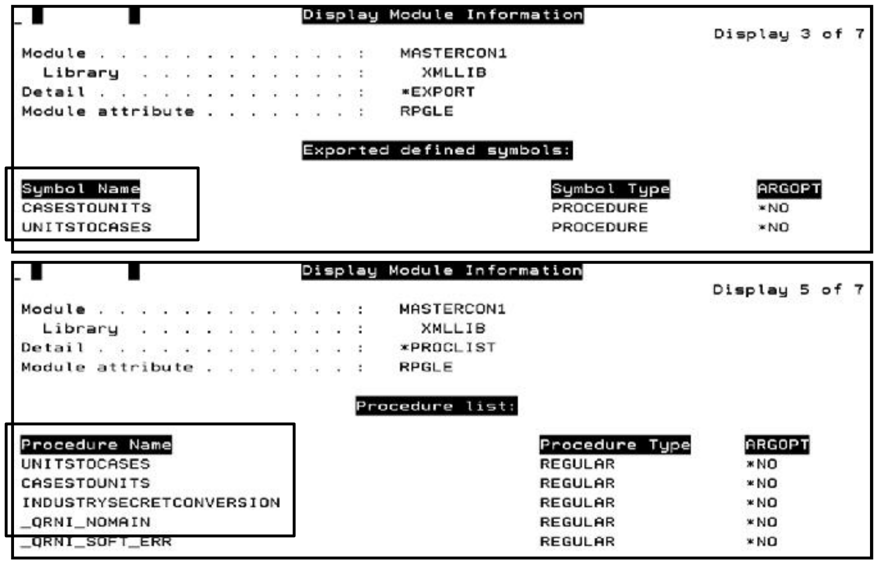
You will use CRTRPGMOD !!!

Possible solution #3 - Module MASTERCON1



```
  ctl-opt nomain;


   dcl-pr  UnitsToCases;
     item         char(20)   const;
     unitsin      zoned(9:2) const;
     casesout     zoned(9:2);
  end-pr;


   dcl-pr  CasesToUnits;
     item         char(20)   const;
     casesin      zoned(9:2) const;
     unitsout     zoned(9:2);
  end-pr;



  // *****************************************************************
  // This procedure converts units to case quantity
  // *****************************************************************

dcl-proc  UnitsToCases export;

  dcl-pi *n;
     item         char(20)   const;
     units        zoned(9:2) const;
     casesout     zoned(9:2);
  end-pi;
```

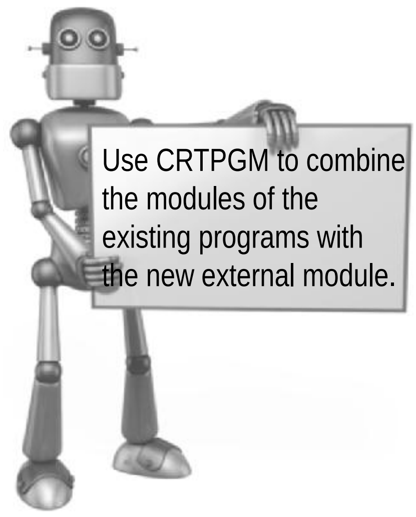
Possible solution #3 - DSPMOD MASTERCON1



```
                          Display Module Information
                                                          Display 3 of 7
Module . . . . . . . . . . . . . :   MASTERCON1
  Library  . . . . . . . . . . . :     XMLLIB
Detail . . . . . . . . . . . . . :   *EXPORT
Module attribute . . . . . . . . :   RPGLE

                          Exported defined symbols:

Symbol Name                                   Symbol Type        ARGOPT
CASESTOUNITS                                  PROCEDURE          *NO
UNITSTOCASES                                  PROCEDURE          *NO
```

```
                          Display Module Information
                                                          Display 5 of 7
Module . . . . . . . . . . . . . :   MASTERCON1
  Library  . . . . . . . . . . . :     XMLLIB
Detail . . . . . . . . . . . . . :   *PROCLIST
Module attribute . . . . . . . . :   RPGLE

                              Procedure list:

Procedure Name                                Procedure Type     ARGOPT
UNITSTOCASES                                  REGULAR            *NO
CASESTOUNITS                                  REGULAR            *NO
INDUSTRYSECRETCONVERSION                      REGULAR            *NO
_QRNI_NOMAIN                                  REGULAR            *NO
_QRNI_SOFT_ERR                                REGULAR            *NO
```

Possible solution #3a – Bind modules w/static binding

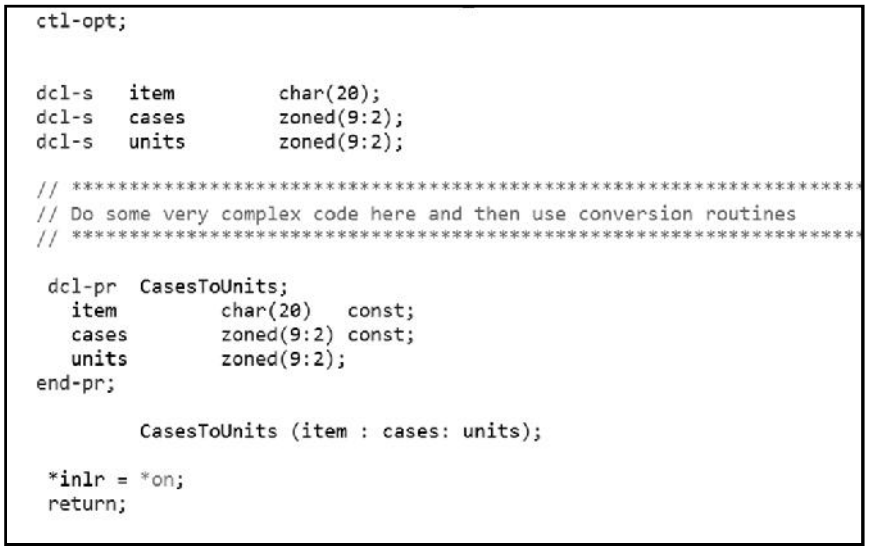

Use CRTPGM to combine the modules of the existing programs with the new external module.

Module ITEMINQB

+ Module MASTERCON1

= Program ITEMINQB

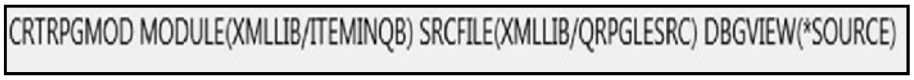
Possible solution #3a - Module ITEMINQB



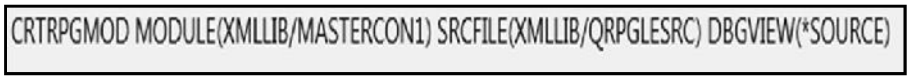
```
ctl-opt;


dcl-s   item        char(20);
dcl-s   cases       zoned(9:2);
dcl-s   units       zoned(9:2);

// ***********************************************************************
// Do some very complex code here and then use conversion routines
// ***********************************************************************

 dcl-pr  CasesToUnits;
   item         char(20)   const;
   cases        zoned(9:2) const;
   units        zoned(9:2);
end-pr;

         CasesToUnits (item : cases: units);

 *inlr = *on;
 return;
```

Possible solution #3a

Create the modules

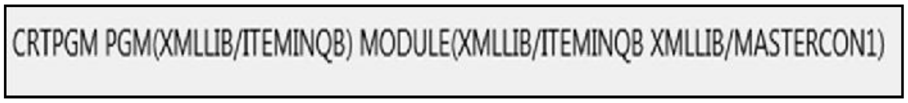


```
CRTRPGMOD MODULE(XMLLIB/ITEMINQB) SRCFILE(XMLLIB/QRPGLESRC) DBGVIEW(*SOURCE)
```



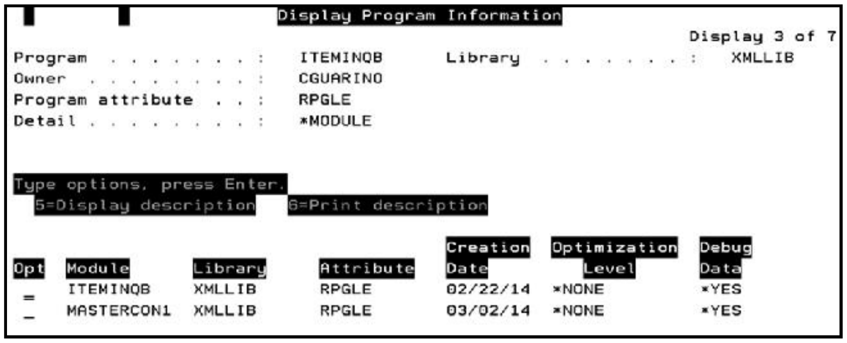
```
CRTRPGMOD MODULE(XMLLIB/MASTERCON1) SRCFILE(XMLLIB/QRPGLESRC) DBGVIEW(*SOURCE)
```

Create the program

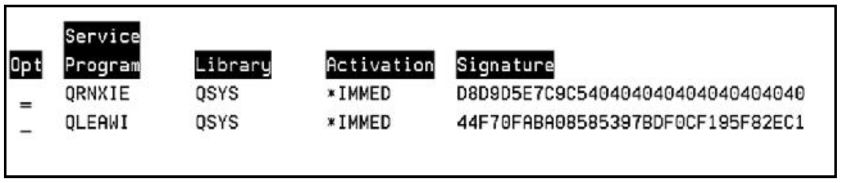


```
CRTPGM PGM(XMLLIB/ITEMINQB) MODULE(XMLLIB/ITEMINQB XMLLIB/MASTERCON1)
```

Possible solution #3a - DSPPGM ITEMINQB



```
                              Display Program Information
                                                                   Display 3 of 7
Program  . . . . . . . . :   ITEMINQB      Library  . . . . . . . :   XMLLIB
Owner  . . . . . . . . . :   CGUARINO
Program attribute  . . . :   RPGLE
Detail . . . . . . . . . :   *MODULE


Type options, press Enter.
  5=Display description    6=Print description

                                                Creation  Optimization  Debug
Opt  Module       Library      Attribute       Date        Level       Data
 _   ITEMINQB     XMLLIB       RPGLE           02/22/14  *NONE         *YES
 _   MASTERCON1   XMLLIB       RPGLE           03/02/14  *NONE         *YES
```



```
     Service
Opt  Program    Library    Activation  Signature
 _   QRNXIE     QSYS       *IMMED      D8D9D5E7C9C540404040404040404040
 _   QLEAWI     QSYS       *IMMED      44F70FABA08585397BDF0CF195F82EC1
```

Possible solution #3a

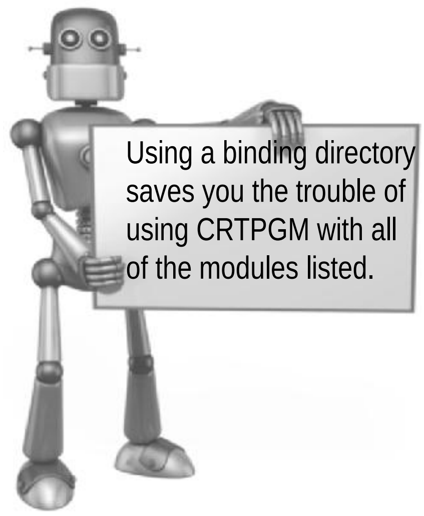
Advantages

Statically bound modules do execute more quickly

Disadvantages

- Newer hardware diminishes the performance argument
- Must bind *ALL modules to create usable programs
- Future modifications will require rebinding of *ALL programs

Possible solution #3b – Bind modules w/ binding directory


Using a binding directory saves you the trouble of using CRTPGM with all of the modules listed.

A Binding Directory is nothing more than a table of contents for your modules (and service programs).

It provides an easy way to reference these objects.

Possible solution #3b

Create the binding directory

```
CRTBNDDIR BNDDIR(XMLLIB/UTILITIES)
```

Add the modules as a directory entry

```
ADDBNDDIRE BNDDIR(XMLLIB/UTILITIES) OBJ((XMLLIB/MASTERCON1 *MODULE))
```

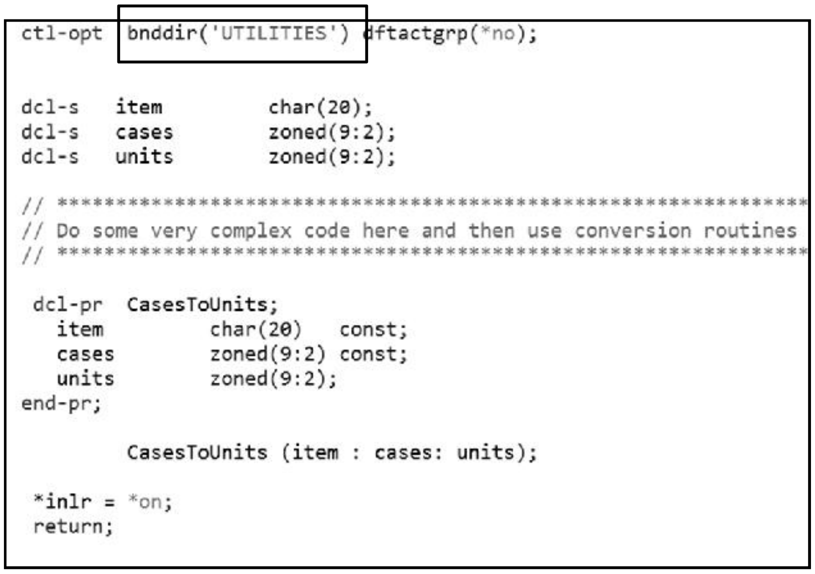
Reference the binding directory in your program

```
ctl-opt  bnddir('UTILITIES') dftactgrp(*no);
```

Compile your program, CRTPGM not required

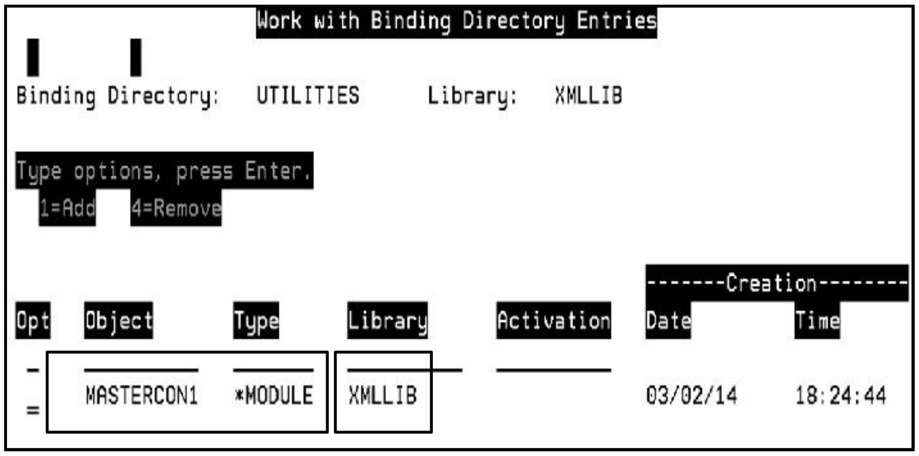
```
CRTBNDRPG PGM(XMLLIB/ITEMINQC) SRCFILE(XMLLIB/QRPGLESRC) DBGVIEW(*SOURCE)
```

Possible solution #3b - Program ITEMINQC



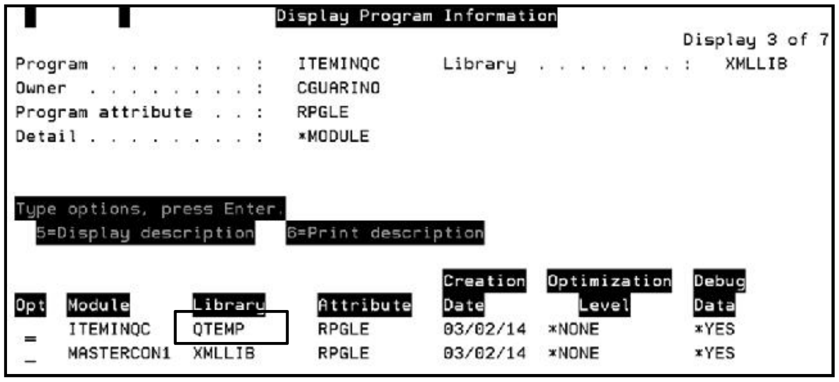
```
ctl-opt  bnddir('UTILITIES') dftactgrp(*no);


dcl-s   item        char(20);
dcl-s   cases       zoned(9:2);
dcl-s   units       zoned(9:2);

// *****************************************************************
// Do some very complex code here and then use conversion routines
// *****************************************************************

 dcl-pr  CasesToUnits;
   item         char(20)   const;
   cases        zoned(9:2) const;
   units        zoned(9:2);
end-pr;

         CasesToUnits (item : cases: units);

 *inlr = *on;
 return;
```

Possible solution #3b - WRKBNDDIRE UTILITIES

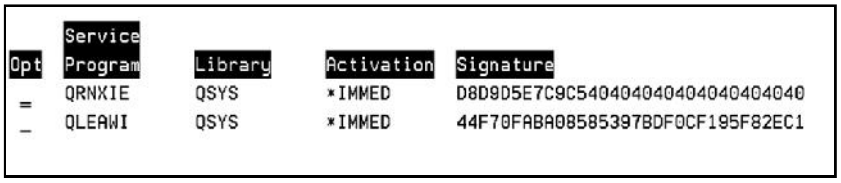


```
                         Work with Binding Directory Entries

Binding Directory:    UTILITIES      Library:    XMLLIB

Type options, press Enter.
  1=Add    4=Remove

                                                          -------Creation--------
Opt  Object         Type      Library      Activation    Date           Time
 _   __________     ________  __________   __________
 _   MASTERCON1     *MODULE   XMLLIB                     03/02/14       18:24:44
```

Possible solution #3b - DSPPGM ITEMINQC



```
                               Display Program Information
                                                                  Display 3 of 7
Program  . . . . . . . :   ITEMINQC      Library  . . . . . . . :   XMLLIB
Owner  . . . . . . . . :   CGUARINO
Program attribute  . . :   RPGLE
Detail . . . . . . . . :   *MODULE


Type options, press Enter.
  5=Display description   6=Print description

                                                 Creation  Optimization  Debug
Opt  Module      Library     Attribute   Date      Level         Data
 _   ITEMINQC    QTEMP       RPGLE       03/02/14  *NONE         *YES
 _   MASTERCON1  XMLLIB      RPGLE       03/02/14  *NONE         *YES
```



```
     Service
Opt  Program     Library     Activation  Signature
 _   QRNXIE      QSYS        *IMMED      D8D9D5E7C9C540404040404040404040
 _   QLEAWI      QSYS        *IMMED      44F70FABA08585397BDF0CF195F82EC1
```

Possible solution #3b

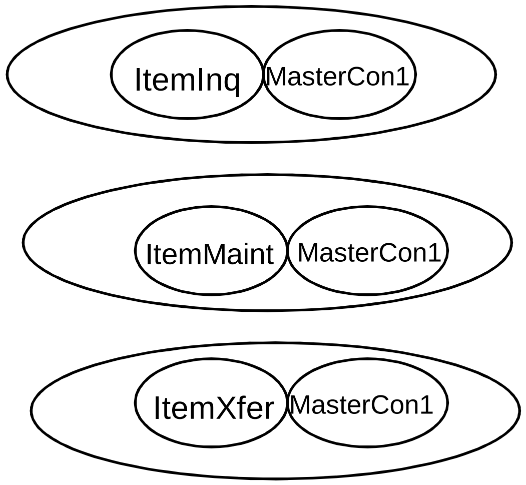
Advantages

- Statically bound modules do execute more quickly
- The binding directory makes it easier to compile programs

Disadvantages

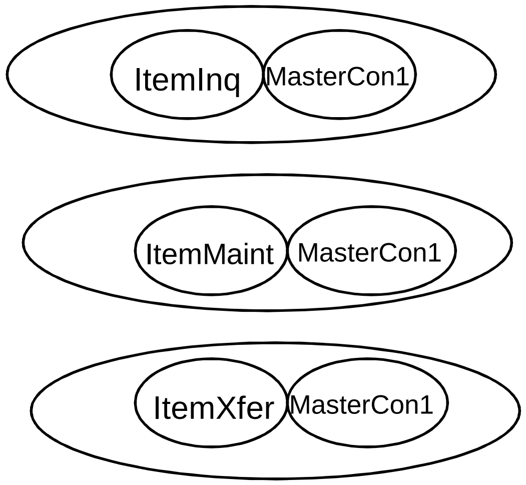
- Newer hardware diminishes the performance argument
- Must compile *ALL programs to include usable modules
- Future modifications will require recompiling of *ALL programs

In these scenarios, a separate copy of module MASTERCON1 is included in every program.

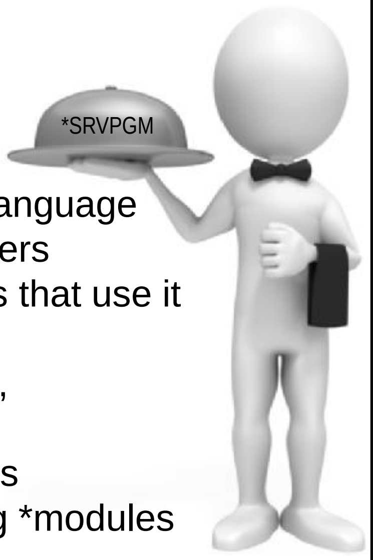


Business Disruption !!!

Now you need to RE-BIND *ALL !!!
Either Re-compile or use the UPDPGM command for EVERY pgm



Enter the Service Program



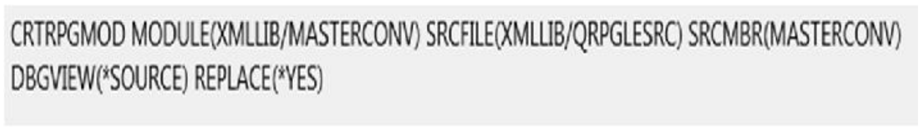
- Created from a module
- Cannot be directly executed
- Has an object type of *SRVPGM
- Used in conjunction with binder language
- Uses a **signature** to validate callers
- Don’t need to rebind all programs that use it when a change is made
- Are statically bound by reference, not statically bound by copy
- Can contain one or more modules
- I definitely prefer them over using *modules

The **simple*** start of a service program

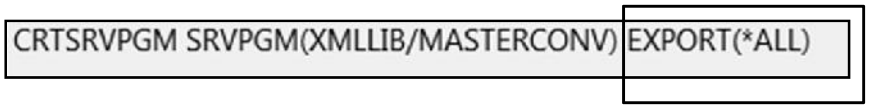
- CRTRPGMOD
- CRTSRVPGM using EXPORT *ALL
- A <u>current</u> signature is automatically assigned
- Add the service program to a binding directory
- Compile the program that references the service program
  (the compiled program will remember the current signature)
- Call your program and the service program is activated.
  (if a valid signature is not found you will have a violation)

* Simple but not always the best

The Steps…

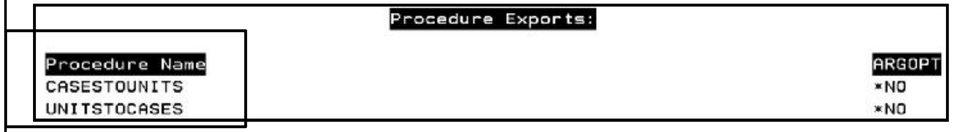


```
CRTRPGMOD MODULE(XMLLIB/MASTERCONV) SRCFILE(XMLLIB/QRPGLESRC) SRCMBR(MASTERCONV)
DBGVIEW(*SOURCE) REPLACE(*YES)
```

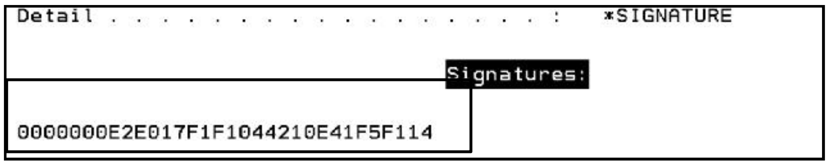


```
CRTSRVPGM SRVPGM(XMLLIB/MASTERCONV) EXPORT(*ALL)
```

DSPSRVPGM MASTERCONV

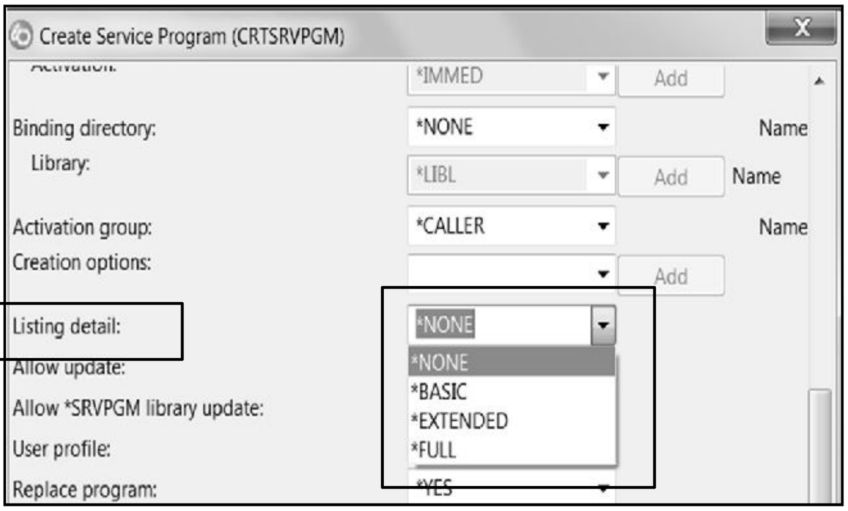


```
                         Procedure Exports:

Procedure Name                                                   ARGOPT
CASESTOUNITS                                                     *NO
UNITSTOCASES                                                     *NO
```



```
Detail . . . . . . . . . . . . . . . . . . :   *SIGNATURE

                                   Signatures:

0000000E2E017F1F1044210E41F5F114
```

The listing detail option….

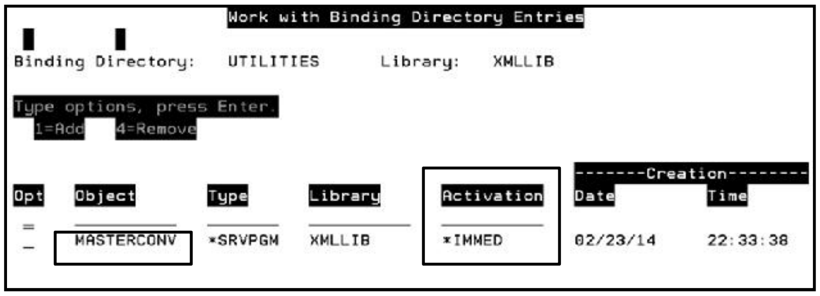


```
Create Service Program (CRTSRVPGM)
Activation:                    *IMMED          Add
Binding directory:             *NONE                   Name
  Library:                     *LIBL           Add     Name
Activation group:              *CALLER                 Name
Creation options:                              Add
Listing detail:                *NONE
Allow update:                  *NONE
Allow *SRVPGM library update:  *BASIC
User profile:                  *EXTENDED
Replace program:               *FULL
                               *YES
```

The Steps…

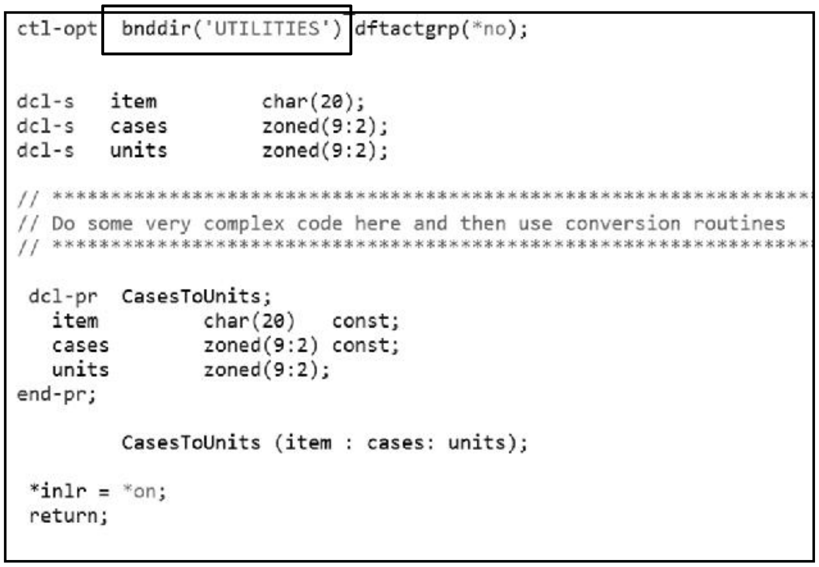
```
ADDBNDDIRE BNDDIR(XMLLIB/UTILITIES) OBJ((XMLLIB/MASTERCONV))
```

WRKBNDDIRE UTILITIES



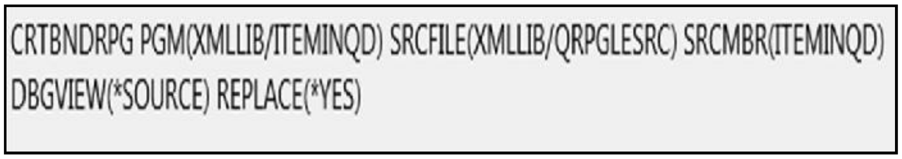
```
                    Work with Binding Directory Entries

Binding Directory:   UTILITIES      Library:   XMLLIB

Type options, press Enter.
  1=Add   4=Remove
                                                          -------Creation-------
Opt  Object       Type      Library     Activation   Date         Time
 _
 _   MASTERCONV   *SRVPGM   XMLLIB      *IMMED       02/23/14     22:33:38
```

The Steps… Program ITEMINQD



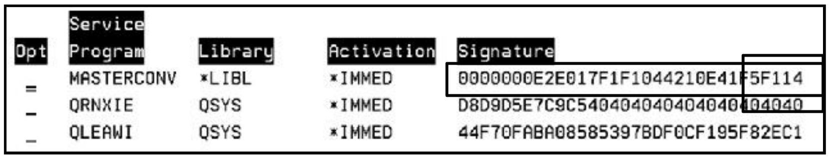
```
ctl-opt  bnddir('UTILITIES') dftactgrp(*no);


dcl-s   item        char(20);
dcl-s   cases       zoned(9:2);
dcl-s   units       zoned(9:2);

// *****************************************************************
// Do some very complex code here and then use conversion routines
// *****************************************************************

 dcl-pr  CasesToUnits;
   item         char(20)   const;
   cases        zoned(9:2) const;
   units        zoned(9:2);
end-pr;

         CasesToUnits (item : cases: units);

 *inlr = *on;
 return;
```

The Steps…



```
CRTBNDRPG PGM(XMLLIB/ITEMINQD) SRCFILE(XMLLIB/QRPGLESRC) SRCMBR(ITEMINQD)
DBGVIEW(*SOURCE) REPLACE(*YES)
```

DSPPGM ITEMINQD



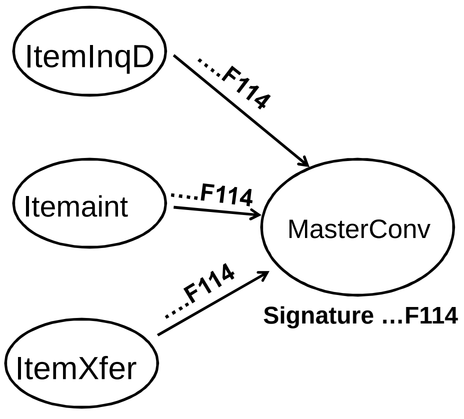
```
     Service
Opt  Program     Library     Activation  Signature
 =   MASTERCONV  *LIBL       *IMMED      0000000E2E017F1F1044210E41F5F114
 _   QRNXIE      QSYS        *IMMED      D8D9D5E7C9C5404040404040404040404040
 _   QLEAWI      QSYS        *IMMED      44F70FABA08585397BDF0CF195F82EC1
```

About that signature…

[illegible]

A.K.A. 0000000E2E017F1F1044210E41F5F114

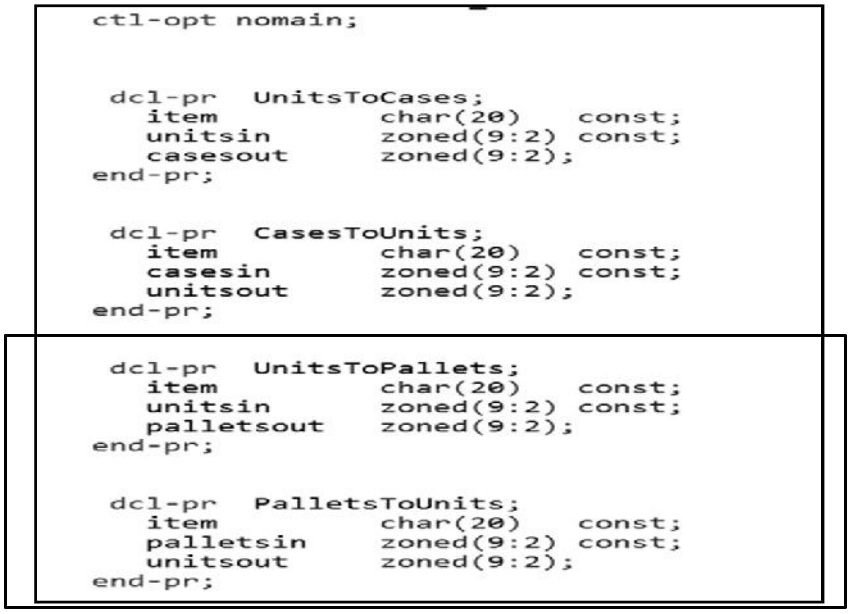
In these scenarios, as long as the signature matches when the service program is called, everything will be fine.



[illegible] [illegible]

Business Disruption !!!

Added two more procedures to MASTERCONV



```
ctl-opt nomain;


 dcl-pr  UnitsToCases;
   item          char(20)    const;
   unitsin       zoned(9:2)  const;
   casesout      zoned(9:2);
end-pr;

 dcl-pr  CasesToUnits;
   item          char(20)    const;
   casesin       zoned(9:2)  const;
   unitsout      zoned(9:2);
end-pr;

 dcl-pr  UnitsToPallets;
   item          char(20)    const;
   unitsin       zoned(9:2)  const;
   palletsout    zoned(9:2);
end-pr;

 dcl-pr  PalletsToUnits;
   item          char(20)    const;
   palletsin     zoned(9:2)  const;
   unitsout      zoned(9:2);
end-pr;
```

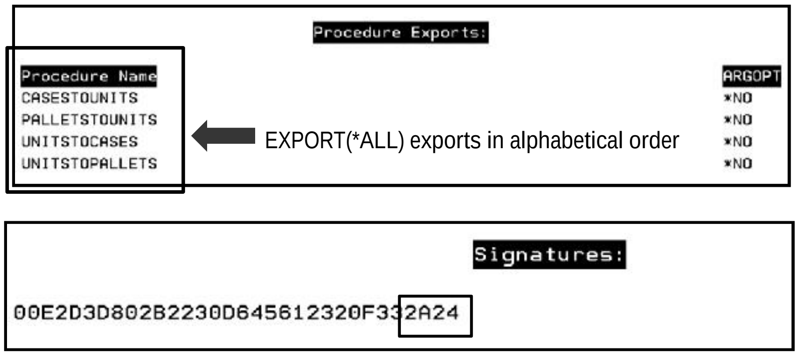
What happens to the signature?

```
CRTSRVPGM SRVPGM(XMLLIB/MASTERCONV) EXPORT(*ALL)
```

[illegible]

[illegible]

DSPSRVPGM MASTERCONV



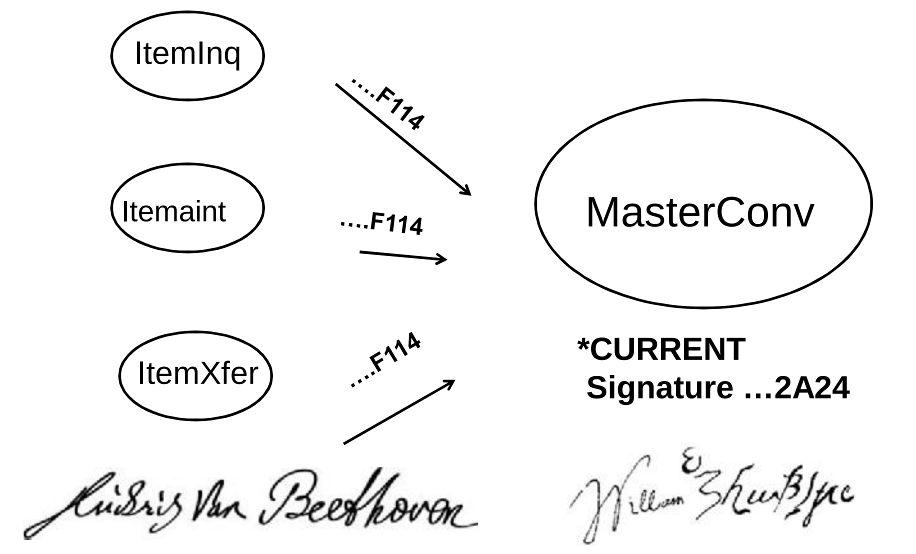
```
                    Procedure Exports:

Procedure Name                                    ARGOPT
CASESTOUNITS                                      *NO
PALLETSTOUNITS                                    *NO
UNITSTOCASES                                      *NO
UNITSTOPALLETS                                    *NO
```

EXPORT(*ALL) exports in alphabetical order

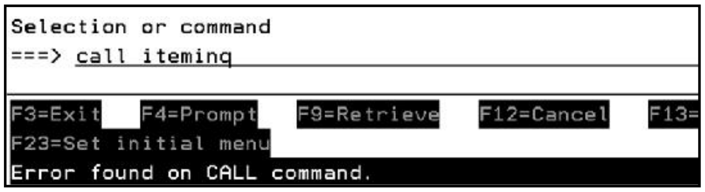
```
                          Signatures:

00E2D3D802B2230D645612320F332A24
```

[illegible]

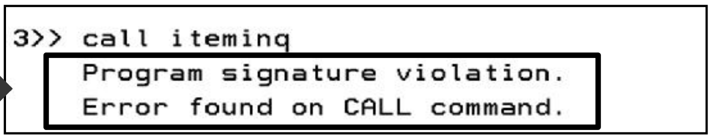
What happens now that the three programs who know “Beethoven” call a “Shakespeare” service program?



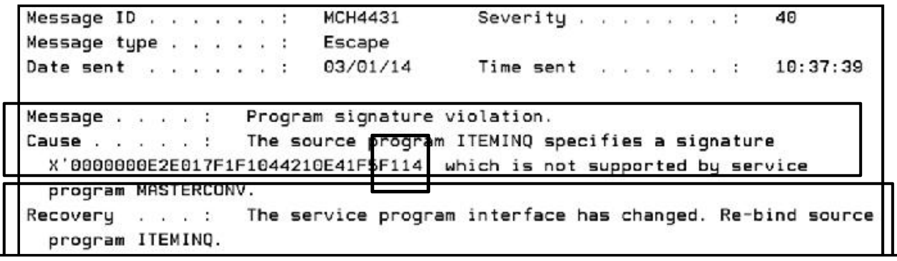
How do you fix this?



```
Selection or command
===> call iteminq

F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel   F13=
F23=Set initial menu
Error found on CALL command.
```



```
3>> call iteminq
     Program signature violation.
     Error found on CALL command.
```



```
Message ID . . . . . . :   MCH4431       Severity . . . . . . . :   40
Message type . . . . . :   Escape
Date sent  . . . . . . :   03/01/14      Time sent  . . . . . . :   10:37:39

Message . . . . :   Program signature violation.
Cause . . . . . :   The source program ITEMINQ specifies a signature
  X'0000000E2E017F1F1044210E41F5F114' which is not supported by service
  program MASTERCONV.
Recovery  . . . :   The service program interface has changed. Re-bind source
  program ITEMINQ.
```

Possible signature violation solution # 1

```
UPDPGM PGM(XMLLIB/ITEMINQ) MODULE(*NONE)
```

```
UPDPGM PGM(XMLLIB/ITEMMAINT) MODULE(*NONE)
```

```
UPDPGM PGM(XMLLIB/ITEMXFER) MODULE(*NONE)
```

This solution works if you can identify ***EVERY*** program referencing the service program.

Also, if done manually will be a VERY tedious process.

Possible signature violation solution # 2

Retrieve the binder source

BEFORE you make any changes !!!

```
RTVBNDSRC SRVPGM(XMLLIB/MASTERCONV) SRCFILE(XMLLIB/QSRVSRC)
```

```
STRPGMEXP  PGMLVL(*CURRENT)
EXPORT     SYMBOL('CASESTOUNITS')
EXPORT     SYMBOL('UNITSTOCASES')
ENDPGMEXP
```

About that signature…

```
CRTSRVPGM SRVPGM(XMLLIB/MASTERCONV) EXPORT(*ALL)
```

(When only two procedures were being exported)

Will produce the same <u>current</u> signature as

```
CRTSRVPGM SRVPGM(XMLLIB/MASTERCONV) SRCFILE(XMLLIB/QSRVSRC)
```

```
STRPGMEXP   PGMLVL(*CURRENT)
                 EXPORT      SYMBOL('CASESTOUNITS')
                 EXPORT      SYMBOL('UNITSTOCASE')
ENDPGMEXP
```

“0000000E2E017F1F1044210E41F5F114”

[illegible]

For your <u>existing</u> bound programs to continue working the signature still needs to match

```
STRPGMEXP   PGMLVL(*CURRENT)
  EXPORT    SYMBOL(CasesToUnits)
  EXPORT    SYMBOL(UnitsToCases)
  EXPORT    SYMBOL(UnitsToPallets)
  EXPORT    SYMBOL(PalletsToUnits)
ENDPGMEXP

STRPGMEXP   PGMLVL(*PRV)
  EXPORT    SYMBOL('CASESTOUNITS')
  EXPORT    SYMBOL('UNITSTOCASES')
ENDPGMEXP
```

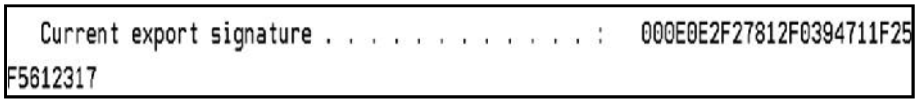
New

“0000000E2E017F1F1044210E41F5F114” will now become the *PRV

A <u>NEW</u> signature will be generated with the *CURRENT export

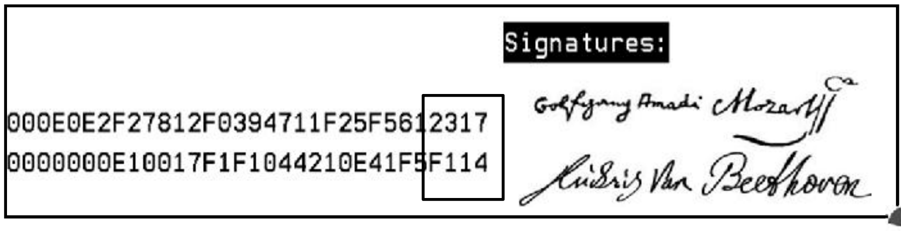
Now service program MASTERCONV has TWO signatures!

```
CRTSRVPGM SRVPGM(XMLLIB/MASTERCONV) SRCFILE(XMLLIB/QSRVSRC)
```



```
  Current export signature . . . . . . . . . . . . . :   000E0E2F27812F0394711F25
F5612317
```

DSPSRVPGM MASTERCONV DETAIL(*SIGNATURE)



```
                                        Signatures:
000E0E2F27812F0394711F25F5612317        Wolfgang Amadè Mozart
0000000E10017F1F1044210E41F5F114        Ludwig van Beethoven
```

Now MASTERCONV has two signatures

MASTERCONV

*CURRENT ...2317 Wolfgang Amadè Mozart

*PRV   …F114   Ludwig van Beethoven

May I see your ID?

Different *CURRENT signatures, SAME *PRV signatures

```
STRPGMEXP   PGMLVL(*CURRENT)
  EXPORT    SYMBOL(CasesToUnits)
  EXPORT    SYMBOL(UnitsToCases)
  EXPORT    SYMBOL(UnitsToPallets)
  EXPORT    SYMBOL(PalletsToUnits)
ENDPGMEXP

STRPGMEXP   PGMLVL(*PRV)
  EXPORT    SYMBOL('CASESTOUNITS')
  EXPORT    SYMBOL('UNITSTOCASES')
ENDPGMEXP
```

```
STRPGMEXP   PGMLVL(*CURRENT)
  EXPORT    SYMBOL(UnitsToPallets)
  EXPORT    SYMBOL(PalletsToUnits)
  EXPORT    SYMBOL(CasesToUnits)
  EXPORT    SYMBOL(UnitsToCases)
ENDPGMEXP

STRPGMEXP   PGMLVL(*PRV)
  EXPORT    SYMBOL('CASESTOUNITS')
  EXPORT    SYMBOL('UNITSTOCASES')
ENDPGMEXP
```

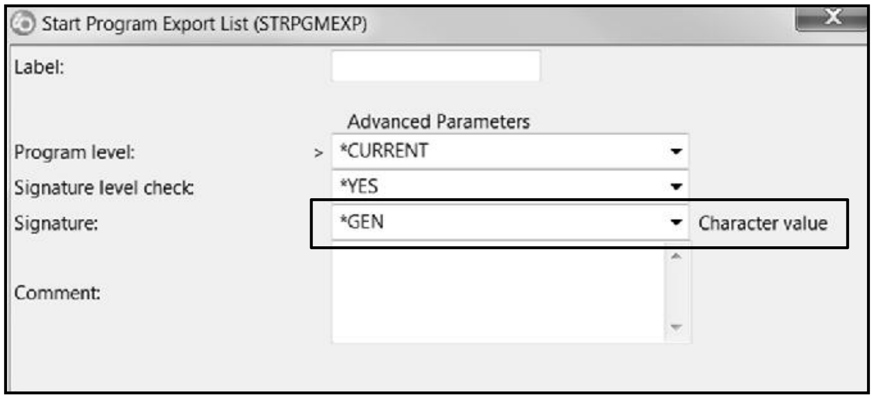
[illegible]

[illegible]

[illegible]

[illegible]

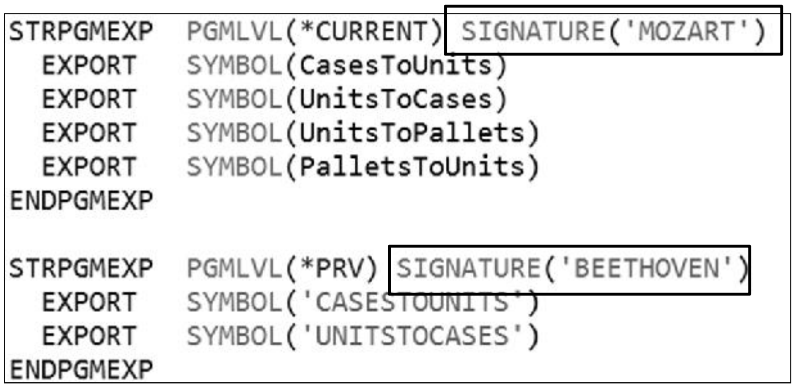
STRPGMEXP prompted



Start Program Export List (STRPGMEXP)

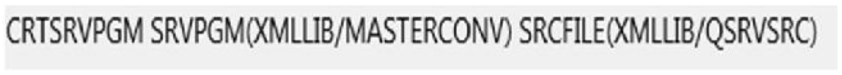
Label:	
	Advanced Parameters
Program level:	> *CURRENT
Signature level check:	*YES
Signature:	*GEN — Character value
Comment:	

Using named signatures



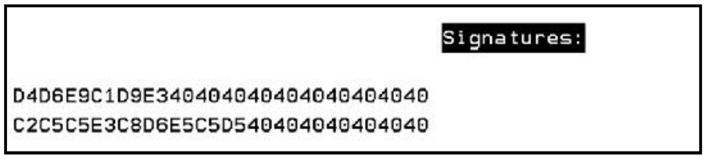
```
STRPGMEXP   PGMLVL(*CURRENT) SIGNATURE('MOZART')
  EXPORT    SYMBOL(CasesToUnits)
  EXPORT    SYMBOL(UnitsToCases)
  EXPORT    SYMBOL(UnitsToPallets)
  EXPORT    SYMBOL(PalletsToUnits)
ENDPGMEXP

STRPGMEXP   PGMLVL(*PRV) SIGNATURE('BEETHOVEN')
  EXPORT    SYMBOL('CASESTOUNITS')
  EXPORT    SYMBOL('UNITSTOCASES')
ENDPGMEXP
```

Using named signatures

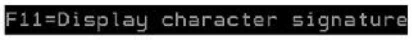


```
CRTSRVPGM SRVPGM(XMLLIB/MASTERCONV) SRCFILE(XMLLIB/QSRVSRC)
```

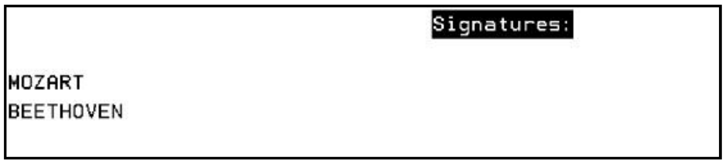
DSPSRVPGM MASTERCONV DETAIL(*SIGNATURE)



```
                                        Signatures:

D4D6E9C1D9E340404040404040404040
C2C5C5E3C8D6E5C5D540404040404040
```

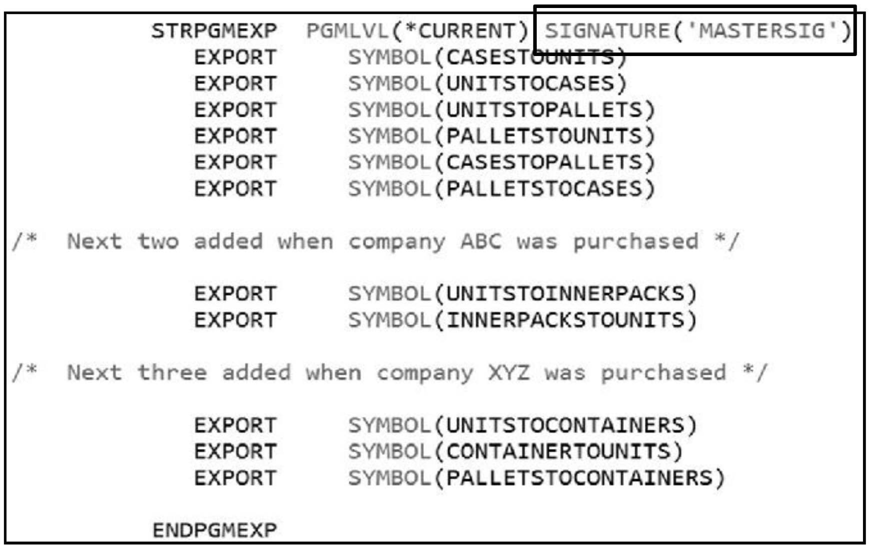


```
F11=Display character signature
```



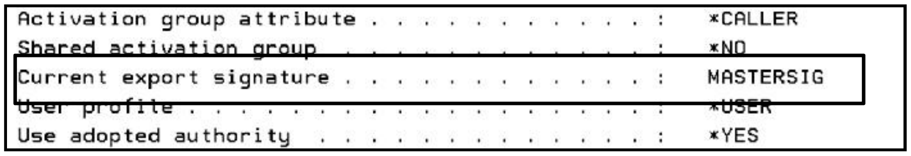
```
                                        Signatures:

MOZART
BEETHOVEN
```

Only one signature?

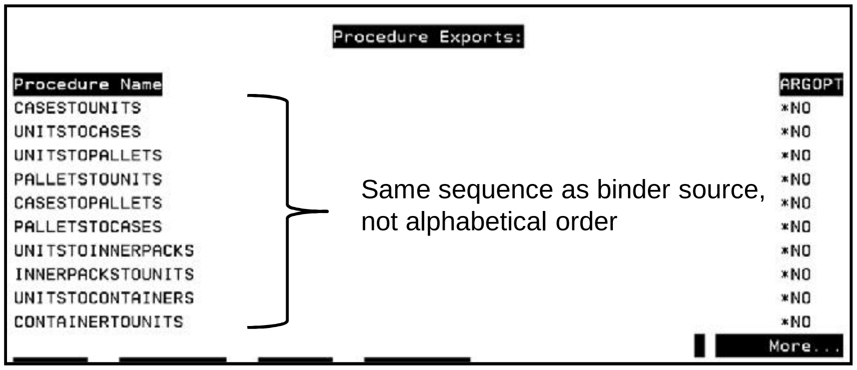


```
          STRPGMEXP  PGMLVL(*CURRENT) SIGNATURE('MASTERSIG')
             EXPORT     SYMBOL(CASESTOUNITS)
             EXPORT     SYMBOL(UNITSTOCASES)
             EXPORT     SYMBOL(UNITSTOPALLETS)
             EXPORT     SYMBOL(PALLETSTOUNITS)
             EXPORT     SYMBOL(CASESTOPALLETS)
             EXPORT     SYMBOL(PALLETSTOCASES)

/*  Next two added when company ABC was purchased */

             EXPORT     SYMBOL(UNITSTOINNERPACKS)
             EXPORT     SYMBOL(INNERPACKSTOUNITS)

/*  Next three added when company XYZ was purchased */

             EXPORT     SYMBOL(UNITSTOCONTAINERS)
             EXPORT     SYMBOL(CONTAINERTOUNITS)
             EXPORT     SYMBOL(PALLETSTOCONTAINERS)

          ENDPGMEXP
```

Only one signature?



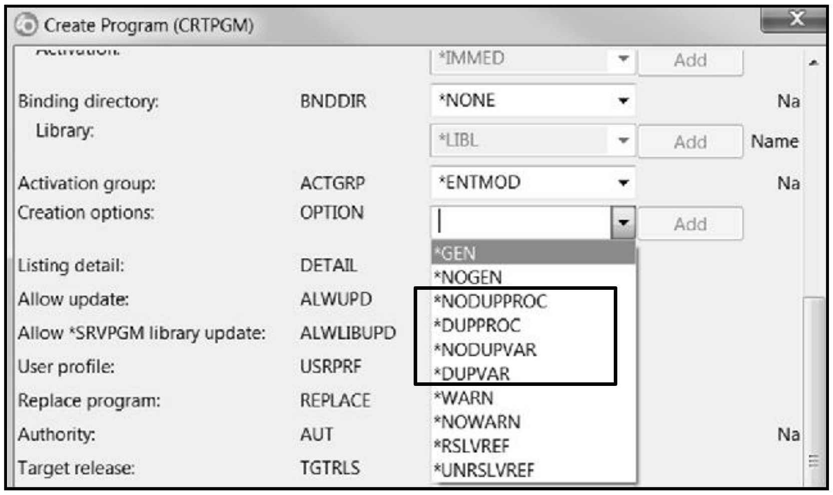
```
Activation group attribute . . . . . . . . . . . . :   *CALLER
Shared activation group  . . . . . . . . . . . . . :   *NO
Current export signature . . . . . . . . . . . . . :   MASTERSIG
User profile . . . . . . . . . . . . . . . . . . . :   *USER
Use adopted authority  . . . . . . . . . . . . . . :   *YES
```



```
                          Procedure Exports:

Procedure Name                                                ARGOPT
CASESTOUNITS                                                  *NO
UNITSTOCASES                                                  *NO
UNITSTOPALLETS                                                *NO
PALLETSTOUNITS                                                *NO
CASESTOPALLETS                                                *NO
PALLETSTOCASES                                                *NO
UNITSTOINNERPACKS                                             *NO
INNERPACKSTOUNITS                                             *NO
UNITSTOCONTAINERS                                             *NO
CONTAINERTOUNITS                                              *NO
                                                         More...
```

Same sequence as binder source, not alphabetical order

When using multiple modules



Create Program (CRTPGM)

Binding directory:	BNDDIR	*NONE	
Library:		*LIBL	
Activation group:	ACTGRP	*ENTMOD	
Creation options:	OPTION		
Listing detail:	DETAIL		
Allow update:	ALWUPD		
Allow *SRVPGM library update:	ALWLIBUPD		
User profile:	USRPRF		
Replace program:	REPLACE		
Authority:	AUT		
Target release:	TGTRLS		

Wrap up

- Modularization is not a new concept – embrace it!
- EXPORT(*ALL) is OK to use when you are willing to use UPDPGM or re-compile after a service program change.
- There is no hard and fast rule about the number of binding directories or modules entries within them.
- A service program signature can be thought of as a record format level identifier.
- There is a LVLCHK(*NO) parameter on the STRPGMEXP command which acts EXACTLY how you think it would.
- A service program can have more than one signature – BUT – the current one is used as the index.

The Business, Science and Uses of ILE Service Programs

Charles Guarino
@charlieguarino
Central Park Data Systems, Inc.