TURNOVER PDQ™

Tutorial

84 Elm Street • Peterborough, NH 03458
Phone: (800) 545-9485 or (603) 924-8818 • Fax: (603) 924-8508
http://www.softlanding.com • Email: webmaster@softlanding.com

TABLE OF CONTENTS

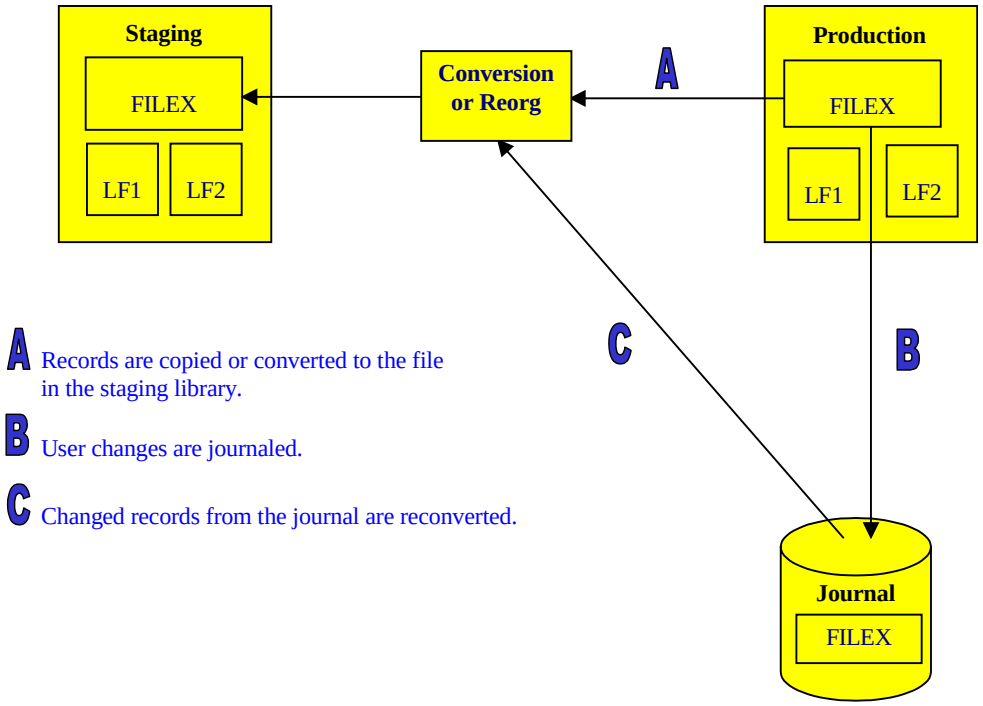
OVERVIEW

This tutorial provides you with a basic demonstration of the TurnOver PDQ process, which allows you to install file changes while your users are accessing those same files. The entire copy/conversion process can be done in the background, before you install the new files and related programs.

The concept behind PDQ is that you do everything in advance. The new version of the files and all the related programs will be compiled (and hopefully tested) in a staging library(s). The object owner, attributes, and object authorities should be set. All that is needed is the production data, then a quick ***move*** of the new objects to production.

As you can imagine, if you had the assistance of a professional Change Management solution, you could further automate, control, and protect this process. This would, in turn, further reduce errors and the risk of downtime. TurnOver PDQ is designed to stand alone, or to snap into any such solution. It is especially well integrated with TurnOver Change Management itself. Later in this tutorial, we will show you how the PDQ process can be managed by the TurnOver Change Management system.

The schematic below illustrates the basic PDQ process for a single file. Two jobs are submitted for each file you are installing. The first job performs the copy/conversion process from the live version of the file in Production to the new version of the file in Staging. The second job is a journal event monitor that simultaneously copies/converts user changes to the live file in Production during the process.



Important Note

This tutorial assumes that you have an RPG or RPG-ILE compiler on the machine. However, while the compiler is required, RPG programming experience is not required. The PDQ Conversion Program Wizard (included) will generate accurate and efficient RPG source for your conversion.

PDQ SYSTEM REQUIREMENTS

The one main requirement for using PDQ is that you must run the copy jobs in an iSeries subsystem that can run the required number of copy jobs *concurrently*. When you go live with this product, we recommend that you set up a job queue that allows an unlimited number of jobs (**MAXJOBS = *NOMAX**) and that you set the ***Job message queue full action*** to***WRAP**.

Some other recommended job description settings are:

- For this tutorial, we will use a job description called TPDQ, which feeds batch jobs into subsystem QINTER. The jobs will run at a batch level priority. Because we will change two relatively small files (the copies will take five minutes or less) and you are most likely working on a development computer, you can probably leave the job description as it is currently defined.

- If you absolutely cannot live with running these jobs in QINTER, you will have to change job description TPDQ to use another multi-threaded job queue. Just make sure that you will be able to run at least five simultaneous jobs, or the tutorial will fail.

- As you will see, all the copy jobs must run concurrently and must stay active until you are ready to install the new versions of the objects. There are several very sound reasons for this:

    - You will want the ability to install a large number of files all at once. Each file will have two jobs running, one to copy or convert the data and one to process the journal transactions.

    - Most importantly, you will want the ability to ***tune*** your system. We used the term *background jobs* earlier. By creating a special job queue/subsystem[1] you can effectively allocate the system resources so that these copy jobs are not slowing down the response times for interactive users and other batch jobs. We will show you how to track the progress of the copy jobs so you can adjust the job run priorities and allocate enough resources to ensure that each copy job completes on time.

[1] See the TurnOver ***PDQ User Guide and Reference*** for a sample subsystem configuration.

INSTALLING DATABASE CHANGES WITH PDQ

In this tutorial, we are going to expand the date fields in two files. Our focus today, however, will be on PDQ. We are assuming that all the necessary analysis, programming, and system testing steps have already been performed; all that is needed at this point is the new data. To get us to that point as quickly as possible, we have provided a production library named ***PDQDEMOPRD*** and a development/staging library called ***PDQDEMODEV***.

RESTORING THE DEMONSTRATION LIBRARIES

Run the command **TPDQINST/RSTPDQDEMO** to restore these libraries for the demo.[2] You can run the command interactively if you want; it only takes a few minutes to restore the libraries from an on-line save file. The command restores these two libraries from save files in the PDQ installation library (TPDQINST). You can rerun the command whenever you need to reset the libraries to reset the libraries to repeat the tutorial or test other scenarios (assuming that the TPDQINST library remains on your system).

On a command line, type the **WRKOBJPDM** (**Work with Objects Using PDM**) command to view the contents of our production library, PDQDEMOPRD. A panel like this appears:

```
                        Work with Objects Using PDM                SLSTRAIN

Library . . . . .   PDQDEMOPRD       Position to . . . . . .
                                      Position to type  . . . . .

Type options, press Enter.
  2=Change       3=Copy        4=Delete      5=Display       7=Rename
  8=Display description        9=Save       10=Restore      11=Move ...

Opt  Object      Type        Attribute   Text
__   ORDER       *FILE       PF-DTA      Order File
__   ORDERL      *FILE       LF          LF over Order File.By Cust ID
__   ORDER1      *FILE       LF          LF over Order File by order date
__   ORDLINE     *FILE       PF-DTA      Order Line.Physical table
__   ORDLINEL    *FILE       LF          LF over Order Line.By Order ID & Sequ
__   ORDLINE1    *FILE       LF          LF over Order Line by Ship Date
__   ORDLINE2    *FILE       LF          LF over Order Line by Ord# and Line
__   QDDSSRC     *FILE       PF-SRC      PDQ demo source for PF's and LF's
                                                                       Bottom
Parameters or command
===> ____________________________________________________________________
F3=Exit          F4=Prompt           F5=Refresh          F6=Create
F9=Retrieve      F10=Command entry   F23=More options    F24=More keys
```

As stated earlier, we are only going to demonstrate the PDQ portion of this change. For that reason, we have supplied two physical files and their dependent logical files, along with the source.

If you want to test your in-house installation programs, you can duplicate some of your own files and programs into these libraries for further testing.

[2] This step assumes you have already restored the TPDQINST installation library and used the **INSTALL** command to load the PDQ software on your system, as instructed in the ***TurnOver PDQ User Guide and Reference***. If the TPDQINST library was deleted after installation, you will have to restore it again to retrieve the demo libraries. Also, you need to make sure that you are signed on with a profile that has enough authority to restore libraries and objects to your system.

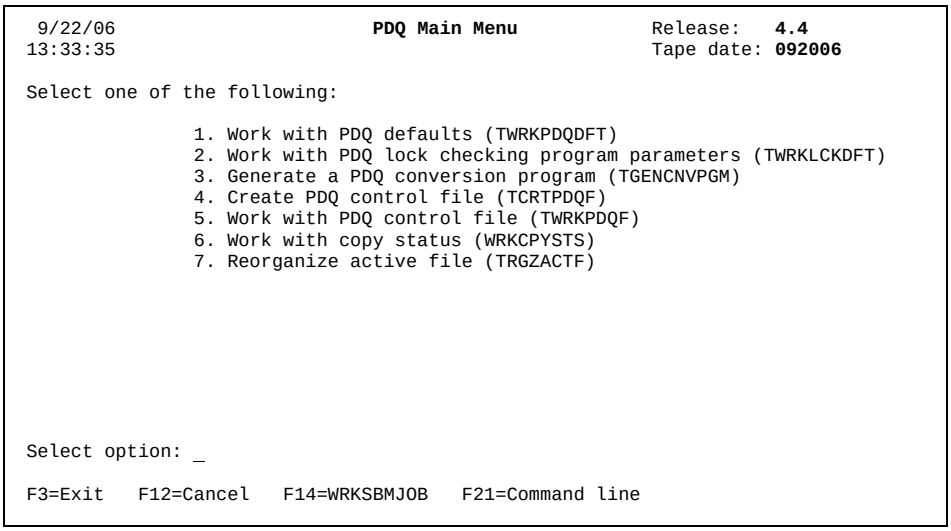
THE PDQ MAIN MENU

Run the following command to modify your interactive library list:

ADDLIBLE TPDQ

Now run the following command to bring up PDQ's Main Menu:

TPDQ

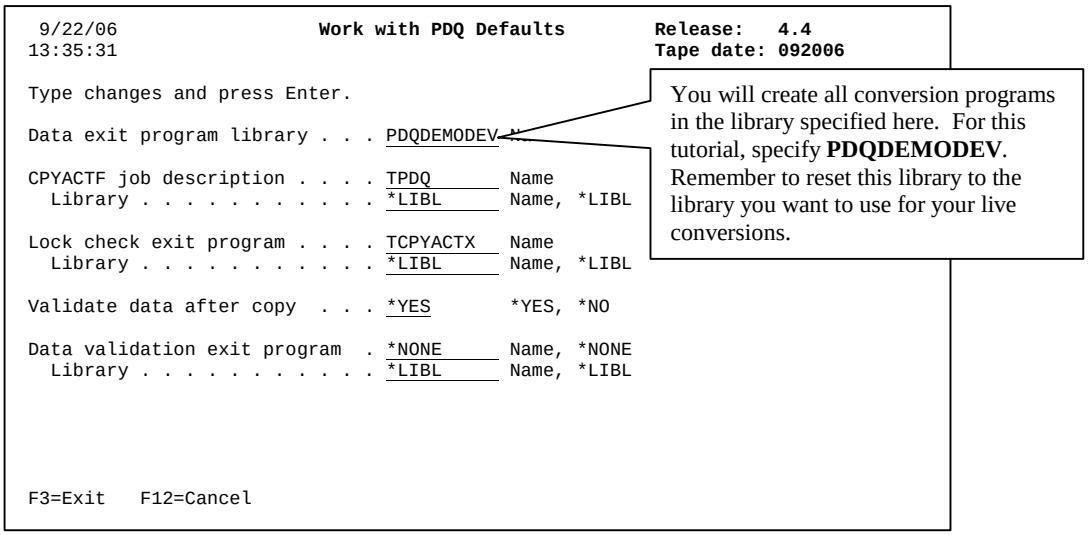


This menu is a collection of the most frequently used PDQ commands. It is meant more as a reference point than a step-by-step path to using PDQ. You will not always need to use each option listed on the menu.

Notice that the command name also appears for each menu item. One of the hallmarks of PDQ is that it is completely API driven. You can run these commands from any authorized command line or program. You can also easily snap PDQ into any Change Management system: your own, SoftLanding's, and even one from another vendor.

WORKING WITH PDQ DEFAULTS

Select option **1** (***Work with PDQ defaults***) on the PDQ Main Menu. A panel like this appears:



Use this panel to review or change default values for the TurnOver PDQ product. For your first run of the product, we recommend the settings shown on this panel.

Descriptions of the parameters for the **TWRKPDQDFT** (**Work with Defaults**) command follow.

Data exit program library

- If you are making a file change that will require a conversion program for installing the new version of the file, you must create the conversion program in the library specified on this line. For this tutorial, we will use PDQDEMODEV.
- If you will be using PDQ with TurnOver, the conversion program you create must have the same name as the file.
- As we will see later, PDQ provides a Conversion Program Generator that creates the conversion programs in the required format. Remember that PDQ is the program that is actually reading and writing records from the old version and new versions of the files. Your conversion program only needs to supply the logic for updating the fields of the record passed to it by the PDQ driver program.
- You will be able specify a copy method of *PRPGM, which will call your data conversion program for each record, or you can specify a method of *PRSRVPGM, which allows you to dynamically bind your data conversion program as a service program. For performance reasons, the service program is the desirable choice for large files.

CPYACTF job description and library

You must specify a job description that uses a multi-threaded job queue. We recommend that you create a job description and job queue that you use exclusively for PDQ. This allows you to control when the subsystem is started and to better allocate the system resources. (For a sample alternative subsystem configuration, see the ***TurnOver PDQ User Guide and Reference***.)

Note: For this tutorial, the TPDQ job description will feed jobs into subsystem QINTER. This is because most iSeries are set up to allow many concurrent jobs to be active in this subsystem. During actual use, you will probably want to change this configuration.

Lock check exit program and library

Specify the name and location of the exit program you want PDQ to call at the end of the copy process. This program's job is to perform whatever operations you desire when PDQ finishes the copy process. For example, you will probably want this program to exclusively lock the files and move the new object(s) in place. If you wrote your own exit program, you would specify that program here.

For this tutorial, we use PDQ's default CL program, TCPYACTX. This program is driven by your responses to the ***Work with PDQ Lock Defaults*** panel described in the next section (see page **13**). It will perform the lock management steps according to your preferences and then immediately end without moving the staging objects into production. This lets you easily repeat the tutorial without having to restore the libraries. The source code for this program is included in TPDQ/SAMPLESRC.

Validate data after copy

When you specify ***YES** for the ***Validate data after copy*** parameter, PDQ compares the number of non-deleted records in the production file (file being copied from) to the number of non-deleted records in the staging file (file being copied to). If the number of non-deleted records does not match between the two files, PDQ displays an inquiry message, asking if you want to recheck the data, cancel the copy operation, or ignore the data discrepancy. You can use a data validation exit program to perform more extensive data validation if necessary.

PDQ uses the value you specify here if, for the **TCPYACTF** (**Process Multiple CPYACTF Jobs**) command, the ***Validate data after copy*** parameter is set to ***DEFAULT**. (For more information, see the next paragraph, ***Data validation exit program and library***, and a brief description of the **TCPYACTF** command beginning on page **29**.)

Data validation exit program and library

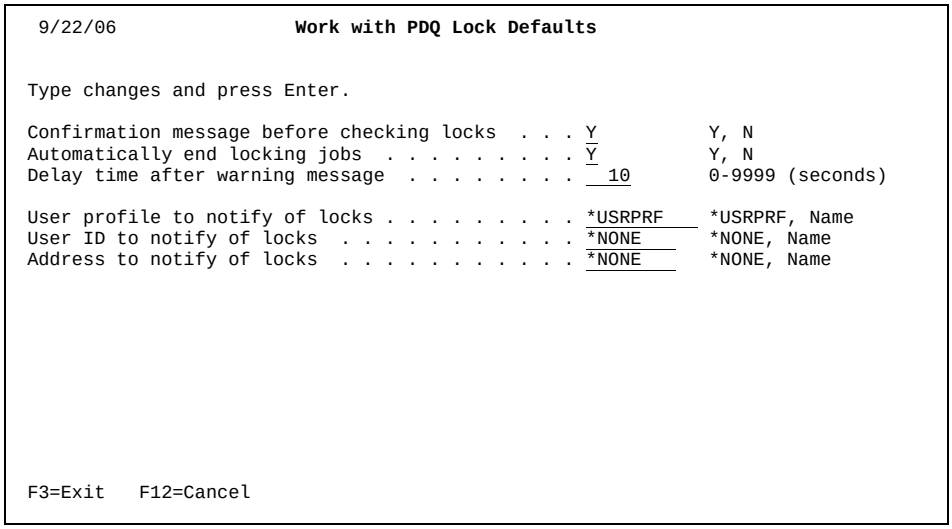
If additional validation is needed, in this field you can specify the name and location of the exit program you want to use. A sample exit program, TCMPPFDTAX, comes with PDQ. This exit program uses the PDQ command, **TCMPPFDTA** (**Compare PF Data**), to perform a record-to-record comparison of data with identical field definitions (name, type, and length) in the two files. The source for this program is included in the SAMPLESRC source file in the PDQ product library.

WORKING WITH OBJECT LOCK DEFAULTS

If, when working with PDQ's defaults, you specified a default lock check exit program (or coded one like it), you need to give it further instructions at this point.

To provide support for various installation methods, we have provided a set of defaults the exit program uses, which will obtain the locks on the production version files you are going to change.

Select option **2** (***Work with PDQ lock checking program parameters***) on the PDQ Main Menu. A panel like this appears:



Descriptions of the parameters for the **TWRKLCKDFT** (**Work with PDQ Lock Defaults**) command are provided on the following pages.

Confirmation message before checking locks

This parameter lets you control when PDQ does the *next step*, which is processing any outstanding journal entries and acquiring exclusive locks on the production version of the files. Specify **Y** here if you want to send a message when all of the convert/reorganize files in your group have reached 100 percent converted.

Example: Suppose you plan to install changes to some of your larger files and you have estimated it will take approximately 30 hours to recopy/convert the existing data to the new formats.

Typically, you would plan to run the copies and conversion jobs over the weekend in order to have the change completely installed by Sunday evening.

With PDQ, you start the data copies and file conversions early in the week, for example on Wednesday morning. The copy jobs and journal monitoring jobs are all submitted. They are running as lower priority jobs, using whatever system resources you have allowed.

On Friday, you check the status of the copy jobs and see that all but one of the copies have finished; there is even a completion percentage and an estimated time of completion, which is 12 noon.

If you set the ***Confirm message before checking locks*** parameter to **Y**, you will receive the message at noon and it will notify you that the copies have completed and that PDQ is ready to move the new objects to production.

In this instance, PDQ is ready, but you are not. You still have hundreds of users who are working and you have to wait for your *predetermined window* time to be reached before you can install the new objects. All you have to do is wait until you are ready to perform the moves. Even though the copies have completed, any changes to the files are captured in the journal and processed by the journal monitoring job.

When the install window time is reached, tell PDQ to continue. It acquires exclusive locks on the files. If anyone is still on, it follows your instructions. Do I send them a friendly message and wait? Do I automatically terminate the jobs and continue? Once the lock is achieved, PDQ rechecks the journals for any non-processed transactions. If there are any non-processed transactions, PDQ recopies/converts those records into the staging file. Now you are ready to perform the moves. If you created everything correctly in the staging library, you will probably be home in time for dinner and, more importantly, did not have to work all weekend.

Automatically end locking jobs

Specify **Y** if you want the lock check exit program to automatically end any jobs that are using the production version of the files you are installing. A message is sent to these users alerting them to exit the application.

This parameter is especially important if you want to install the change ***unattended***. For example, this would be the case if users normally do not work on the weekend and you have written your own CL to submit the PDQ copy process and automatically move the new objects into production.

To ensure that your install job will work, you need to make sure that no other jobs are using any of the files you intend to change. This program alerts anyone using the files to exit the application, and then waits for the specified delay time before ending their jobs.

Using an exit program gives you a lot of flexibility for deciding how to handle acquiring the locks.

Delay time after warning message

Specify the number of seconds you want your program to wait before ending the user jobs. If you specify **0**, any jobs with locks on the files will be ended immediately, with no warning message sent.

User profile to notify of locks

Supply the user profile you want to receive the PDQ "go/no go" message generated by your exit program. This is the profile on the system on which PDQ is running. If you specify ***USRPRF** here, PDQ sends the message to the user profile of the person who submitted the PDQ job. If you are using TurnOver, this is the person who submitted the form job that contains the PDQ conversion instructions.

User ID and address to notify of locks

If you are installing changes on several production systems, you can specify the SNADS User ID and address for sending an additional notification message. Because this is courtesy notification message only, no response is required. This message can go to a different user on the same system or to a user on a different system (for example, to your development box).

CHANGING THE FILE IN DEVELOPMENT

As we mentioned earlier, we have provided a copy of all the objects and source for the production files, in both production and development libraries. To speed up this tutorial, we are assuming that the programmer has already performed the following tasks, either manually, or with the help of a modern Change Management system, as follows:

- Checked out and copied the original source from PDQDEMOPRD to PDQDEMODEV.
- Modified the ORDER and ORDLINE physical files, expanding all six-character date fields to 8-character date fields. (Feel free to view the source in PDQDEMODEV to see what changes were made; they are well marked.)
- Recompiled ORDER and ORDLINE in PDQDEMODEV.
- Recompiled all attached logical files over the physicals in PDQDEMODEV.
- Made sure that all object attributes are correct, especially the number of records, maximum members, and so on.

If this were a real change, the programmer would next change and recompile all the related programs that are impacted by this file change. The programmer would obviously rigorously test the change and ensure that the objects have the correct authorities, attributes, and object owner. (This is the point in the process where the interface to TurnOver's Change Management system comes in especially handy.[3])

[3] For more information on how PDQ interfaces with TurnOver change management, see ***Using PDQ with TurnOver*** beginning on page **45**.

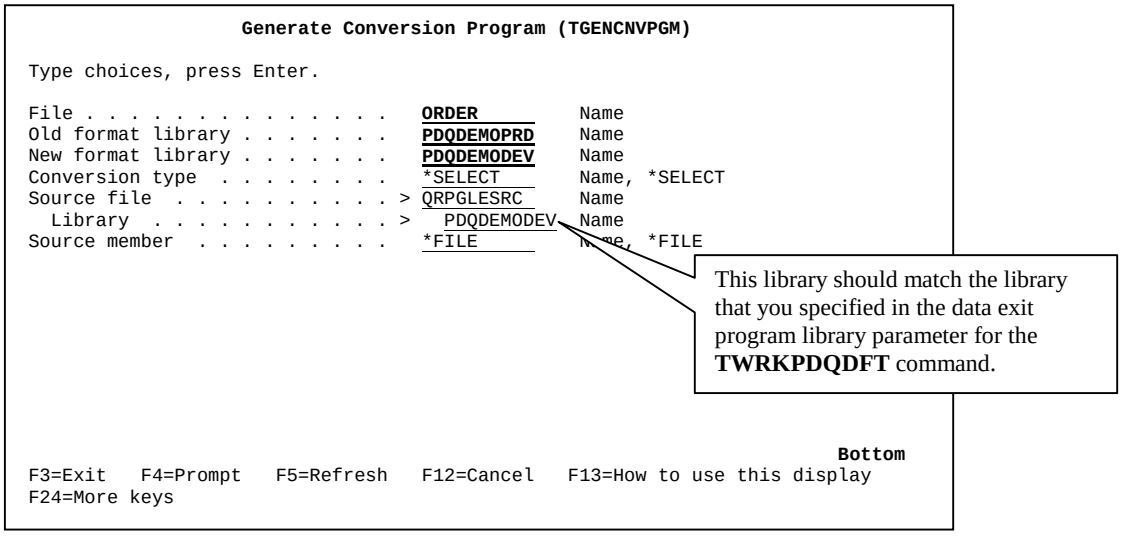
CREATING DATA CONVERSION PROGRAMS

One of the most powerful and impressive features offered with PDQ is a Conversion Program Generator utility. Whether you are adding new fields to a file or changing the attributes of existing fields, you are often faced with the additional burden of writing conversion programs necessary to repopulate the new version of the file. This is often a time-consuming and error-prone process. The Conversion Program Generator utility automates the majority of this process, generating source code that you then compile into a module and use to create a program or service program. This program can be generated as an ILE module that can be tightly bound to the PDQ I/O conversion driver. The end result is a conversion program with performance that is tough to match! An additional advantage of the ILE module method is that you can attach an error-handling module to your conversion to automatically process bad data, without ending your conversion. More information on this follows.

The change we just made (see the second bullet on page **16**) is a perfect example. We expanded the date fields in our order files from 6 to 8 digits. A major part of the installation of these new files will be to convert the existing data in the production library to the new format.

Because we will use PDQ to minimize the application down time, we need to provide a special type of conversion program. Remember, the PDQ driver jobs are the jobs that *read* the records from the old version of the file and *write* the records to the new version of the file. Your data conversion program is executed from within the PDQ driver job. It either calls your program (OPM version) or dynamically binds to it (ILE service program). ***In either case, it is most important that you realize that your program is NOT performing the file I-O; its only job is to convert the data it is passed by the PDQ driver program***.

Select option **3** (***Generate Conversion Program***) on the PDQ Main Menu. A panel like this appears:

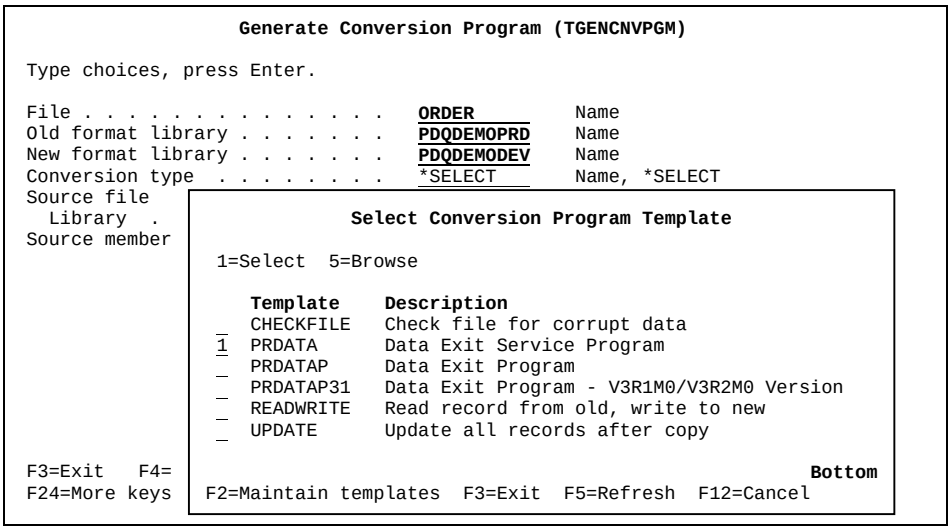


As shown here, you must supply the parameters for the **TGENCNVPGM** (**Generate Conversion Program**) command. We need to generate a separate program for each file.

Important Note

The conversion program you generate will usually be the same name as the file it is converting. Because they are of different object types, both objects can exist in the same library. However, if you already have a program with the same name as the file being converted, make sure you use a library other than your development/staging library for the generated conversion program.

Once you supply the parameters for the **TGENCNVPGM** command and press **Enter**, you see this pop-up panel:



This pop-up panel shows you the standard *program* templates provided with the program generator. Usually, the most efficient (fastest running) method is to create a program module that can later be bound as a service program to the copy active file program. Type **1** by the ***PRDATA*** template and press **Enter**.

A panel like this appears:

```
 9/22/06 14:47:06        Generate Conversion Program

File . . . . . . . . . . ORDER
Old format library . . . PDQDEMOPRD
New format library . . . PDQDEMODEV
Conversion type  . . . . PRDATA
Source file  . . . . . . QRPGLESRC
    Library  . . . . . . PDQDEMODEV
Member . . . . . . . . . ORDER

Type choices for new and changed fields, press Enter.

5=Browse

  Field       Old Definition  New Definition  Conversion
_ OORDDT          6      P        8      P    *MAP
_ OADDDT          6      P        8      P    *MAP
_ OLCGDT          6      P        8      P    *MAP


                                                          Bottom
F3=Exit  F4=Select  F6=Generate  F11=Edit  F12=Cancel  F21=System command
```

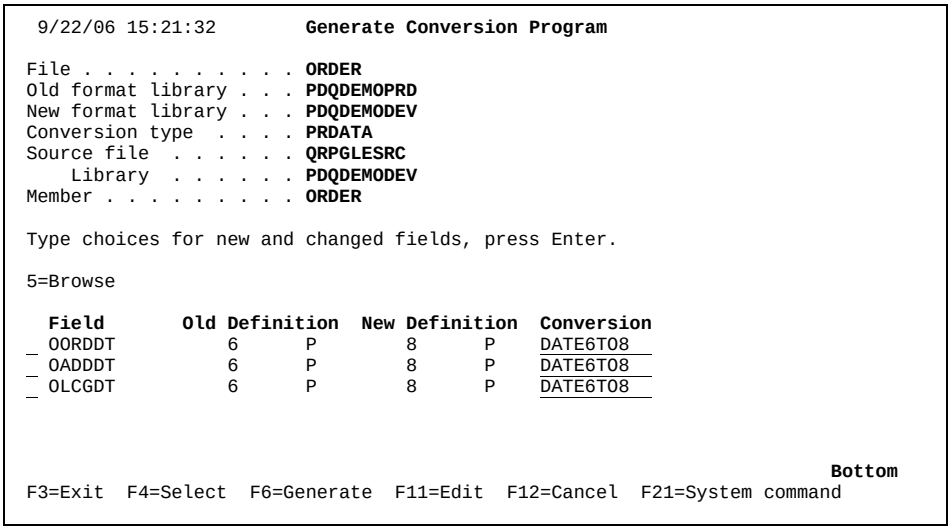
This panel displays the results of a comparison of the file field definitions of the old and new versions of the ORDER file. The fields listed are the ones that have changed between the two versions.

Notice that the ***Conversion*** field is input enabled. Position your cursor on the ***Conversion*** field for the ***OORDDT*** field and press **F4**. A list of field conversion templates appears:

```
 9/22/06 14:47:06        Generate Conversion Program

File . . . . . . . . . . ORDER
Old format library . . . PDQDEMOPRD
New format library . . . PDQDEMODEV
Conversion type  . . . . PRDATA
Source file  . . . . . . QRPGLESRC
    Library  . . . . . . PDQDEMODEV
Member . . .
                    Select Field Conversion Template
Type choices
 1=Select  5=Browse
5=Browse
    Template    Description
  Field         *MAP        Default *MAP *DROP
_ OORDDT        APPENDDATE  Append converted date to end of file
_ OADDDT        DATE6TODAT  Convert YMD date from 6 num. to date field
_ OLCGDT      1 DATE6TO8    Convert YMD date from 6 num. to 8 num.
                DATE6TO8W   Convert YMD date from 6 num. to 8 num. WNDW
                INITIALIZE  Initialize new field to ad-hoc value
                YEAR2TO4    Convert year from 2 num. to 4 num.
                                                          More...
F3=Exit  F4=S  F2=Maintain templates  F3=Exit  F5=Refresh  F12=Cancel
```

Notice that one of the templates, ***DATE6TO8***, converts dates from 6 to 8 digits. Type **1** next to the ***DATE6TO8*** template and press **Enter**. Repeat the process for the remaining fields for this file. Note that you can use any combination of field methods. Our three fields just happen to be using the same method.

If you are curious about what the field templates look like, press **F2** and then browse the lines of source for the template we used. Notice that it is nothing more than a snippet of six lines of logic! This code is inserted in the conversion program to convert the data for the fields we changed. As you can see, you can easily create your own templates for any type of field that you might need in the future.



Now you are ready to generate the actual RPGLE source for this module. Press **F6** to generate the source in the source file and library specified.

Having successfully generated the conversion program's source code for the ORDER file, we will repeat the process for the ORDLINE file. Return to the beginning of this section and repeat the steps, replacing ORDER with ORDLINE. Note that you will see different fields that have changed, but you will still be selecting the DATE6TO8 conversion method.

Once you have generated the source program for both the ORDER and ORDLINE conversions, we will look at the results. To view the generated source code, display the ***Work with Members Using PDM*** panel, change the source file to QRPGLESRC in library PDQDEMODEV, and press **Enter**.

Use option **2** to review the generated source. You will see by the structure of the program that it is only manipulating the data that is passed by the copy program. Search for the value "1940". See how PDQ copied in the field conversion template code for each field. You can easily make any additional changes to this code before you compile the module.

Use option **15** to submit the compiles for the two program modules:

```
                          Work with Members Using PDM                SLSTRAIN

File  . . . . . .   QRPGLESRC
  Library . . . .     PDQDEMODEV          Position to  . . . . .

Type options, press Enter.
 2=Edit         3=Copy  4=Delete 5=Display       6=Print     7=Rename
 8=Display description  9=Save  13=Change text  14=Compile  15=Create module...

Opt  Member      Type        Text
15   ORDER       RPGLE       Order File
15   ORDLINE     RPGLE       Order Line File

                                                                       Bottom
Parameters or command
===>
F3=Exit          F4=Prompt            F5=Refresh           F6=Create
F9=Retrieve      F10=Command entry    F23=More options     F24=More keys
```

After the compiles complete, you must create service programs.[4] We have shipped a PDM user-defined option to help you do this step quickly and easily. If you want to use this option, press **F18** (***Change defaults***) to change your PDM user defaults. The ***Change Defaults*** panel appears:

```
                              Change Defaults

Type choices, press Enter.

  Object library . . . . . .   *SRCLIB      Name, *CURLIB, *SRCLIB
  Replace object . . . . . .   N             Y=Yes, N=No
  Compile in batch . . . . .   Y             Y=Yes, N=No
  Run in batch . . . . . . .   N             Y=Yes, N=No
  Save session defaults  . .   Y             Y=Yes, N=No
  Save/Restore option  . . .   1             1=Single, 2=All
  Job description  . . . . .   QBATCH        Name, *USRPRF, F4 for list
    Library  . . . . . . . .     *LIBL       Name, *CURLIB, *LIBL
  Change type and text . . .   Y             Y=Yes, N=No
  Option file  . . . . . . .   QAUOOPT       Name
    Library  . . . . . . . .     TPDQ        Name, *CURLIB, *LIBL
  Member . . . . . . . . . .   QAUOOPT       Name
  Full screen mode . . . . .   N             Y=Yes, N=No

                                                                       More...
F3=Exit     F4=Prompt     F5=Refresh     F12=Cancel
```

On this panel, specify QAUOOPT as the option file and TPDQ as its library (as shown here) and press **Enter**. Then, on the ***Work with Members Using PDM*** panel, press **F16** (***User options***) to work with user options.

[4] If you used the PRDATAP31 conversion program template, or are NOT generating a service program, skip this step.

A panel like this appears:

```
                      Work with User-Defined Options                SLSTRAIN

File . . . . . :   QAUOOPT          Member . . . . . :   QAUOOPT
  Library  . . . :     TPDQ           Position to  . . . :   ____

Type options, press Enter.
  2=Change         3=Copy         4=Delete         5=Display

Opt  Option  Command
 _      PS    CRTSRVPGM SRVPGM(TPDQ/&N) MODULE(TPDQ/&N TPDQ/TRPGHDLR) EXPORT(*S
 _      SP    /*demo version-do not alter*/ CRTSRVPGM SRVPGM(PDQDEMODEV/&N) MOD









                                                                        Bottom
Command
===> ____________________________________________________________________________
F3=Exit             F4=Prompt           F5=Refresh       F6=Create
F9=Retrieve         F10=Command entry                    F24=More keys
```

On this panel, type **2** (***Change***) next to option **SP**. This panel appears:

```
                          Change User-Defined Option

Type changes, press Enter.

  Option  . . . . . .   SP   Value to change to

  Command . . . . . .   /* create service program */ CRTSRVPGM SRVPGM(PDQ
DEMODEV/&N) MODULE(PDQDEMODEV/&N TPDQ/TRPGHDLR) EXPORT(*SRCFILE) SRCFILE(TPDQ/QS
RVSRC) SRCMBR(PRDATA)
```

Hard code the error handling module ***TRPGHDLR*** into your create service program option.

You must use these values for the **EXPORT** and **SRCFILE** parameters to capture the correct ***Export Signature***.

```
F3=Exit       F4=Prompt       F12=Cancel
```

Important Notes

- Be sure to use the Export source file indicated above. Source file ***QSRVSRC*** in library ***TPDQ*** contains the ***Export Signature*** required for the service program.
- You must include the special error-handling program (***TPDQ/TRPGHDLR***) as an additional module when creating your service program. For information about how you can deal with corrupted data, see ***Dealing with Corrupted Data*** on page **23**.
- For this tutorial, we are creating our service modules and programs in library PDQDEMODEV. During actual use, you will want to change this to your actual library. You should specify that library name in place of PDQDEMODEV in the **TWRKPDQDFT** and **TGENCNVPGM** commands, and in the above user-defined PDM option in the **CRTSRVPGM** command.

DEALING WITH CORRUPTED DATA

Even the most knowledgeable programmer and tightest conversion program have been brought to their knees by unexpected or corrupted data. Who knows how that numeric field got filled with all Zs? As a result, down comes the conversion program! (And why does this always seem to happen during the middle of the night?) No matter when—or how—it happens, it is never a pretty picture.

Note: ***Remember, even if your PDQ conversion fails abnormally, PDQ has not touched your live production data, so all you have lost is the run time.***

Fortunately, PDQ has two features that will help prevent your program from crashing because of bad data. One is an ILE error-handling module that processes error messages generated by corrupted data in your file. The other is a technique that identifies records in your file that contain corrupted data.

ILE error handling module

PDQ includes an error-handling module (***TRPGHDLR***), which you can attach as an additional module when you create the service program you will use to convert a file. This module captures and manages error messages associated with corrupted data, rather than letting them through to the default error processor, which would end the conversion.

The conversion programs you generate this way will use OS/400 APIs to register TRPGHDLR as the error handler for your conversion program. If your conversion program encounters corrupted data in a field, it writes the record to the new version of the file without conversion.

Identifying records with corrupted data

If you think you might have bad data in some of your production files, you can use the **TGENCNVPGM** command to generate a program module, which you then use to create an ILE program to verify a file. You can run this "trial" version of the PDQ conversion in the days or weeks ahead of your actual live conversion to flush out and correct any bad data records. This process makes no changes to your live production data; it is a read-only process.

You can use the **CHECKFILE** program template to generate the ILE module. When using the CHECKFILE template, you do not have to select field-level templates for all the changed fields; you can leave them as *MAP. However, it makes no difference if you do select field-level templates. Also, to prevent the program from being confused with the data conversion program, we recommend that you name it something different than the file. To create the program generated from the CHECKFILE template, use these commands:

1) **CRTRPGMOD MODULE(*anylib/pgmname*)**
2) **CRTPGM PGM(*anylib/pgmname*) MODULE(*anylib/pgmname* TPDQ/TRPGHDLR)**

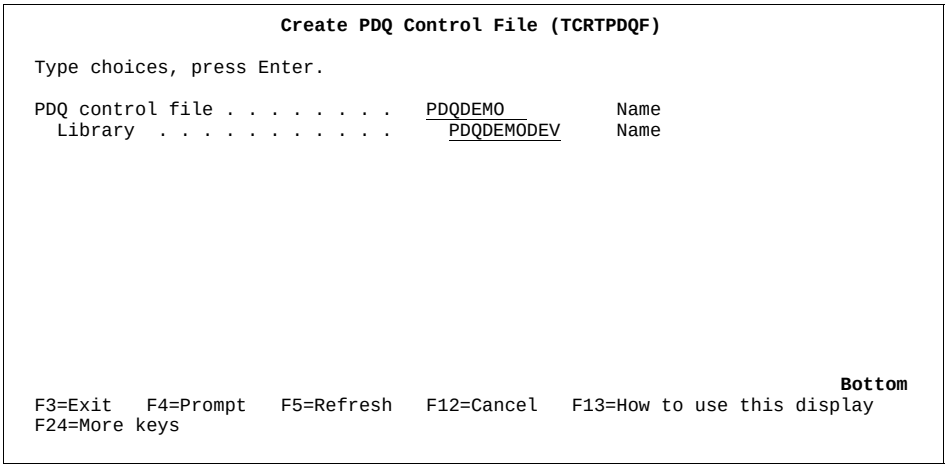
You can run the generated program to produce a report listing of records in the production file that contain corrupted or invalid data. You can manually correct the data after PDQ completes.

CREATING A PDQ CONTROL FILE

The PDQ control file allows you to identify all the files you want to install for this change. You will most likely create a new control file[5] for every change that involves file changes. You can enter any number of files in the PDQ control file. ***The important thing to remember is that PDQ will manage the conversion of all of these files together as one group.***

When you are ready to install your change, you can reference the PDQ control file name you want to install. PDQ then submits the jobs it needs to run for each file listed in the control file.

Select option **4** (***Create PDQ control file***) on the PDQ Main Menu. A panel like this appears:



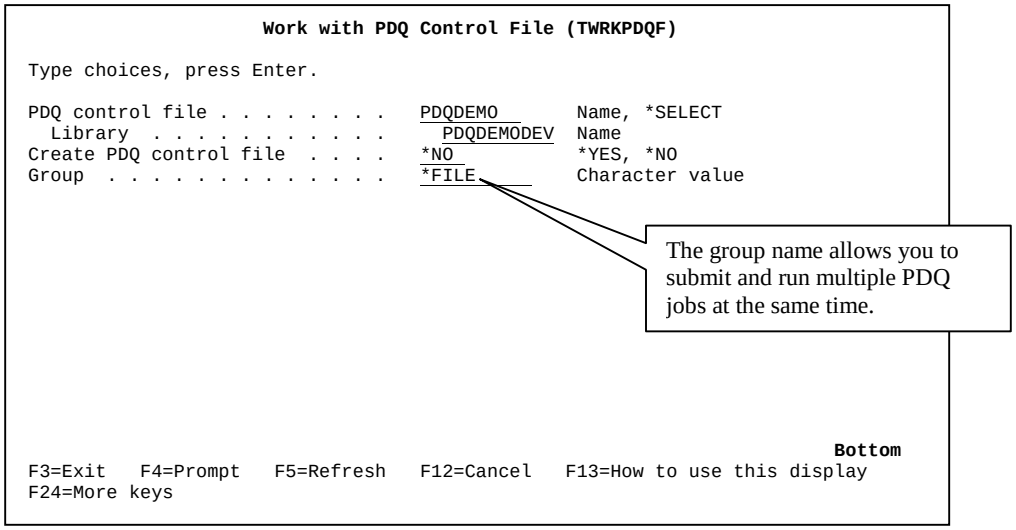
As shown here, supply the name of the control file and the library in which you want it created. Press **Enter**.

If you have developed your own installation programs, you can use the **TCRTPDQF** (**Create PDQ Control File**) command to create the control file from within your program, and then populate it using the **TADDPDQFE** (**Add PDQ Control File Entry**) API for each file.

[5] If you use any type of project management, a good standard to follow is to use that request/task number or reference code for the name of the control files you create.

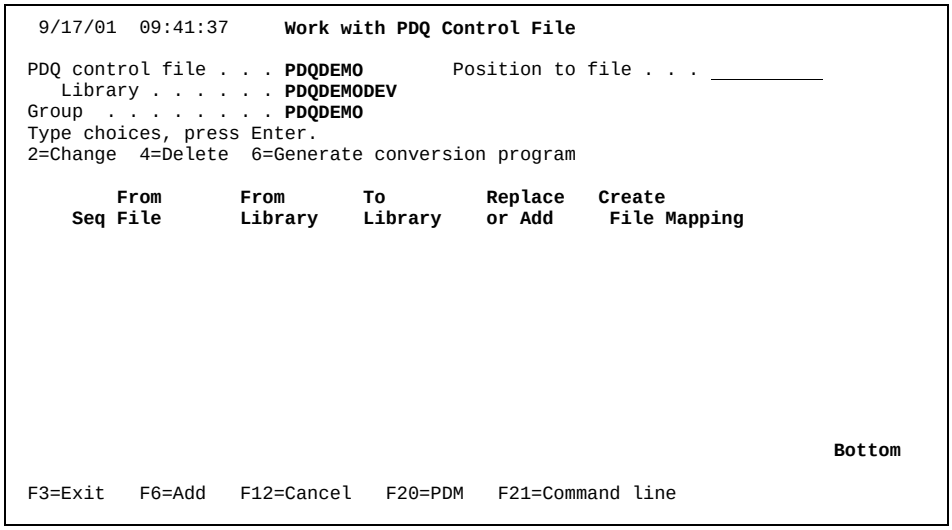
WORKING WITH A PDQ CONTROL FILE

Select option **5** (***Work with PDQ control file***) on the PDQ Main Menu. Supply the name of the control file and library and press **Enter**. Notice that by entering ***YES** for the ***Create PDQ control file*** parameter in this (**TWRKPDQF**) command, you could have combined the steps for creating and working with a PDQ control file.



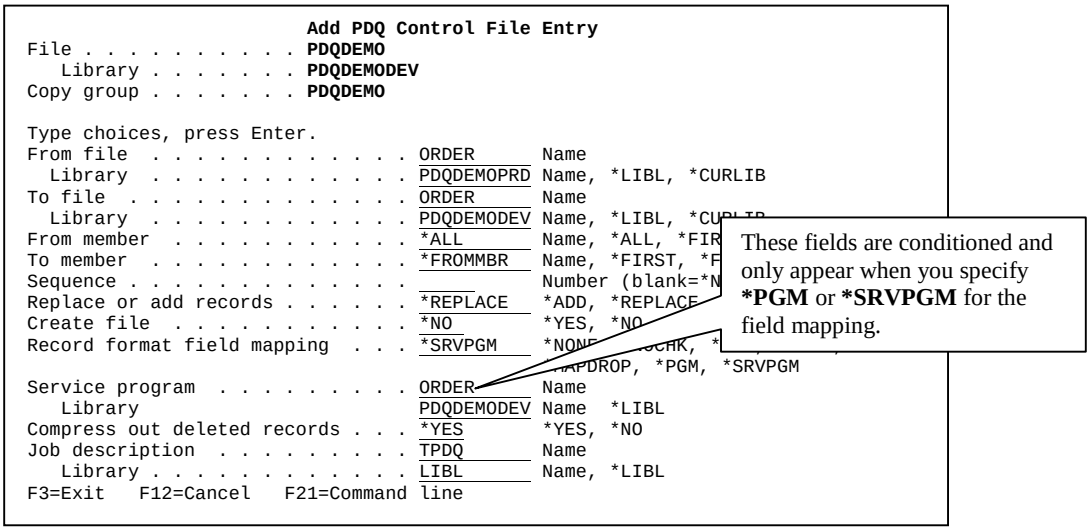
Entering the list of files

Once you have created the PDQ control file, you must then supply the list of files that you will install. When you are ready to start the copy process, PDQ handles submission of all individual copy and journal monitoring jobs.



Press **F6** to add the list of files you will be changing.

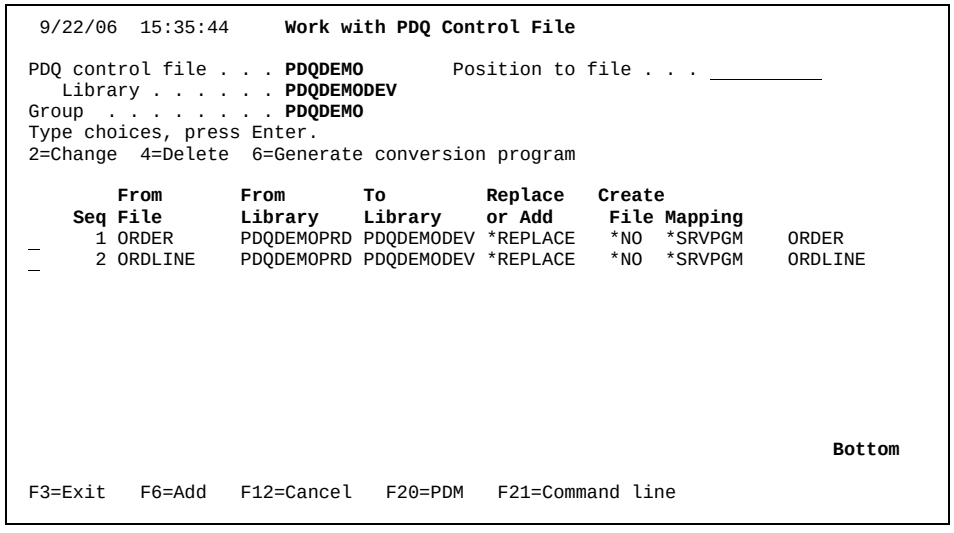
A panel like this appears:



Supply the parameters as shown here and press **Enter**. Because you have entered ****SRVPGM*** as the value for the ***Record format field mapping*** parameter, additional parameters appear for the service program and its library. Type the correct service program and library names, then press **Enter** again to add this file to the PDQ control file. (Repeat the process for the ORDLINE file.)

Note: Make sure you enter the libraries correctly! Remember, you are defining the copy path just as you would with the CPYF command. The ***From file/Library*** is production, the ***To file/Library*** is development/staging.

When you finish, your list of files looks like this:



Defining the copy method

You can define the copy method to use for each file, and the method can be different for each file. Each file will have its own set of jobs, one to copy/convert the data and one to process the file changes from the journal receiver transactions. The copy methods are performing the same function as the equivalent parameter used with **CPYF** command:

*NONE	No field mapping or dropping is done during the copy operation.
*NOCHK	If the record formats of the database files are different, the copy operation continues despite the differences.
*MAP	Fields with the same name in the from-file and to-file record formats are copied, and any fields in the to-file that do not exist in the from-file format are set to the default value.
*DROP	This value must be specified for field-level mapping if any of the field names in the from-file record format do not exist in the to-file format.
*MAPDROP	Fields with the same name in the from-file and to-file record formats are copied. Fields that do not exist in the to-file are dropped.
*PGM	A conversion program of the same name as the file being copied is called for each record during the PDQ copy process.
*SRVPGM	A conversion service program is dynamically bound to the PDQ copy job program. (Initial testing indicates a significant improvement in throughput when copying the records with the bound service program rather than calling a program for each record.)

SUBMITTING THE COPY JOBS

We are now ready to start the installation process. It is important to emphasize that, when doing an actual change, you have created the new versions of all of the objects in the staging library.

You have done everything in advance:

- Set the object owner.
- Reviewed and reset the object attributes, including the number of records attribute on all of the files.
- You have set the authority on the objects. (Make sure the profile you use to submit the copy process has sufficient authority to update the files in the staging library.)

All you need is the production data, then a quick move of the new objects to production.

Simulating user transactions

Before we start the copy process, we need to make sure that one or both of the files are in use and are being changed by active users.

Using another session on this same iSeries, type the **UPDDTA** (**Update Data Using Temporary Program**) command to create a temporary DFU job that will simulate a user updating the files. Make sure you specify the production or "live" file in PDQDEMOPRD. A panel like this appears:

```
                    Update Data Using Temporary Program

Type choices, press Enter.

  Data file . . . . . .   ORDER        Name, F4 for list
    Library . . . . . .     PDQDEMOPRD  Name, *LIBL, *CURLIB
  Member  . . . . . . .   *FIRST       Name, *FIRST, F4 for list

F3=Exit     F4=Prompt     F12=Cancel
                                  (C) COPYRIGHT IBM CORP. 1981, 2000.
```

On this panel, specify the ORDER or ORDLINE file and the production library (PDQDEMOPRD), as shown. Press **Enter**.

Starting the copy jobs

To begin, type the **SBMJOB** (**Submit Job**) command on a command line and press **F4** to prompt that command. A ***Submit Job (SBMJOB)*** panel appears:

```
                              Submit Job (SBMJOB)

Type choices, press Enter.

Command to run . . . . . .   TCPYACTF
                                                                          ...
Job name . . . . . .   *JOBD          Name, *JOBD
Job description  . . . . . .   *USRPRF      Name, *USRPRF
  Library  . . . . . .                      Name, *LIBL, *CURLIB
Job queue  . . . . . .   *JOBD          Name, *JOBD
  Library  . . . . . .                      Name, *LIBL, *CURLIB
Job priority (on JOBQ) . . . . .   *JOBD          1-9, *JOBD
Output priority (on OUTQ)  . . .   *JOBD          1-9, *JOBD
Print device . . . . . .   *CURRENT       Name, *CURRENT, *USRPRF...

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

On this panel, type the **TCPYACTF** command (as shown here) and press **F4** to prompt that command.

When you prompt the **TCPYACTF** command, this panel appears:

```
                  Process Multiple CPYACTF Jobs (TCPYACTF)

Type choices, press Enter.

PDQ control file . . . . . .   TPROMOTF       Name
  Library  . . . . . .           *LIBL         Name, *LIBL
Lock management exit program . .   TCPYACTX       Name
  Library  . . . . . .           *LIBL         Name, *LIBL
Validate data after copy . . . .   *DEFAULT      *DEFAULT, *YES, *NO
Earliest lock check date . . . .   *ASAP         Date, *ASAP
Earliest lock check time . . . .   *ASAP         Time, *ASAP









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

On this panel, type the name of the PDQ control file (PDQDEMO) and library (PDQDEMODEV):

```
                  Process Multiple CPYACTF Jobs (TCPYACTF)          Level: 2

Type choices, press Enter.

PDQ control file . . . . . .   PDQDEMO        Name
  Library  . . . . . . . . .     PDQDEMODEV   Name, *LIBL
Lock management exit program .  TCPYACTX      Name
  Library  . . . . . . . . .     *LIBL        Name, *LIBL
Validate data after copy . . .  *DEFAULT     *DEFAULT, *YES, *NO
Earliest lock check date . . .  *ASAP        Date, *ASAP
Earliest lock check time . . .  *ASAP        Time, *ASAP

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

Note: You can specify the earliest date and time that you want the lock management exit program to check for locks. (For details about the ***Earliest lock check date*/*time*** parameters, see page **41**.)

Press **Enter**.

The ***Submit Job (SBMJOB)*** panel reappears with the **TCPYACTF** command parameters you just specified filled in:

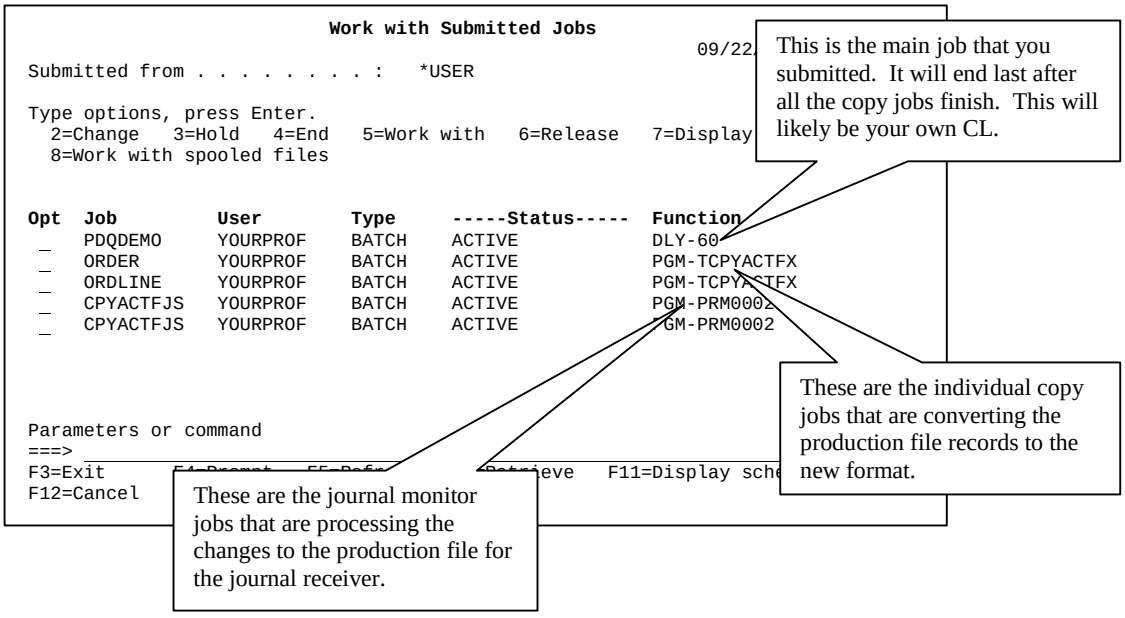
```
                           Submit Job (SBMJOB)

Type choices, press Enter.

Command to run . . . . . .   TCPYACTF FILE(PDQDEMODEV/PDQDEMO)
                                                                        ...
Job name . . . . . . . . . .   *JOBD         Name, *JOBD
Job description  . . . . . .   *USRPRF       Name, *USRPRF
  Library  . . . . . . . . .                 Name, *LIBL, *CURLIB
Job queue  . . . . . . . . .   *JOBD         Name, *JOBD
  Library  . . . . . . . . .                 Name, *LIBL, *CURLIB
Job priority (on JOBQ) . . .   *JOBD         1-9, *JOBD
Output priority (on OUTQ)  .   *JOBD         1-9, *JOBD
Print device . . . . . . . .   *CURRENT      Name, *CURRENT, *USRPRF...

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


Before you submit the job, type a job name and job description, as shown here:

```
                              Submit Job (SBMJOB)

Type choices, press Enter.

Command to run . . . . . . . . . > TCPYACTF FILE(PDQDEMODEV/PDQDEMO)




                                                                      ...
Job name . . . . . . . . . . . . > PDQDEMO       Name, *JOBD
Job description  . . . . . . . . > TPDQ          Name, *USRPRF
  Library  . . . . . . . . . . . >   *LIBL       Name, *LIBL, *CURLIB
Job queue  . . . . . . . . . . .   *JOBD         Name, *JOBD
  Library  . . . . . . . . . . .                 Name, *LIBL, *CURLIB
Job priority (on JOBQ) . . . . .   *JOBD         1-9, *JOBD
Output priority (on OUTQ)  . . .   *JOBD         1-9, *JOBD
Print device . . . . . . . . . .   *CURRENT      Name, *CURRENT, *USRPRF...

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

Now press **Enter** to submit the job. The **TCPYACTF** command handles submission of the individual copy jobs and journal monitoring jobs for each file in your PDQ control file.

Type the **WRKSBMJOB *USER** command on a command line to view all the jobs that are now running by your submitting the **TCPYACTF** command. You see a panel like this:



Note: If you plan to run PDQ using your current installation methods, you will execute the **TCPYACTF** command from within your program rather than using the **SBMJOB** command; your CL install program would be the job running in batch.

WORKING WITH COPY STATUS

Once the copy jobs are running, you can check on the progress of each job. Select option **6** (***Work with copy status***) on the PDQ Main Menu. A panel like this appears:

```
                          Work with Copy Status                    SLSTRAIN
                                                          09/22/06  11:48:33
Type options, press Enter
  4=End   5=Display details   8=Display copy detail   13=Set idle
  14=Release idle

                                        Process              Time to    Copy
Opt  Library     File        Member     Action   Group       Completion  Pct
 _   PDQDEMOPRD  ORDLINE     *ALL       Copying  PDQDEMO      00:04:40    4
 _   PDQDEMOPRD  ORDER       *ALL       Copying  PDQDEMO      00:01:18   15









                                                                    Bottom
Command
===> ____________________________________________________________________
F1=Help   F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve
F11=View record information     F12=Cancel
```

You can view the status of each file, including such important information as the action being performed by the copy operation (***Process Action***),[6] the estimated time left for the operation (***Time to Completion***) and the estimated percentage completed (***Copy pct***). Are you on target for your installation window? This information helps you balance the system resources so the larger file copy jobs get more time.

Next, press **F11** for an alternate view and the record count information. A panel like this appears:

```
                          Work with Copy Status                    SLSTRAIN
                                                          09/22/06  11:48:35
Type options, press Enter
  4=End   5=Display details   8=Display copy detail   13=Set idle
  14=Release idle

                          ---------------File Records----------------  Copy
Opt  Library     File              Total          Copied      Remaining  Pct
 _   PDQDEMOPRD  ORDLINE         39,673           1,863         37,810    4
 _   PDQDEMOPRD  ORDER           12,912           1,956         10,956   15









                                                                    Bottom
Command
===> ____________________________________________________________________
F1=Help   F3=Exit   F4=Prompt        F5=Refresh   F9=Retrieve
F11=View journal event information   F12=Cancel
```

[6] Valid values for this column are ***Idling***, ***AccPath***, ***Locking***, ***Setup***, ***Cleanup***, ***ERROR***, ***Cancel***, ***Done***, ***Copying***, and ***Backup***. See the on-line Help for this panel for descriptions of these values.

You can also check the progress of the journal monitoring job. Press **F11** again to see how many records have changed in the production file, how many have been applied, how many are remaining, and the percentage completed. The journal monitoring jobs are on a timer to only wake up every so often. Therefore, you might notice a lag time before PDQ catches up to any outstanding journal transactions.

```
                          Work with Copy Status                    SLSTRAIN
                                                        09/22/06  11:49:04
Type options, press Enter
  4=End   5=Display details   8=Display copy detail   13=Set idle
  14=Release idle

                           --------------Journal Events--------------  Jrn
Opt  Library     File                Total      Processed      Remaining  Pct
 _   PDQDEMOPRD  ORDLINE               10              5              5   50
 _   PDQDEMOPRD  ORDER                 2              2              0  100









                                                                      Bottom
Command
===> ____________________________________________________________________________
F1=Help   F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve
F11=View general information    F12=Cancel
```

Type option **8** next to one of the files to see additional details about the copy job.

```
                          Display Copy Status                      SLSTRAIN
                                                        09/22/06  11:49:09
View information, press Enter.

Current state  . . :   Block transfer processing

From file  . . . . :   ORDLINE           To file  . . . . . :   ORDLINE
From library . . . :   PDQDEMOPRD        To library . . . . :   PDQDEMODEV
From member  . . . :   *ALL              To member  . . . . :   *ALL

Copy percentage  . :    27               Journal percentage :     0
Time to finish . . :   00:03:36          Group  . . . . . . :   PDQDEMO

Total records  . . :          39,673     Copy active  . . . :   *YES
Records copied . . :          10,910     Copy rate bytes/sec:          12,408
Records remaining  :          28,763

Journal active . . :   *YES              Journal rate/sec . :            0
Total events . . . :               0     Journal name . . . :   PRCX3YJ1MX
Events processed . :               0     Journal library  . :   PDQDEMOPRD
Events remaining . :               0

F1=Help   F3=Exit   F5=Refresh   F12=Cancel
```

Processing the journals

Earlier, we had you start a DFU session to place a lock on at least one of the files. By doing this, we simulated the condition you would have if users were actually updating the files using the production application.

If you want to observe and verify the accuracy of the journal monitoring jobs, you should start changing records in the production files with DFU. Be sure to keep track of the records you change so you can verify that they are converted correctly in the new version of the file.

If you return to the **WRKCPYSTS** (**Work with Copy Status**) command (also option **6** on the PDQ Main Menu), you can see that the changes you are making are being applied to the new version of the file in our staging library.

When doing a real change, you will probably have several files with significantly fewer records. With PDQ, you will always be managing the entire change—all of the files, regardless of the number of records. You are using PDQ to minimize the actual application downtime. You should not necessarily be concerned about the number of jobs that will be running for the copies; the iSeries can handle it.

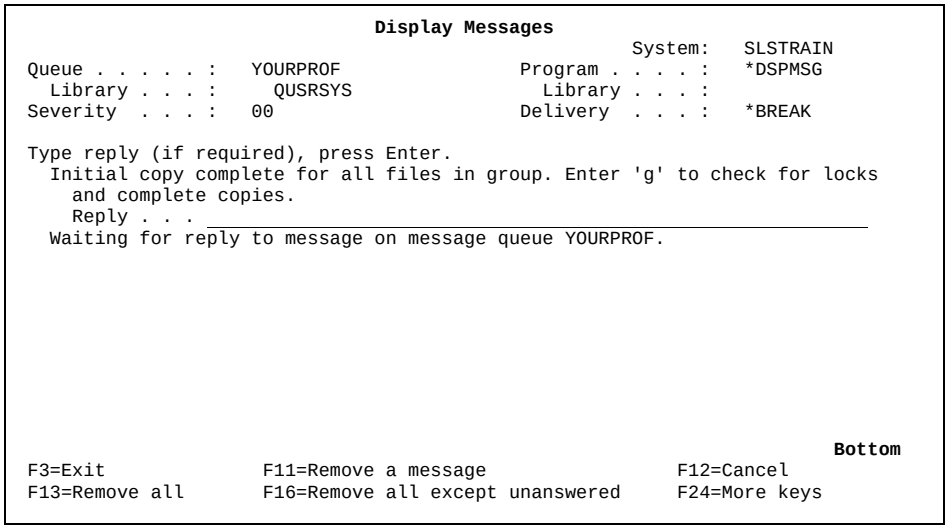
As the copy jobs finish, they go to DLY status and do not take up any additional system resources. The journal monitoring jobs continue to process the changes being made to the production files.

Important Notes

- All of the copy and journal monitoring jobs must remain active until you are ready to actually move the new versions from the staging library to the production library. The reason is obvious; you need these jobs running so that any changes to the production files are processed by the journal monitoring jobs.
- PDQ is completely independent of the actual journal receivers being generated. PDQ picks up the relative record number of the record changed and stores it at the moment the entry gets written to the journal. Therefore, you are free to generate new receivers and delete the old ones as you want.
- If you end any of the copy or journal monitoring jobs (or the subsystem in which they are running), the entire process ends automatically. You then have to resubmit the **TCPYACTF** command to restart the process. Make sure you used ***REPLACE** for each of the files in the PDQ control file. Otherwise, you will add records already present in your staging file.

WHAT TO DO WHEN THE COPY JOBS FINISH

When all of the copy jobs reach 100 percent, PDQ uses the defaults that you supplied earlier to determine what to do next. We specified that the lock check exit program named **TCPYACTX**[7] is to run at this point. This program is going to use the parameters that we earlier supplied with the **TWRKLCKDFT** (**Work with PDQ Lock Defaults**) command.



As you see on this panel, we requested that the lock check exit program send us a message before it attempts to acquire the exclusive locks on the production files. We did this because we are targeting the actual installation of the new objects for after hours. We need to wait until we reach the scheduled install window, which is immediately after the normal workday; you are planning on being home for dinner on time too.

So, we will not answer this message until we are ready to perform the actual install. The PDQ jobs all remain active; only the changes to the production files are being processed during this wait period.

Note: If you planned for the copies to finish at a time when no users are on the system, and you are home in bed, you would have set the lock check defaults to not send the message. Your program[8] would then proceed with the next step of acquiring the locks and moving the objects automatically.

[7] Source for the exit programs can be found in source file SAMPLESRC in library TPDQ.
[8] Assumes that you are running the **TCPYACTF** command from your own CL that will also move the new objects to production.

Obtaining object locks

Before we end any of the PDQ jobs, we must make sure that users are out of the production application, that batch jobs using the files have finished, and that no further updates will be made to the production files.

Based on the values you have specified earlier, PDQ uses the lock check exit program we supply (TCPYACTX) to check for locks on each file in the PDQ control file, and to perform these actions:

- Send messages to any interactive users that are actively using any of the files.
- Wait for the specified time interval.
- Automatically end any user job that has not relinquished its lock on the file.
- Allocate all the files with an exclusive lock.
- Perform the default, basic level of file validation.

Once exclusive locks are acquired, no further changes can be made to any files that are listed in the PDQ control file.

Final updating from the journals

Once the exit program completes, PDQ makes one final check and ensures that every change has been processed from the journal receiver file. Then PDQ ends all of its copy and journal monitoring jobs.

Because you are probably running the PDQ process from within your own CL install program, your program will now have locks on the production files. If you are doing this manually, you will lose the locks on the production files once the original submitted job ends (ours was PDQDEMO).

Make sure you know what you are doing and that you have control of your system. If a user signs back on and makes a change, those changes will be lost when you replace the current production version of the object.

Doing the actual move

You are now ready to move the new version of the objects into the production library. In real life, your exit program (or even better, TurnOver itself!) would handle this critical step. You do not want to make any mistakes now! And remember, this is your downtime window, so you must move quickly!

For this tutorial, our exit program does not perform any moves or deletions. This lets you easily repeat this tutorial without having to restore the save files.

The users can now begin using the application. You were only down for the time it took to remove the original objects and move in the new versions. Now, wasn't that ***Pretty Darn Quick***?

REORGANIZING AN ACTIVE FILE (TRGZACTF)

You can use the **TRGZACTF** (**Reorganize Active File**) command to remove deleted records in a physical file while the file is still actively being used by applications.

Overview of the TRGZACTF command

When you execute the **TRGZACTF** command, it automatically performs these steps:

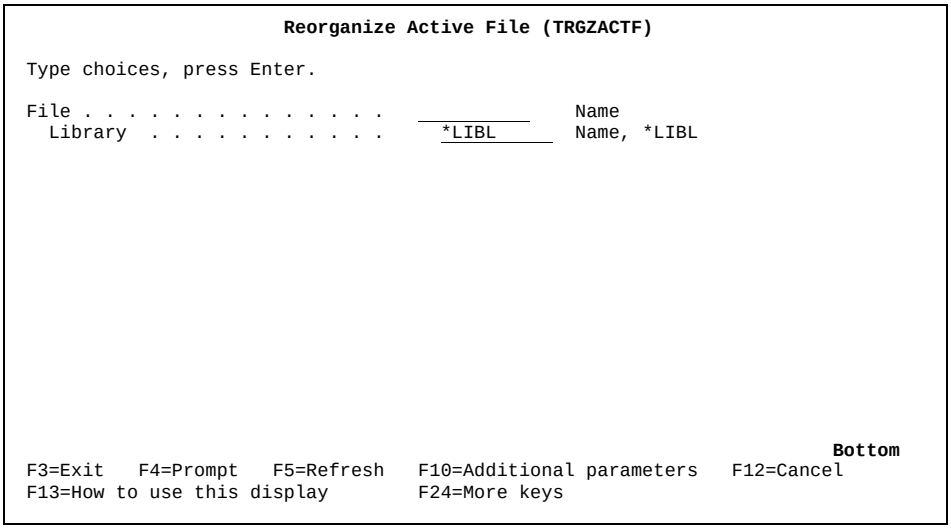
1. Creates a staging library.
2. Duplicates, into the staging library, the physical file and all its dependent logical files. **Note:** The **TRGZACTF** command does not reorganize the physical file unless all the dependent logical files are in the same library as the physical file.
3. Creates a PDQ control file using the **TCRTPDQF** (**Create PDQ control file**) command.
4. Adds a PDQ control file entry using the **TADDPDQFE** (**Add PDQ Control File Entry**) command.
5. Executes the **TCPYACTF** (**Process Multiple CPYACTF Jobs**) command to process the PDQ control file and submit the copy jobs. The **TCPYACTF** command uses the values you specify (for the **TRGZACTF** command) for the ***PDQ control file***/***Library***, ***Lock management exit program***/***Library***, and ***Validate data after copy*** parameters (see page **40**). For information about the operations **TCPYACTF** performs, see pages **29** through **31**.
6. Moves the original file and all its logical files to the archive library upon completion of the copy operation**.**
7. Moves the new file and all its logical files from the staging library to the production library. For additional information about the events that follow the copy operation, see pages **35** and **36**.
8. Deletes the archive library if you specified ***NO** for the ***Retain archived library*** parameter (see the parameter descriptions on the following pages).
9. Deletes the staging library.

Getting started

For this tutorial, let's reorganize the ORDER file in the PDQDEMOPRD library. Before we execute the **TRGACTF** command, however, let's delete some records. Use DFU again to delete (using **F23**) some records from the ORDER file to see how the **TRGZACTF** command handles them.

For your information, the **TRGZACTF** command removes, at the time you execute it, all deleted records that are currently in the ORDER file. Records that you delete while the **TRGZACTF** command is active do not necessarily remain in the reorganized file once the command completes. Whether or not these deleted files remain depends on the timing. As an example, if the ORDER file has 100,000 deleted records when you execute the **TRGZACTF** command, and you delete 1,000 records while the command is active, the resulting file will have, at most, 1,000 deleted records; and possibly less.

Now select option **7** (***Reorganize active file***) on the PDQ Main Menu. A panel like this appears:



Here, specify the name and location of the physical file you want to work with:

```
                      Reorganize Active File (TRGZACTF)

Type choices, press Enter.

File . . . . . . . . . . . . . .   ORDER        Name
  Library  . . . . . . . . . . .     PDQDEMOPRD  Name, *LIBL

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

After you type the file name and library (and before you press **Enter)**, press **F10** to display additional parameters:

```
                      Reorganize Active File (TRGZACTF)

Type choices, press Enter.

File . . . . . . . . . . . . . .   __________   Name
  Library  . . . . . . . . . . .     *LIBL      Name, *LIBL

                          Additional Parameters

PDQ control file . . . . . . . .   *GEN         Name, *GEN
  Library  . . . . . . . . . . .     *PDQ       Name, *PDQ
Lock management exit program . .   *DFT         Name, *DFT
  Library  . . . . . . . . . . .     *LIBL      Name, *LIBL
Staging library  . . . . . . . .   *GEN         Name, *GEN
Retain archived file . . . . . .   *NO          *YES, *NO
Archive library  . . . . . . . .   *GEN         Name, *GEN
Validate data after copy . . . .   *DEFAULT     *DEFAULT, *YES, *NO
Reapply triggers . . . . . . . .   *PRESWAP     *PRESWAP, *POSTSWAP
Earliest lock check date . . . .   *ASAP        Date, *ASAP
Earliest lock check time . . . .   *ASAP        Time, *ASAP
                                                                      Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Parameter descriptions

While the ***PDQ control file*/*Library***, ***Lock management exit program*/*Library***, and ***Validate data after copy*** parameters are used by several commands (including **TCPYACTF**), their values are not necessarily the same for every command. This table lists other places in this document where these parameters are discussed:[9]

For information about …	See …
PDQ control files	***Creating a PDQ Control File*** on page **24**. ***Working with a PDQ Control File*** on page **25**.
Lock management exit programs	***Working with PDQ Defaults*** beginning on page **11.**
Validating data after copy	***Working with PDQ Defaults*** beginning on page **11**.

The additional parameters for the **Reorganize Active File** (**TRGZACTF**) command are briefly described below (and continuing on the next page).

PDQ control file and library

Specify the name of the control file and the library in which you want it created. Type a file name (the control file must not already exist) or specify ***GEN** to have PDQ automatically generate the control file name. PDQ deletes the control file once the **TRGZACTF** command successfully completes its operations. The on-line Help also provides detailed descriptions of the parameters and their values.

Lock management exit program and library

Specify the name and location of the exit program you want PDQ to call once it has completed the copy process. The default lock management exit program, TCPYACTX, is located in the TurnOver PDQ product library.

Staging library

Specify the name of the staging library into which you want PDQ to copy records from the active file. Type a name (the library must not already exist) or specify ***GEN** to have PDQ automatically generate the staging library name. PDQ deletes the staging library once the **TRGZACTF** command successfully completes its operations.

Retain archived file

After it copies records from the active file to the staging library, PDQ moves the original file to the archive library you specify (see the next parameter, **Archive library**). Specify ***YES** to have PDQ keep the archive library on the system once the **TRGZACTF** command successfully completes its operations. Specify ***NO** to have PDQ delete the archive library once the **TRGZACTF** command successfully completes.

[9] The on-line Help also provides detailed descriptions of the parameters and their values.

Archive library

Specify the name of the archive library into which you want PDQ to move the original file. Type a name (the library must not already exist) or specify ***GEN** to have PDQ automatically generate the archive library name.

Validate data after copy

Specify ***YES** if you want PDQ to validate data after it copies records from the production file to the staging file. Specify ***NO** if you do not want PDQ to perform data validation. Specify ***DEFAULT** to have PDQ use its default value to determine if it should validate data.

Reapply triggers

If the file you are reorganizing has triggers, specify when you want PDQ to reapply those triggers:

- Specify ***PRESWAP** to have PDQ:
  - Remove the triggers from the file in the staging library before copying the data.
  - Reapply the triggers to the file in the staging library after copying the data—and BEFORE moving the file from the staging library to the production library.
- Specify ***POSTSWAP** to have PDQ:
  - Remove the triggers from the file in the staging library before copying the data.
  - Reapply the triggers to the file in the staging library after copying the data—and AFTER moving the file from the staging library to the production library.

Earliest lock check date

Specify the earliest date on which you want the lock management exit program to check the file for locks after the copy phase reaches 100 percent. If you specify a date, PDQ will not call the lock management exit program until that date. In the meantime, the **TRGZACTF** command will continue to monitor journal transactions. Alternatively, you can specify ***ASAP** to have the exit program check the file for locks on the same date that the copy phase reaches 100 percent on all files.

Earliest lock check time

Specify the earliest time at which you want the lock management exit program to check the file for locks after the copy phase reaches 100 percent. If you specify a time, PDQ will not call the lock management exit program until that time. In the meantime, the **TRGZACTF** command will continue to monitor journal transactions. Alternatively, you can specify ***ASAP** to have the exit program check the file for locks at the same time that the copy phase reaches 100 percent on all files and, if you supplied a date for **LOCKDATE** (see above), as soon as the earliest lock check date (**LOCKDATE**) is reached.

Note: The previous two parameters are also associated with the **TCPYACTF** (**Process Multiple CPYACTF Jobs**) command. For information about that command, see page **29**.

Submitting the reorganize job

Once you supply the necessary parameters for the **TRGZACTF** command, press **Enter** to submit the job. Then press **F14** (***Work with Submitted Jobs***) to see, as was described earlier, the three jobs that are active for this reorganization:

- Control job (TRGZACTF)
- Reorganization job (ORDER)
- Journal monitoring job (CPYACTFJS).

Because we are reorganizing a relatively small file, you will soon receive the familiar "go/no go" message. Do not reply to this message just yet; first let's have some fun!

From the PDQ Main Menu, select option ***6*** (***Work with copy status***). What you see looks similar to what we saw when we did our data conversion example earlier in this tutorial. Press **F11** to see alternate views. If you used the default value of ***GEN** for the staging library, select the file with option **8** (***Display copy detail***) to see the name of the staging library that PDQ generated for you.

Now, using DFU, change or delete some records. Look at the staging file to see if the records were updated there as well. Also, has PDQ done its job of compressing out the deleted records that were originally present in the file?

When you finish this short exercise, find the "go/no go" message and type **g** to complete your processing.

ADDITIONAL CONSIDERATIONS AND TIPS FOR PDQ

Although PDQ might seem like magic in the amount of application downtime that you save, there are still a few operational items you should consider. These are listed in this section.

- If you are running ***nightly backups*** of your data libraries, make sure you are using the ***save while active*** option or your back up jobs will fail. Also make sure that your backup jobs ***do not automatically end subsystems*** in which your PDQ jobs are running.

- If you are converting or reorganizing a file with ***joined logicals***, make sure you read and understand (this is **IMPORTANT!**) the section entitled ***Distributing forms that use PDQ to process files*** in the ***TurnOver PDQ User Guide and Reference***.

- When you submit a PDQ copy job, if any of the files you are copying encounters an error, PDQ immediately sends you a message to help you troubleshoot the problem. Look at your message queue for the message "**PDQ file copy job has failed for file *library/file*.**" The first of these messages indicates which file encountered the first error.

- As much as possible, ***avoid running jobs*** that will be updating the production files while they are being processed by PDQ. This will create many journal transactions and certainly lengthen the time of your PDQ jobs. Any such jobs that can be temporarily postponed will speed up the PDQ conversion process.

- ***If you are already journaling*** your files, PDQ will use the existing journal receiver for processing the user changes starting from the point in time that the PDQ copy job begins.

- ***If PDQ starts journaling for the file***, the journal file and receiver will be deleted automatically when the PDQ jobs end.

- ***If you will be using conversion programs***, make sure the program name is the same as the file name and that you create the object in the library specified in the ***data exit program library*** parameter in the PDQ defaults. It is also your responsibility to install the conversion program on any remote systems before you start the PDQ process there. If you are using TurnOver, the conversion programs are distributed and installed on the remote system automatically.

- ***If you have created files with LVLCHK(*NO) in the past***, you should create a program with the CHECKFILE template and check the files for bad data.

- ***If you have multi-member files,*** make sure you have created the new versions of these files exactly as they are in the production libraries. PDQ provides special commands that ensure that new versions of the files have the same members, and so forth. Be sure to read the ***TurnOver PDQ User Guide and Reference*** for information about when and how to use these commands.

- ***What happens if the data exit program (or service program) fails***? PDQ will handle the error in the exit program, preventing the job from crashing. The copy job will be ended. If other files are being copied as part of a group, those copy jobs will also be ended (at the next 10 percent interval).

- ***How do you know that it was the data exit that caused the copy to fail***? A joblog will be produced for any copy that ends abnormally. There will be a message in the joblog that says "**Copy active file command failed . . . see joblog for details.**" Shortly before this message, look for an error message sent to the data exit program or service program (if it is a service program, the message is sent to procedure PRDATAXFER). A typical example would be: "**Receiver value too small to hold result (MCH1210).**"

- ***What if there is no error message sent to the data exit program or service program in the joblog?*** Look for a message indicating that a user-defined exit program ended the copy operation. This means that the copy operation was ended because another copy operation in the group has ended abnormally.

- ***Why doesn't the TGENCNVPGM utility stop me from creating an exit program that will encounter errors***? When you select a field conversion template for a changed field, it is not possible for the TGENCNVPGM to analyze the logic of the selected template in order to determine whether or not it will actually work for converting the field. For example, if you converted a date field from 6 to 8, but mistakenly selected the YEAR2TO4 template (instead of the DATE6TO8), your conversion program would get the "Receiver value too small to hold result" error, mentioned previously. This is why it is critical that your conversion programs are tested by copying from QA-Staging to QA (level 2 of model PDQ application) as a "dress rehearsal" before copying from Production-Staging to Production (level 4).

- ***Can I use the DATE6TO8 and YEAR2TO4 templates exactly as they are shipped***? Not necessarily. Before selecting any template, you should browse it and be sure that you understand what it is doing. Besides understanding the template, you will need to know something about your database and the field that is being converted. For example, the DATE6TO8 template assumes that the field contains a date if it is not equal to zero. If the field contains zero, it is converted to zero. If it contains anything else, the century bytes are added. For example, 980722 converts to 19980722. But, does your database use "special" values, for example 999999 = "as soon as possible", or -1 = "date not known". The shipped template would convert these values to 19999999 and 18999999 respectively, which is probably not the intended conversion. If this is the case, then you need to modify the template to account for the special values.

  Note: SLS recommends that you do not change the shipped templates; instead, make a copy and modify the copy.

- ***If you already use a Change Management product***, call us about how you can interface your current product with PDQ.

USING PDQ WITH TURNOVER

Throughout this tutorial, we have mentioned using your own CL install programs for handling the actual move of the new objects into production. This next section is intended for current TurnOver customers, or for anyone looking to ***truly automate*** and "***fail-safe***" the change process by using PDQ for their file changes.

If you are already familiar with, and are using, the many features of TurnOver, skip ahead to ***TurnOver Application Setup*** on page **47**.

If you are interested in learning more about how PDQ interfaces with TurnOver and how your company could take advantage of the many automated features we provide, you should read the next section.

OVERVIEW OF CHANGE MANAGEMENT

To gain a better perspective on the impact change management could have on your organization, let's use the analogy of taking a trip. The bigger the change or the larger the files, the longer the trip will be.

- Doing the entire process manually is like ***walking***. It is going to take you longer to reach your destination. It will be difficult to keep everyone together, going at the same pace and in the right direction. If you have large files or have to install the changes on the weekend and on holidays, you probably will not make the trip very often.

- If you have written your own installation programs, it is like ***taking the trip by bus***. It is certainly better than walking, but there are probably quite a few extra stops and you spend a lot of time looking out the window waiting for your stop.

- If you add PDQ to either of the above scenarios, it is like ***taking a train***. It is more comfortable and there is plenty of room to handle large files. You may even be able to take advantage of the sleeper car while you chug along to your destination.

- Making your change with TurnOver is like ***taking a plane***. Long trips are much easier, so you can take them more frequently. The flight plan is always predetermined, with state-of-the-art safety checks, and it is fully supported by our well-trained maintenance crew. TurnOver's reliability and automated backout and recovery routines ensure that you will always have a ***soft landing***.

The following list of features illustrates just how comprehensive and complete the TurnOver change manage product is.

TurnOver features

EXPRESSDESK HELPDESK™
Logs each call, collects and manages user requests, and reports task status back to the user. Escalation queues ensure nothing is overlooked.

WISEDESK™
Provides the framework for your Q&A decision tree so help desk personnel can answer questions quickly.

APPLICATION DEFINITIONS
Offers predefined development models for fast, easy setup, and fully customizable defaults to handle the unique aspects of your development environment.

INTEGRATED PROJECT MANAGEMENT
Includes enforced workflow and tracks/reports the status of changes from initial user request to distribution on remote production systems.

PROGRAMMER WORKLIST MANAGER™ (PWM™)
Provides a complete view of the development project, worklist of object and source to be changed, and access to cross-reference and source-compare information for fast redevelopment of objects.

SOURCE CODE MANAGEMENT
Provides secure checkout, including vendor-supplied source, version control, and conflict resolution.

OBJECT HISTORY DATABASE
Provides a change registry for all software under TurnOver's control to speed up future development and audit analysis.

OBJECT MANAGEMENT AND INSTALLATION
Automatically handles compile sequencing and database management, including recreation of logical files over physical files—a single-step process that you schedule at your convenience.

OBJECT ARCHIVES
Backs up your source and object code so that if your installation fails for any reason, your application can be automatically restored to its previous state.

SYNCHRONIZER™
Manages your vendor software updates. Includes full source-compare-and-merge utilities.

SOFTWARE DISTRIBUTION
Automatically transmits changes via your choice of tape media or network (SNADS and TCP/IP supported).

REMOTE INSTALLATION
Receives, unpacks, and installs changes in libraries (recompiling objects, if necessary) and passes status at every step to the source system without manual intervention.

VENDOR SOFTWARE INTERFACES
You can use TurnOver's many APIs to bridge to any vendor software package, but we include special interfaces for:

- J.D. Edwards®
- MacPac™
- BPC®
- Infinium™
- CA/PRMS™
- ASI™

CASE TOOL INTERFACES

- LANSA™
- AllFusion 2E
- AllFusion Plex
- AS/SET®
- ProGen Plus™

And TurnOver's many APIs make it easy to hook into any other similar tool.

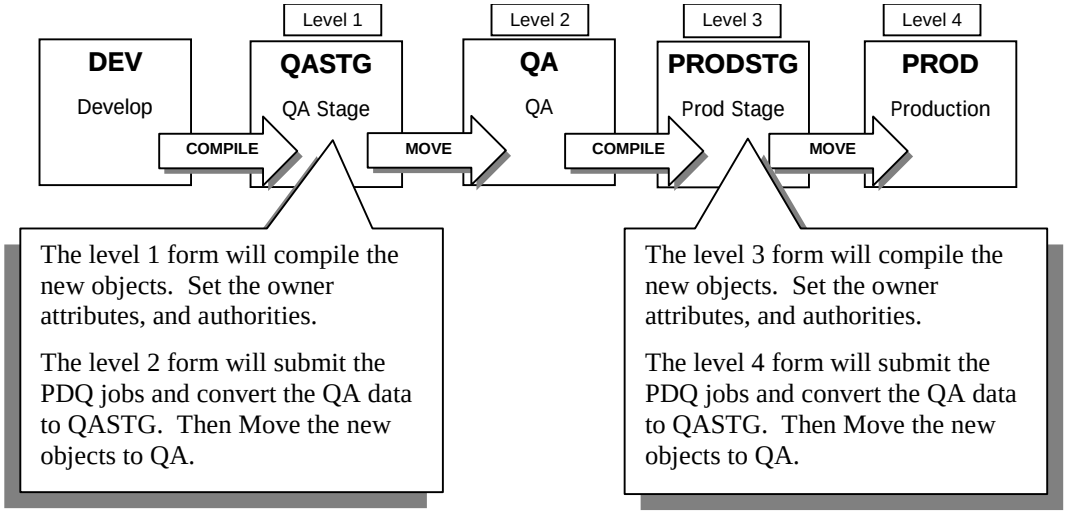
TURNOVER APPLICATION SETUP

The TurnOver – TurnOver PDQ interface is seamless. The only requirement is that you must have a Staging environment defined prior to the environment to which that you will be promoting. The Staging library allows TurnOver to move the objects from the Staging level to the Target environment. For most customers, this means that you will be inserting a new level(s) in your applications so that we will not move the data objects from the QA level where they will likely be needed for future testing.

The ***Copy Option*** parameter used on each PF line of the TurnOver form will then control the PDQ process; we will discuss the copy options a little later.

Most TurnOver customers use a two-level application for development. To ensure that you fully test the PDQ process, as well as your conversion routines, we recommend that you insert a Staging level in front of your current QA and production levels, as indicated in the following schematic.

Recommended Setup



This type of a setup will let you create the test data in the QA environment with the same conversion routines you will be using when you run the production level form. The level 2 promotion is essentially performing a ***dress rehearsal*** of the production install on a limited number of records. The QA testing will now be done using the test data that was converted by the same process that will run on production.

Note: See the TPDQ application for an example or, for a more detailed discussion, see the ***TurnOver PDQ User Guide and Reference***.

Copy options

The TurnOver form's ***copy file FMTOPT*** parameter controls how the data in the target environment will be handled with each promotion form.

The following special copy option values have been added to initiate the PDQ process from the TurnOver form jobs:

***PRNOCHK**	Equivalent to **CPYF** w/ ***NOCHK**
***PRMAP**	Equivalent to **CPYF** w/ ***MAP**
***PRDROP**	Equivalent to **CPYF** w/ ***DROP**
***PRMAPDROP**	Equivalent to **CPYF** w/ ***MAP *DROP**
***PRPGM**	Program for data conversion
***PRSRVPGM**	Service program for data conversion

Using our example application on the previous page, these values would be used for the level 2 and level 4 forms. Because there will be no objects, and therefore no data to re-copy, in the staging library, the level 1 and level 3 forms should have ***NOCOPY** for the default copy option in the application definition.

RUNNING A TURNOVER PROMOTION FORM

When the PDQ level forms (2 & 4) are run, the form job interrogates the PF lines and creates a PDQ control file entry for any PF lines with a *PRxxx copy option. Once the control file is loaded, the TCPYACTF job command is submitted for the control file just created by the form job, and:

- The PDQ control file name will be the same as the TurnOver form job (T########G).
- Each file will be added to the control file with ***REPLACE**.
- The PDQ Group name will also be the same as the TurnOver form job (T########G) name so that you will be able run, identify, and track the progress of multiple forms or groups of forms.

Once the all copy jobs finish, the PDQ exit program will send you a message so you can delay the actual installation until your predetermined install window, or you can acquire the object locks immediately and return control to the form job.

When control is returned to the TurnOver form job, the original objects are moved to the from archive library, and the new versions of the objects are moved to the target libraries.

DISTRIBUTION

You need to install PDQ on each production system and configure the PDQ defaults and lock check exit program as specified earlier.

Typically, TurnOver automatically receives and restores the TurnOver distribution save file on each production machine. This process reconstructs the form installation information to the form and line files on the production computer. The changed objects will be restored to a T*form#*S library.

YOU ARE STRONGLY CAUTIONED!!

If you are going to distribute, you must follow the setup recommendations for your remote application definition(s) that are described in the section entitled ***Distributing forms that use PDQ to process files*** in the ***TurnOver PDQ User Guide and Reference***. If you do not set up your remote application definition(s) properly, you could have serious data corruption problems.

Note: Any conversion programs that you have created on the development computer are automatically included in the distribution library that is sent to each production system. These programs are then restored to the data exit program library defined in the **TWRKPDQDFT** (**Work with PDQ Defaults**) command on each production system.

ADDITIONAL CONSIDERATIONS AND TIPS FOR TURNOVER

Application choices

Should you modify your existing applications or create new ones to be used only for major file changes? Good question! Unfortunately, you will have to decide which way is better for your company. For what it is worth, an informal poll of the SLS developers revealed that they would use a separate application for the PDQ file changes. Here are some of the issues that might help you with your decision:

- Your object change history will be more accurate and easier to use with everything under the same application.

- Using the same libraries in different applications causes the cross-referencing to ask which application to use when objects need to be recompiled. TurnOver defaults to the application you are using, but some people find it easier to use a single application.

- There will be two additional lines on your worklist if you add the two staging levels. Because the source library is probably the same for levels 1 & 2 and levels 3 & 4, you will see two lines highlighted because of the duplicate source library names. You can use the worklist filtering if you find this confusing.

Running multiple forms

Inserting additional levels in your application means that you have to create and run forms for each level of the application. Because most of your changes will not involve file changes, you may find it annoying to have to run the extra forms required to pass through the staging levels. You can use a TurnOver exit to automatically create and run the second form. Using the example application we described earlier, this is how it would work:

- The programmer creates and runs the level 1 form to promote to QASTG. A TurnOver EXIT01 program is called.

- If the level 1 form RAN-OK, the program checks to see if there are any PFs on the form. If there are no PFs, the form is copied and run to the next level using the CPYFORM and RUNFORM APIs.

You can add additional logic to control which levels to automate and how form approvals may have to be handled.

Sample source for programs EXIT01PDQ and EXIT01C is provided in source file SAMPLESRC in library TPDQ.

Copying forms

Using the example application we described earlier, you will be required to enter the appropriate *PRxxx copy option manually when you create the level 2 form. When you create the higher-level forms, copy the level 1 form(s) to create level 3 and copy the level 2 form(s) to create level 4. This eliminates having the manually reenter the copy option for each form.

Making a file change without PDQ

You can use any valid copy option on a TurnOver form. If you are installing a file with a small number of records, you might choose to use one of the standard installation options. When the TurnOver form runs, it will only submit PDQ jobs for those lines with *PRxxx in the copy option. When the actual install step occurs, the original objects are moved to the TurnOver form archive library. If you specified the standard copy option (***MAP *DROP**) for any of the PF lines, the data will be copied from the archive library, as usual.

