A Division of the UNICOM Group of Companies

UNICOM SYSTEMS, INC.

**Supplement #60**

84 Elm Street • Peterborough, NH 03458 USA
TEL (010)1-603-924-8818 • FAX (010)1-603-924-6348
Website: http://www.softlanding.com Email: techsupport@softlanding.com

# MANAGING ILE PROGRAMS

## TABLE OF CONTENTS

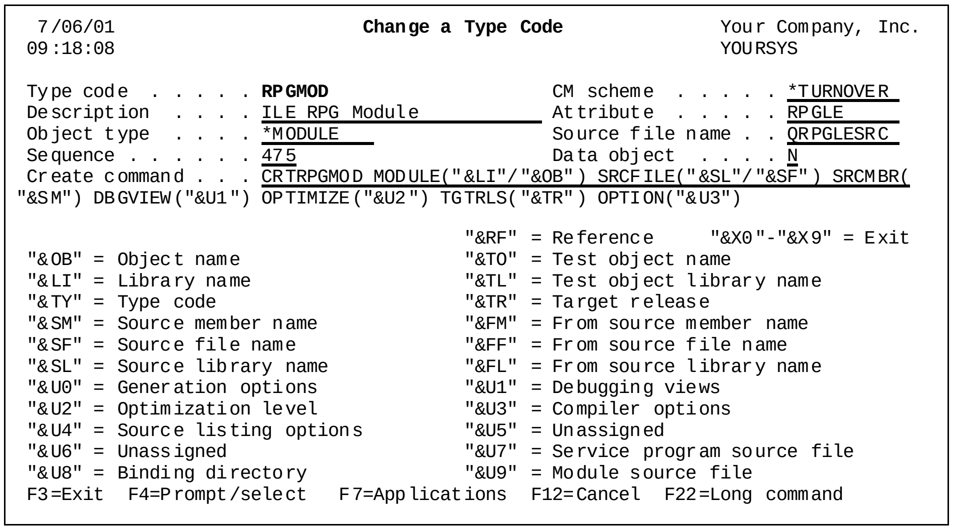
This document describes how you create and promote Integrated Language Environment (ILE) object types within TURNOVER® for iSeries v100. Included are such topics as the commands for creating ILE programs, the steps for managing ILE programs using TURNOVER® for iSeries v100, and the methods for converting RPG source to ILE source. While this document uses ILE RPG for most of its examples, the same principles apply to all of the ILE languages.

### UNICOM Systems, Inc. Note

Be aware that the recommended record length of the ILE RPG source file is 112 bytes, up from the previous standard default of 92 bytes.

## CREATING MODULES

To create and maintain ILE ***MODULE** object types, you need to use the appropriate type code, shipped with TURNOVER® for iSeries v100, for handling these object types. Generally, you want to use this method when a module object needs to be “bound” into several programs or service programs. We recommend creating a separate library in your application definition for storing your ***MODULE** objects, but you can also use your existing object libraries. A typical TURNOVER® for iSeries v100 type code entry[1] for a ***MODULE** object follows:



```
 7/06/01                        Change a Type Code              Your Company, Inc.
09:18:08                                                        YOURSYS

Type code  . . . . . RPGMOD                       CM scheme  . . . . . *TURNOVER
Description  . . . . ILE RPG Module               Attribute  . . . . . RPGLE
Object type  . . . . *MODULE                      Source file name . . QRPGLESRC
Sequence . . . . . . 475                          Data object  . . . . N
Create command . . . CRTRPGMOD MODULE("&LI"/"&OB") SRCFILE("&SL"/"&SF") SRCMBR(
"&SM") DBGVIEW("&U1") OPTIMIZE("&U2") TGTRLS("&TR") OPTION("&U3")

                                     "&RF" = Reference      "&X0"-"&X9" = Exit
"&OB" = Object name                  "&TO" = Test object name
"&LI" = Library name                 "&TL" = Test object library name
"&TY" = Type code                    "&TR" = Target release
"&SM" = Source member name           "&FM" = From source member name
"&SF" = Source file name             "&FF" = From source file name
"&SL" = Source library name          "&FL" = From source library name
"&U0" = Generation options           "&U1" = Debugging views
"&U2" = Optimization level           "&U3" = Compiler options
"&U4" = Source listing options       "&U5" = Unassigned
"&U6" = Unassigned                   "&U7" = Service program source file
"&U8" = Binding directory            "&U9" = Module source file
F3=Exit  F4=Prompt/select  F7=Applications  F12=Cancel  F22=Long command
```

As you can see, this TURNOVER® for iSeries v100 type code uses the standard IBM-supplied creation command for creating ***MODULE** objects. For CBL, CL, or C, make the appropriate adjustments to the ***Create command***, ***Attribute***, and ***Source file name*** fields.

You can check out and promote ***MODULE** object types as you would any other standard iSeries type (such as “non-ILE” RPG and CL programs).

TURNOVER® for iSeries v100 also provides support in its cross-reference database for ***MODULE** objects. If you’re changing a module object, you can view all of the programs and service programs that refer to that module, so that they can be “re-bound” after the module is recreated.

[1] Select Main Menu option **8**, then option **4**.

## CREATING PROGRAMS

The two different ways to create an ILE program object are:

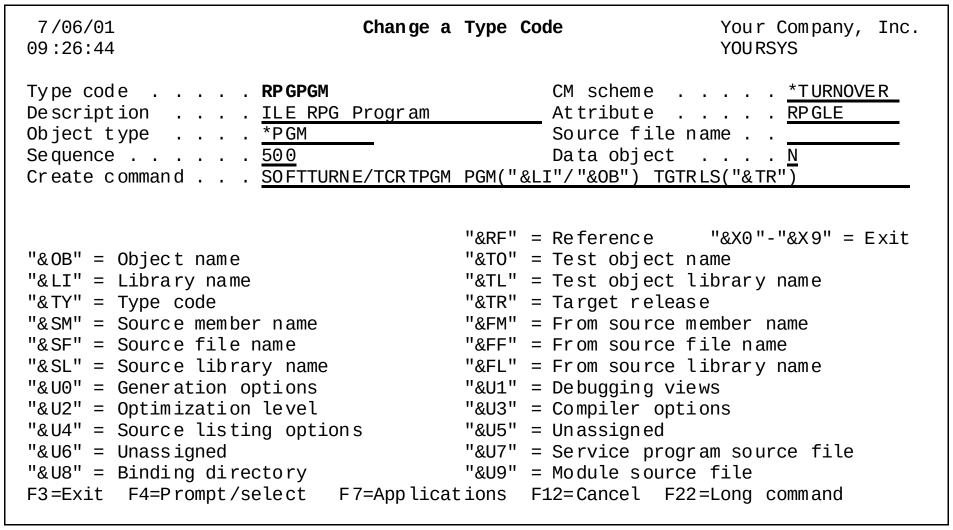
- Using the **CRTPGM** command and specifying the modules that make up the program. You *must* use this method if your program consists of more than one module.
- Using the **CRTBND*xxx*** command (where *xxx* is RPG, CBL, CL, or C). This command creates a program object directly from the program's source, bypassing the module creation process. You can use this command only when your program consists of a single module. This method is provided primarily to maintain familiarity with the "old" way of creating programs in one step, as you did before ILE was available.

Both of these methods have their problems if you are not using TURNOVER® for iSeries v100, but TURNOVER® for iSeries v100 provides some utilities for enhancing each. The method you choose is entirely up to you. TURNOVER® for iSeries v100 supports both methods, even within the same application.

## Using CRTPGM to create programs

An annoyance with the **CRTPGM** command is that you have to specify all of the modules and service programs that make up your program each time you want to recreate it, even if you've only updated a single module. To address this issue, the TURNOVER® for iSeries v100 command, **TCRTPGM**, serves as a "wrapper" for the **CRTPGM** command. The **TCRTPGM** command uses operating system APIs to automatically specify all of the modules and service programs that comprise your program. The only time you have to specify which modules to include, when you create an ILE program, is when you're adding or removing modules from the list.

The TURNOVER® for iSeries v100 type code entry that you use for creating ILE RPG programs with this method looks as follows:



```
 7/06/01                      Change a Type Code             Your Company, Inc.
09:26:44                                                     YOURSYS

Type code  . . . . . RPGPGM                      CM scheme  . . . . . *TURNOVER
Description  . . . . ILE RPG Program             Attribute  . . . . . RPGLE
Object type  . . . . *PGM                        Source file name . .
Sequence . . . . . . 500                         Data object  . . . . N
Create command . . . SOFTTURNE/TCRTPGM PGM("&LI"/"&OB") TGTRLS("&TR")


                                  "&RF" = Reference     "&X0"-"&X9" = Exit
"&OB" = Object name               "&TO" = Test object name
"&LI" = Library name              "&TL" = Test object library name
"&TY" = Type code                 "&TR" = Target release
"&SM" = Source member name        "&FM" = From source member name
"&SF" = Source file name          "&FF" = From source file name
"&SL" = Source library name       "&FL" = From source library name
"&U0" = Generation options        "&U1" = Debugging views
"&U2" = Optimization level        "&U3" = Compiler options
"&U4" = Source listing options    "&U5" = Unassigned
"&U6" = Unassigned                "&U7" = Service program source file
"&U8" = Binding directory         "&U9" = Module source file
F3=Exit  F4=Prompt/select   F7=Applications  F12=Cancel  F22=Long command
```

Because ILE programs really are language-independent, you could just have a single type code that you use to create all ILE program objects using this method. For clarity, though, TURNOVER® for iSeries v100 provides a type code for each language.

The **TCRTPGM** command has all of the same parameters and values as the **CRTPGM** command. (Ultimately, that is the command that it executes anyway.) The **TCRTPGM** command determines the proper module and service program list, then builds and executes the appropriate **CRTPGM** command string. If you want to override some of the other command parameters, you can, and your overrides are merged into the command string. If you don't specify a value for a particular parameter, the **CRTPGM** command default value for that parameter is used.

When determining the module and service program list to use, TURNOVER® for iSeries v100 applies the following rules:

- If you are using the **TCRTPGM** command to create an object you are testing (as when you are working with PWM or as a PDM user option), the command uses the library list to find the version of the program it uses for retrieving the reference information.

- If you are using the **TCRTPGM** command on a form to promote an object, the command uses, as the reference object, the program you are promoting in the ***from*** library. If the program doesn't exist in that library, the command looks for the program in the target library. If it doesn't exist there, the command looks for the program in the library list, and uses that.

- Usually, as long as you want to recreate programs as they exist, with the same modules bound together, you can execute the command with its defaults. If you want to add or remove a module from a program, you need to recreate it in your development library, from a command line using the iSeries **CRTPGM** command, and specify the modules you want to bind together. To determine what modules to use, the TURNOVER® for iSeries v100 form promotion job looks in your ***from*** library for the reference program. You don't need to specify this information again when you promote the program.

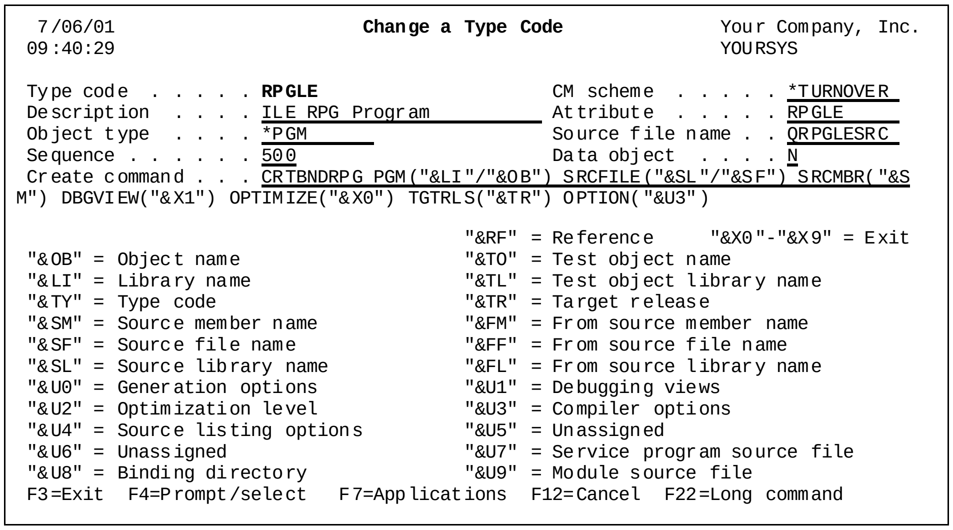
You can check out and promote programs as you would any other object-only type. However, because there is no source, but there is a create command to execute, the default method is always **COPR** (of course, you can also use **MO** or **CD**).

## Using CRTBNDxxx to create programs

Bound programs created using the **CRTBND*xxx*** command do not contain any information, in the object's description, about the source from which they were created. Because TURNOVER® relies on this information during checkout and auditing, this presented a problem in the past. In Release 4.1, we created the **TCBP** command to address this issue. In Release 4.2, we enhanced the way TURNOVER® handles ILE bound programs, as is described below.

### *Defining the underlying create command (as of Release 4.2)*

**Note:** As of Release 4.2, you no longer need to use the **TCBP** command in your create commands. You can specify the iSeries create command (**CRTBNDRPG**, **CRTBNDCBL**, and so on) as your default create command. (If you're using a version of TURNOVER® Release 4.2 with a date prior to November 25, 1997, you can revise your global type codes to remove the **TCBP** command and its parameters.) An example of the RPGLE global type code is shown below:



```
 7/06/01                   Change a Type Code             Your Company, Inc.
09:40:29                                                  YOURSYS

Type code  . . . . . RPGLE                        CM scheme  . . . . . *TURNOVER
Description  . . . . ILE RPG Program              Attribute  . . . . . RPGLE
Object type  . . . . *PGM                         Source file name . . QRPGLESRC
Sequence . . . . . . 500                          Data object  . . . . N
Create command . . . CRTBNDRPG PGM("&LI"/"&OB") SRCFILE("&SL"/"&SF") SRCMBR("&S
M") DBGVIEW("&X1") OPTIMIZE("&X0") TGTRLS("&TR") OPTION("&U3")

                                        "&RF" = Reference     "&X0"-"&X9" = Exit
"&OB" = Object name                     "&TO" = Test object name
"&LI" = Library name                    "&TL" = Test object library name
"&TY" = Type code                       "&TR" = Target release
"&SM" = Source member name              "&FM" = From source member name
"&SF" = Source file name                "&FF" = From source file name
"&SL" = Source library name             "&FL" = From source library name
"&U0" = Generation options              "&U1" = Debugging views
"&U2" = Optimization level              "&U3" = Compiler options
"&U4" = Source listing options          "&U5" = Unassigned
"&U6" = Unassigned                      "&U7" = Service program source file
"&U8" = Binding directory               "&U9" = Module source file
F3=Exit  F4=Prompt/select   F7=Applications  F12=Cancel  F22=Long command
```

Because we have changed our default shipping values to reflect these changes, TURNOVER® installations dated November 25 or later, automatically have this change. If you choose not to continue using the **TCBP** command, your object descriptions will no longer include information about the source they were created from. However, you can still view this information using TURNOVER®'s **TDSPOBJD** command, or using the **DSPPGM** command.

### *Continuing support of the TCBP command*

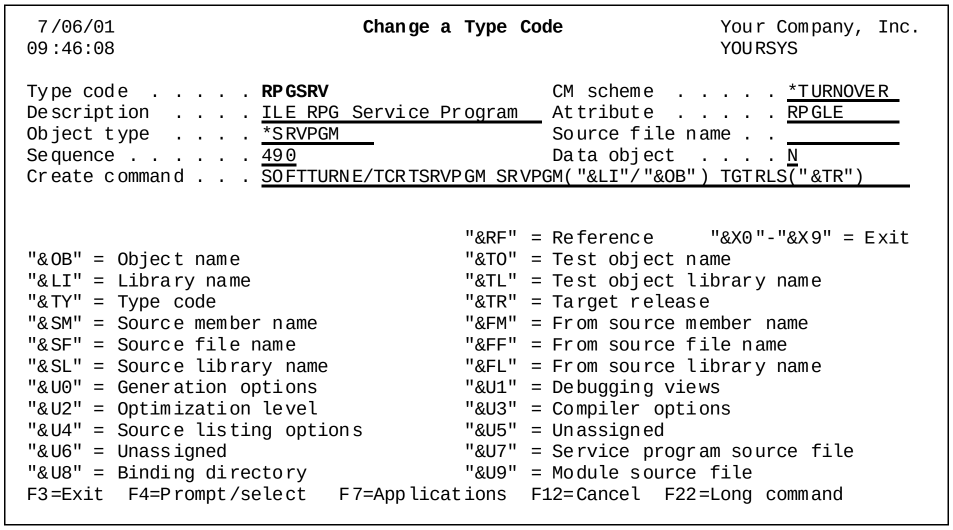
If the **TCBP** command has been working well for you, or if you really need to have the source information in the object description stored as it was before, you can keep using the **TCBP** command. We are not removing the command, or the support for it, from our product.

Another alternative available to you is to use the Post-Form Exit (Exit 1) to have a program update source information in the object description.

## CREATING SERVICE PROGRAMS

Service programs are handled in the same way as the CRTPGM method for creating programs. We have a special command called **TCRTSRVPGM** that operates exactly the same way as **TCRTPGM**.

A typical service program TURNOVER® for iSeries v100 type code entry should look like this:



```
 7/06/01                        Change a Type Code             Your Company, Inc.
09:46:08                                                       YOURSYS

Type code  . . . . . RPGSRV                       CM scheme  . . . . . *TURNOVER
Description  . . . . ILE RPG Service Program      Attribute  . . . . . RPGLE
Object type  . . . . *SRVPGM                      Source file name . .
Sequence . . . . . . 490                          Data object  . . . . N
Create command . . . SOFTTURNE/TCRTSRVPGM SRVPGM("&LI"/"&OB") TGTRLS("&TR")


                                        "&RF" = Reference     "&X0"-"&X9" = Exit
"&OB" = Object name                     "&TO" = Test object name
"&LI" = Library name                    "&TL" = Test object library name
"&TY" = Type code                       "&TR" = Target release
"&SM" = Source member name              "&FM" = From source member name
"&SF" = Source file name                "&FF" = From source file name
"&SL" = Source library name             "&FL" = From source library name
"&U0" = Generation options              "&U1" = Debugging views
"&U2" = Optimization level              "&U3" = Compiler options
"&U4" = Source listing options          "&U5" = Unassigned
"&U6" = Unassigned                      "&U7" = Service program source file
"&U8" = Binding directory               "&U9" = Module source file
F3=Exit  F4=Prompt/select   F7=Applications  F12=Cancel  F22=Long command
```

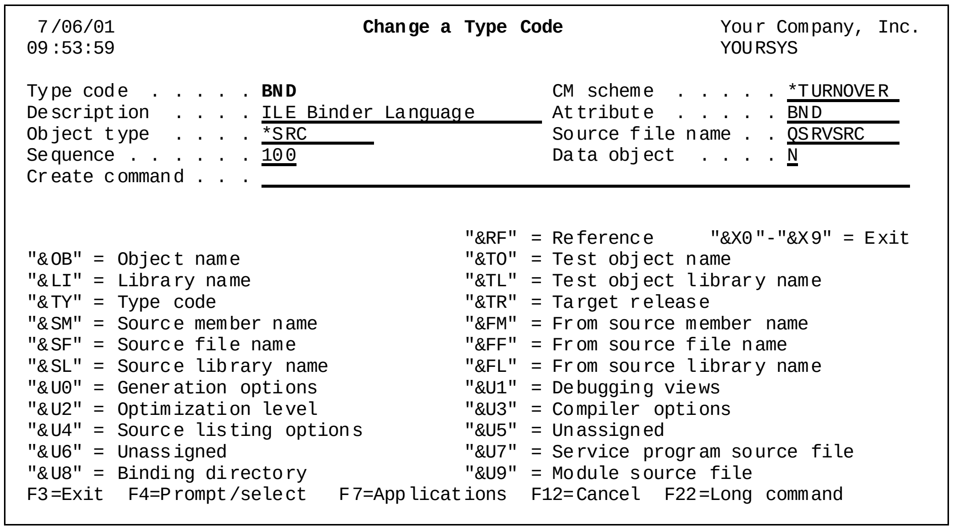
Like ILE programs, ILE service programs are language independent. You could use one type code for all languages. The important thing to consider when setting up a service program type code is the sequence number. Because service programs are created from modules, they should have a higher sequence number than modules. However, because they're used to create programs, they must have a lower sequence number than programs. This ensures that the promotion order on a TURNOVER® for iSeries v100 form will be: modules, service programs, and then programs.

You can check out and promote service programs as you would any other object-only type. However, because there is no source, but there is a create command to execute, the default method is always **COPR** (of course, you can also use **MO** or **CD**). The same is true for programs using the **CRTPGM** method.

# OTHER ILE OBJECT TYPES

## Binding language source (BND)

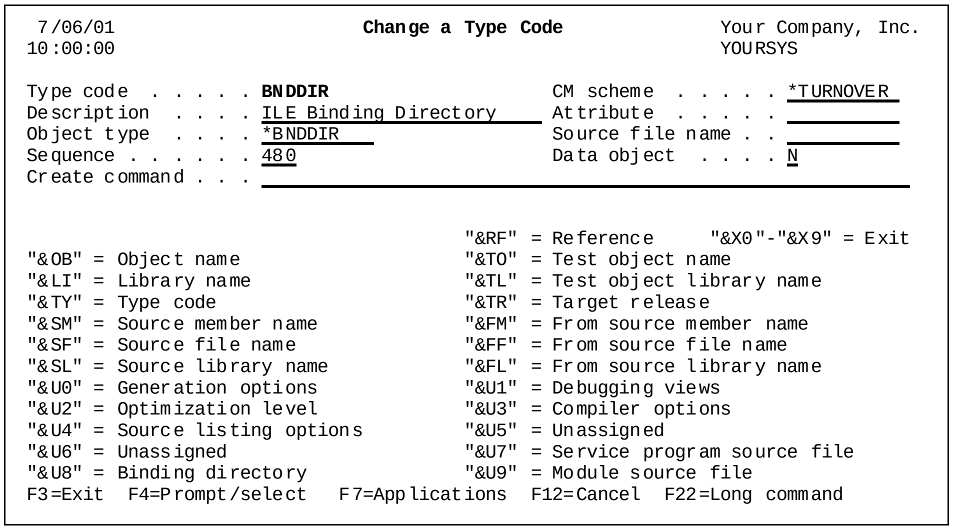
This is a source-only type that contains statements that you can use when creating service programs.  If you want to promote the binding language source separately as a source-only type in TURNOVER® for iSeries v100, you must have a TURNOVER® for iSeries v100 type code entry defined like this:



```
 7/06/01                        Change a Type Code             Your Company, Inc.
09:53:59                                                       YOURSYS

Type code  . . . . . BND                         CM scheme  . . . . . *TURNOVER
Description  . . . . ILE Binder Language         Attribute  . . . . . BND
Object type  . . . . *SRC                        Source file name . . QSRVSRC
Sequence . . . . . . 100                         Data object  . . . . N
Create command . . .


                                  "&RF" = Reference     "&X0"-"&X9" = Exit
"&OB" = Object name               "&TO" = Test object name
"&LI" = Library name              "&TL" = Test object library name
"&TY" = Type code                 "&TR" = Target release
"&SM" = Source member name        "&FM" = From source member name
"&SF" = Source file name          "&FF" = From source file name
"&SL" = Source library name       "&FL" = From source library name
"&U0" = Generation options        "&U1" = Debugging views
"&U2" = Optimization level        "&U3" = Compiler options
"&U4" = Source listing options    "&U5" = Unassigned
"&U6" = Unassigned                "&U7" = Service program source file
"&U8" = Binding directory         "&U9" = Module source file
F3=Exit  F4=Prompt/select   F7=Applications  F12=Cancel  F22=Long command
```

## Bind directory (BNDDIR)

This is an object-only type; that is, it is moved or duplicated from test to target libraries.  You can work with the directory and add directory entries using the **CRTBNDDIR** and **WRKBNDDIR** commands.  Once a directory is created, the **CRTPGM** command can reference it to indicate what modules or service programs should be included in the ILE program.  To set up a BNDDIR type, you must have a TURNOVER® for iSeries v100 type entry like this:

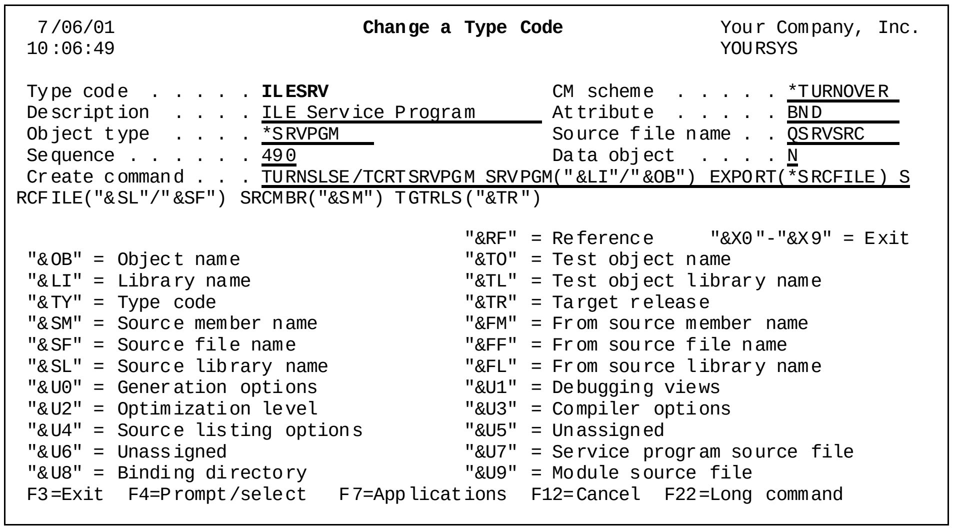


```
 7/06/01                        Change a Type Code             Your Company, Inc.
10:00:00                                                       YOURSYS

Type code  . . . . . BNDDIR                      CM scheme  . . . . . *TURNOVER
Description  . . . . ILE Binding Directory       Attribute  . . . . .
Object type  . . . . *BNDDIR                     Source file name . .
Sequence . . . . . . 480                         Data object  . . . . N
Create command . . .


                                  "&RF" = Reference     "&X0"-"&X9" = Exit
"&OB" = Object name               "&TO" = Test object name
"&LI" = Library name              "&TL" = Test object library name
"&TY" = Type code                 "&TR" = Target release
"&SM" = Source member name        "&FM" = From source member name
"&SF" = Source file name          "&FF" = From source file name
"&SL" = Source library name       "&FL" = From source library name
"&U0" = Generation options        "&U1" = Debugging views
"&U2" = Optimization level        "&U3" = Compiler options
"&U4" = Source listing options    "&U5" = Unassigned
"&U6" = Unassigned                "&U7" = Service program source file
"&U8" = Binding directory         "&U9" = Module source file
F3=Exit  F4=Prompt/select   F7=Applications  F12=Cancel  F22=Long command
```

## Promoting a service program with its binder language

You can define a TURNOVER® for iSeries v100 type code that specifies a service program's binder language source file. This allows you to change the binder language and promote it, along with the service program. To change binder language for a service program, follow these steps:

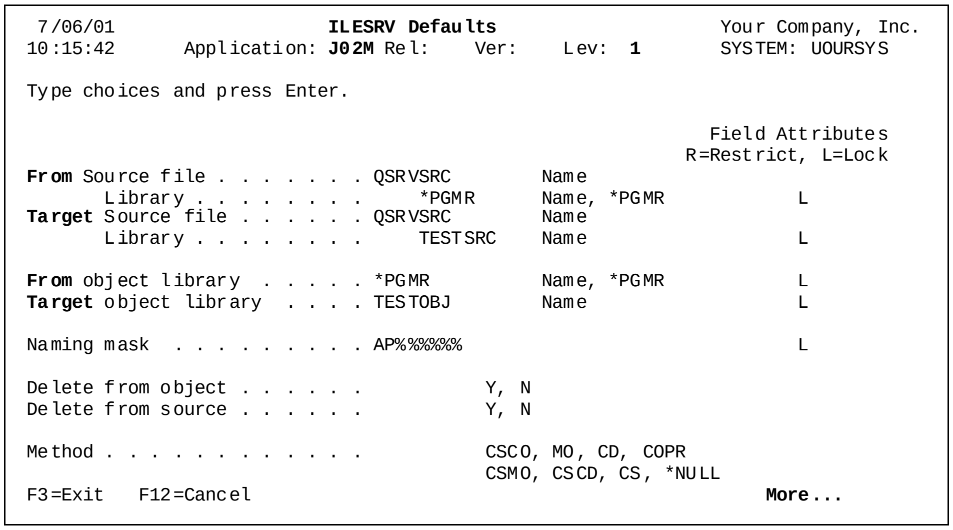
1. Define a new TURNOVER® for iSeries v100 global type code called ILESRV, as shown here:



```
 7/06/01                      Change a Type Code            Your Company, Inc.
10:06:49                                                    YOURSYS

Type code  . . . . . ILESRV                   CM scheme  . . . . . *TURNOVER
Description  . . . . ILE Service Program      Attribute  . . . . . BND
Object type  . . . . *SRVPGM                  Source file name . . QSRVSRC
Sequence . . . . . . 490                      Data object  . . . . N
Create command . . . TURNSLSE/TCRTSRVPGM SRVPGM("&LI"/"&OB") EXPORT(*SRCFILE) S
RCFILE("&SL"/"&SF") SRCMBR("&SM") TGTRLS("&TR")

                                        "&RF" = Reference     "&X0"-"&X9" = Exit
"&OB" = Object name                     "&TO" = Test object name
"&LI" = Library name                    "&TL" = Test object library name
"&TY" = Type code                       "&TR" = Target release
"&SM" = Source member name              "&FM" = From source member name
"&SF" = Source file name                "&FF" = From source file name
"&SL" = Source library name             "&FL" = From source library name
"&U0" = Generation options              "&U1" = Debugging views
"&U2" = Optimization level              "&U3" = Compiler options
"&U4" = Source listing options          "&U5" = Unassigned
"&U6" = Unassigned                      "&U7" = Service program source file
"&U8" = Binding directory               "&U9" = Module source file
F3=Exit  F4=Prompt/select   F7=Applications  F12=Cancel  F22=Long command
```

**Figure 1: Sample ILESRV global type code definition**

2. Include the type code in your application:



```
 7/06/01                   ILESRV Defaults                  Your Company, Inc.
10:15:42      Application: J02M Rel:    Ver:    Lev:  1      SYSTEM: UOURSYS

Type choices and press Enter.

                                                              Field Attributes
                                                            R=Restrict, L=Lock
From Source file . . . . . . . QSRVSRC        Name
       Library . . . . . . . .     *PGMR      Name, *PGMR           L
Target Source file . . . . . . QSRVSRC        Name
       Library . . . . . . . .     TESTSRC    Name                  L

From object library  . . . . . *PGMR          Name, *PGMR           L
Target object library  . . . . TESTOBJ        Name                  L

Naming mask  . . . . . . . . . AP%%%%%                              L

Delete from object . . . . . .          Y, N
Delete from source . . . . . .          Y, N

Method . . . . . . . . . . . .          CSCO, MO, CD, COPR
                                        CSMO, CSCD, CS, *NULL
F3=Exit   F12=Cancel                                          More...
```

**Figure 2: Sample ILESRV application type code definition**

Note that the name of the binder language source member in the application type code must match the name of the service program that you want to change.

3. You can now change the binder language by adding the service program to the Programmer Worklist using the ILESRV type code. Check out the binder language by selecting the service program with option **21**. (This copies the binder language to your development library.) Select the service program in development using option **32** to open the binder language for editing.

4. Make any desired changes to the binder language.

5. Now use **CRTSRVPGM** in development on the command line, to recreate the service program, using the updated binder language.

6. To promote the new service program, together with its binder language, add it to a TURNOVER® for iSeries v100 form by selecting it with option **46**.

When you run the form, both the binder language and the bound service program are promoted to the target library, using method CSCO, CSCD, or CSMO as defined by the ILESRV application type code (refer to ***Figure 1: Sample ILESRV global type code definition*** on page 9).

## WORK FLOW FOR MANAGING ILE OBJECTS USING TURNOVER® FOR ISERIES V100

We have prepared a list of steps for managing ILE objects with TURNOVER® for iSeries v100. These steps follow.

### Setting up TURNOVER® for iSeries v100 to manage ILE objects

Before you can begin handling ILE objects with TURNOVER® for iSeries v100, there is some basic setup work you have to perform:

1. Make sure that the TURNOVER® for iSeries v100 application you plan to use contains the appropriate ILE type codes you will need. This is a list of the ILE type codes we ship with TURNOVER® for iSeries v100:

| For COBOL: | For CL: | For SQL C: | For SQL COBOL: |
|---|---|---|---|
| CBLLE<br>CBLMOD<br>CBLSRV<br>CBLPGM | CLLE<br>CLPMOD<br>CLPSRV<br>CLPPGM | SQLC<br>SQLCM<br>SQLCS | SQLCBI<br>SQLCBM<br>SQLCBS |

| For C: | For RPG: | For SQL RPG: | For Binding: |
|---|---|---|---|
| CLE<br>CLEMOD<br>CLESRV<br>CLEPGM | RPGLE<br>RPGMOD<br>RPGSRV<br>RPGPGM | SQLRPI<br>SQLRPM<br>SQLRPS | BND<br>BNDDIR |

2. Make sure the source files you use have a record length of 112.

   Many companies use a source file (for example, QRPGLESRC for RPG ILE source) that they create with the suggested record length of 112. Specify the name of this source file in the appropriate type codes for ILE.

3. Create a Programmer Worklist to use for checking out, editing, and promoting the ILE objects you will manage. Specify the programmer as the resource, the application as the one you've prepared for ILE, and supply a reference number (usually the number of a TURNOVER® for iSeries v100 project task). If you're using the TURNOVER® for iSeries v100 project system, you can select a task with option **20** to create a worklist for it.

**UNICOM Systems, Inc. Note**

You can set the ***List sequence*** value for your Programmer Worklist defaults (**F18** on your worklist) to *CRTSEQ. This shows items on your worklist in the order they'd be created - modules first, followed by the service programs and programs into which you'll bind them. The sequence parameter in the global type codes specifies the correct create order for each object type (module, 475; service program, 490; program, 500).

Now you're ready to use your worklist to manage ILE objects. Continue with the next topic for step-by-step instructions for RPG and CL ILE objects.

## Handling individual bound modules

Use this approach for objects that you plan to create such that one source member is created, compiled into a module, and immediately bound into its own program, all in one operation. In this case, only one type code is needed for each executable bound program.

1. Add the module name(s) to the worklist as object type RPGLE or CLLE.
2. Check out the source for the module(s) (option **21**).
3. Edit the source in the development library (option **32**).
4. Compile the source in development library (option **36**).
5. Add the module name(s) to the TURNOVER® for iSeries v100 form for the first application level (option **46**).
6. Run the form to promote the bound modules and their source members to the first application level (option **47**).
7. Copy the form to the next level (option **43**).
8. Run the form to promote to the next level (option **47**).
9. Copy the form and run the form for the remaining unlocked levels.

## Binding several modules together

Do the following steps for modules that you will link together into programs or service programs.

1. Add the module name(s) to the worklist as object type RPGMOD or CLPMOD.

2. Add to the worklist the program that will contain the linked modules, using object type RPGPGM or CLPPGM. Or, if modules are to be linked into a service program, add the name of the service program to the list as object type RPGSRV or CLPSRV.

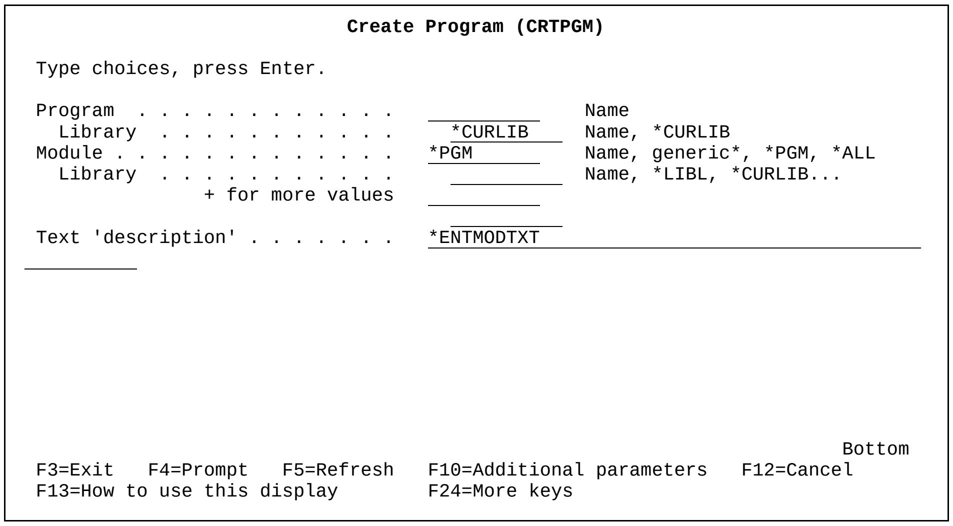
3. Check out each module and program or service program (option **21**).

4. Edit each module’s source in development (option **32**).

5. Compile each module in development (option **36**).

6. Create new service programs or programs with the appropriate iSeries command, as follows:

   Type **CRTPGM** or **CRTSRVPGM** on a TURNOVER® for iSeries v100 command line and press **F4** to prompt. (Our example shows **CRTPGM**.)



```
                         Create Program (CRTPGM)

 Type choices, press Enter.

 Program  . . . . . . . . . . .   __________    Name
   Library  . . . . . . . . . .     *CURLIB     Name, *CURLIB
 Module . . . . . . . . . . . .   *PGM          Name, generic*, *PGM, *ALL
   Library  . . . . . . . . . .     __________  Name, *LIBL, *CURLIB...
               + for more values  __________

 Text 'description' . . . . . .   *ENTMODTXT



                                                                      Bottom
 F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
 F13=How to use this display        F24=More keys
```

   Use the ***Module*** parameter to provide names of all the modules you’re linking. To enter additional modules and libraries, use **+ *for more values***.

### Important!

You must identify all the modules that you’re linking, if:

- This is the first time you are linking them; **or**
- You are adding or removing modules from a program or a previously bound service program.

### UNICOM Systems, Inc. Note

Note that if a program is to be recreated with the same module list, then you can rebind as usual from the worklist using option **36**. TURNOVER® for iSeries v100 looks at the object found in the library list to determine the correct module list.

Press **Enter** to link the modules together into a service program or program in the development object library.

7. Use option **46** to add to a TURNOVER® for iSeries v100 form the names of all modules (RPGMOD, CLPMOD…) and all programs or service programs (RPGPGM, RPGSRV, CLPPGM, CLPSRV) into which the modules will be linked.

   Notice that TURNOVER® for iSeries v100 can order the form correctly with modules coming first, followed by service programs, and then programs. Also, TURNOVER® for iSeries v100 links the modules as you specified when you created their program or service program (step 6. During cross-reference checking for the form, TURNOVER® for iSeries v100 can add, to the form, additional programs or service programs that need to be recreated because their modules have changed.

8. Run the TURNOVER® for iSeries v100 form to promote to the first level (option **47**).
9. Copy the TURNOVER® for iSeries v100 form to the next level (option **43**).
10. Run the TURNOVER® for iSeries v100 Form to promote to the next level (option **47**).
11. Copy the form and run the form for the remaining unlocked levels.

# CONVERTING EXISTING OBJECTS TO ILE OBJECTS

Two conversion methods are available in TURNOVER® for iSeries v100 for converting from existing objects to ILE objects. You can manually convert existing objects to ILE objects using TURNOVER® for iSeries v100's Programmer Worklist Manager (PWM). However, even with TURNOVER® for iSeries v100 and PWM, the manual process requires many steps. (Refer to ***Using the worklist for manual conversion*** on page 19 for instructions for manually converting the objects.) In TURNOVER® Release 4.2, we created the Convert Object Type (**TCVTOBJTYP**) utility to automate the conversion process. Both methods are described in this document.

## Using the TCVTOBJTYP utility for automated conversion

With Release 4.2 (tape date 3/26/98) or higher, you can use the Convert Object Type (**TCVTOBJTYP**) utility to convert an object from one type to another. This utility is particularly useful for converting RPG to RPGLE, and RPG to RPGMOD.

The **TCVTOBJTYP** utility allows you to perform the conversion in four steps, as follows:

### ***Setup (steps 1-3)***

1. Create a type code conversion table, with entries that specify the command to be executed to convert the object from one type to another.

2. Verify that the type codes exist at all levels of your application definition(s).

3. Create user-defined options to run the **TCVTOBJTYP** utility from your worklist or from within PDM.

### ***Conversion execution (step 4)***

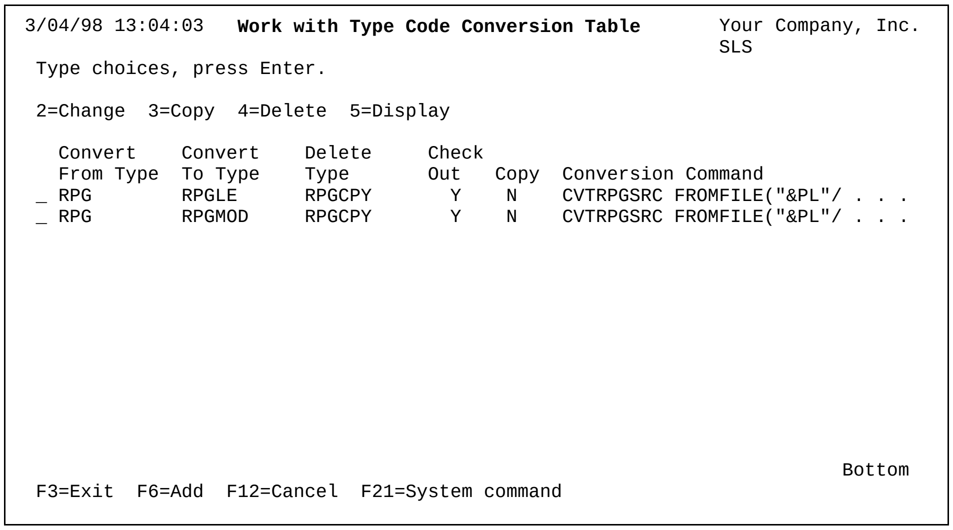
Perform the conversion, using one of the user-defined options you created to run the utility.

Descriptions of each of these steps follow.

## Step 1. Creating conversion table entries

To manage entries in the type code conversion table, you must have TURNOVER® for iSeries v100 system defaults authority. All the usual TURNOVER® for iSeries v100 rules apply to the **TCVTOBJTYP** utility; that is, all the type codes necessary for this utility to function properly must exist at the global level and at each level of the application (including permanently locked levels) that you will use to perform the conversion.

From the TURNOVER® for iSeries v100 Main Menu, select option **8** for the Utility Menu. From the Utility Menu, select option **13** (***Work with type code conversion table***). The ***Work with Type Code Conversion Table*** panel appears:



```
3/04/98 13:04:03   Work with Type Code Conversion Table       Your Company, Inc.
                                                              SLS
 Type choices, press Enter.

 2=Change  3=Copy  4=Delete  5=Display

   Convert    Convert    Delete     Check
   From Type  To Type    Type       Out   Copy  Conversion Command
 _ RPG        RPGLE      RPGCPY       Y    N    CVTRPGSRC FROMFILE("&PL"/ . . .
 _ RPG        RPGMOD     RPGCPY       Y    N    CVTRPGSRC FROMFILE("&PL"/ . . .











                                                                          Bottom
 F3=Exit  F6=Add  F12=Cancel  F21=System command
```

Within this table, use **F6** to create an entry (for example, the entry for converting RPG to RPGLE). For more information, see ***Chapter 8***: ***Utility Menu*** in the ***TURNOVER® for iSeries v100 User Guide***. Also, detailed online Help is available to assist you.

## Step 2. Verifying type codes

Ensure that the type codes you specify in your conversion table entry exist at each level of the application (including permanently locked levels) that you will be using to perform the conversion. This includes any type code you have defined as the ***Type code to delete*** in the conversion table entry.

## ***Step 3. Creating user-defined options***

To perform the conversion, run the **TCVOBJTYP** utility by selecting items with user-defined options, either from the Programmer Worklist, or from within PDM.

Create one or more of the user-defined options, depending on which ones you think you will use, specifying the **TCVTOBJTYP** utility as shown here:

| | |
|---|---|
| **For a Programmer Worklist user-defined option** | `TCVTOBJTYP OBJ("&OB") FROMTYPE("&OA") TOTYPE(*SELECT) REF("&RF") APPL("&A") REL(-11) VER(-22) LEV(-33) PGMR("&PG")` |
| **For Converting from PDM *Work with Objects*** | `TCVTOBJTYP OBJ(&N) FROMTYPE(&A) TOTYPE(*SELECT)` |
| **For Converting from PDM *Work with Members*** | `TCVTOBJTYP OBJ(&N) FROMTYPE(&T) TOTYPE(*SELECT)` |

For PDM ***Work with Objects***

For PDM ***Work with Members***

**Table 1: User-defined option specifications**

Whether you convert from your Worklist or from within PDM, you must create a user-defined option for the specific environment you choose before proceeding. The following instructions assume you have already created the option as shown above.

## ***Step 4. Performing the conversion***

To convert objects, you select them either from your Programmer Worklist, or from object or member lists within PDM, using the appropriate user-defined option you created in Step 3.

### ***Converting from the Programmer Worklist:***

1. Add the object to the Programmer Worklist using the “old” type code (for example, RPG), but **DON’T CHECK IT OUT**. Select the production-level object with the Worklist user-defined option you created (instead of using option **21**). Press **Enter**.

2. The ***TCVTOBJTYP Type Code Selection*** popup panel appears. Indicate the type code to which you want the object converted by selecting the desired conversion table entry. Press **Enter**.

At this point, the conversion process takes place interactively. When the conversion is finished, your worklist is refreshed to show the new object type. You then see that the object has been converted to RPGLE, or to whatever type code you selected, and it has been checked out (if you specified **Y** for ***Checkout*** in the conversion table entry).

### ***Converting from within PDM:***

1. Starting from the Programmer Worklist, use **F20** to go to PDM. **Note: DO NOT ADD THE ITEM TO THE WORKLIST!**

2. Select option **2**, ***Work with Objects*** OR option **3**, ***Work with Members*** and find the object or member you want to convert.

3. Select the object or member you want to convert with the appropriate PDM user-defined option you created. (Remember – the option is different depending on whether you're in ***Work with Objects*** or ***Work with Members***.) Press **Enter**.

4. The ***TCVTOBJTYP Type Code Selection*** popup panel appears. Indicate the type code to which you want the object or member converted by selecting the desired conversion table entry. Press **Enter**.

At this point, the conversion process takes place interactively. When the conversion is finished, return to the Programmer Worklist. The object is added as the type code you selected, and it is checked out (if you specified **Y** for ***Checkout*** in the conversion table entry).

## ***What happens to the "old" source member?***

When you first created an entry in the type code conversion table, you may have entered a type code you wanted to delete. If you did, when a form is created for the highest unlocked level (Production) of the application, a **D** delete line is added to the form for the object being converted. The ***Type code to delete*** is used to remove the old source member (for example, the RPG member that was converted). In most cases, you will have defined RPGCPY as the type code you want deleted**,** but your company may use some other RPXXXX type code to designate a **source-only** type for RPG members.

If you did not specify a ***Type code to delete***, then a **D** delete line is NOT added to the highest unlocked level (Production) form.

## UNICOM Systems, Inc. Recommends

We recommend that you enter a value for the Type code to delete parameter in the conversion table entry. As long as the highest unlocked level of the application being used is set to archive source on line, TURNOVER® for iSeries v100 archives the "old" source member.

## Using the worklist for manual conversion

You can also use the worklist to manage the activities involved in converting existing objects to ILE objects. This is a little different from the steps outlined for managing ILE objects, because it involves a conversion step, which you specify. Proceed as follows.

1. Add the name of the object(s) you want to convert to your worklist as object type RPGLE.
2. Check out the source (option **21**). (There is no need to copy source to development during checkout.)
3. Convert the source member(s) from the RPG in production library directly into your development library, using the **CVTRPGSRC** command (or some other conversion tool) and supplying the appropriate library names.
4. Edit the converted source members(s) in your development library and make any necessary changes (option **32**).
5. Compile the source member(s) into a bound module in development (option **36**).
6. Add the module name(s) to a TURNOVER® for iSeries v100 form for the first application level (option **46**).
7. Run the TURNOVER® for iSeries v100 form to promote the source and bound module(s) to the first level (option **47**).
8. Copy and run the form for any additional unlocked levels.

### ***If after the conversion you want to delete the original RPG source, do the following:***

1. Add the RPG source name(s) to the worklist as a **source-only** type (type code RPGCPY). (**Note: DO NOT check it out!**)
2. Add the source name(s) you're deleting to a TURNOVER® for iSeries v100 form (option **46** beside the **source-only** type at the level where you want to delete the source).
3. Edit the form (option **42**).
4. Select with option **2** the **source-only** lines for RPG source you're deleting.
5. For each line, change the ***Code*** to D for delete. (**F7** lets you update the top portion.) Set ***Delete test object*** and ***Delete test source*** to **N.**
6. Run the TURNOVER® for iSeries v100 form to delete the RPG source member(s) at the target level (option **47**).

If you want to delete source at any other levels of your application, create and run forms for those levels according to these instructions. If you're archiving source in your application, TURNOVER® for iSeries v100 archives the old source before deleting it.

If you have any questions about the information in this document, please contact a UNICOM Systems, Inc. Technical Support Representative by phone, fax, or email at the locations shown at the beginning of this document.