Supplement #63

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 MULTIPLE VERSIONS

TABLE OF CONTENTS

You should read this document if your organization has the following traits and requirements:

- Is a software company;
- Needs to support custom modifications for several customers or locations from a central location;
- Introduces changes to your software periodically, as releases;
- Needs to maintain multiple national language versions of your software;
- Has requirements that span more than one of the topics above.

OVERVIEW

TURNOVER® for iSeries v100 employs several unique features to enable you to manage more complex software development environments. This is an advanced TURNOVER® for iSeries v100 topic. In this document, we assume that you already have a working understanding of TURNOVER® for iSeries v100 application definitions.

About versions and releases

With its library structure, the iSeries is ideal for managing multiple releases of software. As you start work on a new release, you create a new "release" library, start modifying objects from the last release and promote them to your new library. It doesn't matter whether you call the work in your new library a 'Release', a 'Version' or 'Product/2001'. The important thing is that you intend to introduce all the changes to your application together.

This chapter describes how to manage several application releases. But we also touch on object 'versioning' in this document (see page 27). What's the difference? An object version number makes it easier to distinguish one version of an object from its predecessors or successors. In other words, it's a number that indicates how many times the object has been modified. Frequently, this has little to do with the application version (or release) number. After all, you may be preparing Payroll Release 4.0, but the payroll print program may have been modified fifty times over the life of the application. Therefore, its version number might be '050'.

Object version numbers help you verify what version of an object your user has installed. Rather than saying to your user, display the object description and tell me the date, the time, the source member information, and so on, you can simply ask, "What is the version number?" It's shorthand. You still can retrieve all of those other details, and they are just as valid as a version number, but a single version number is easier to verify. Later in this document (see page 27) we explain how TURNOVER® for iSeries v100 accomplishes object versioning. For now, just keep in mind that application "versions" and object "versions" are different.

Related applications

To understand how TURNOVER® for iSeries v100 manages multiple versions of software, you must understand TURNOVER® for iSeries v100's ***Related Application*** feature. The diagram below illustrates a related application in its most basic form.[1] Each box represents one or more libraries defined to that level of the application.

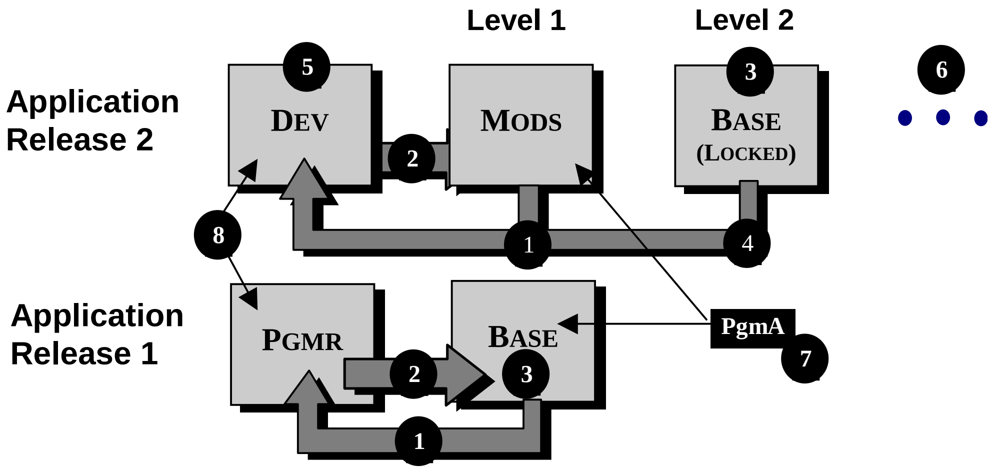


Figure 1

If you had to support more than one release, then both of these applications might be active, simultaneously.

[1] More complex scenarios are illustrated later in this document.

How do related applications work?

Here are the basic rules:

1 Checkout is from right to left; 2 promotions are from left to right.

3 The 'BASE' application library is defined to each application.

The BASE library for Release 2 is locked. Changes are not promoted to locked levels for any application definition. (However, the BASE library can be changed using the Release 1 application.) MODS libraries contain just the objects that you have modified for that release.

Release 2 is defined as 'related to' Release 1 (see panel on page 7).

4 If, using Release 2, you check out an object that has not been changed before (and it therefore does not exist in the MOD library), TURNOVER® for iSeries v100 will look for it in the locked libraries, defined at a higher level, starting from left to right.

5 If an object was already in development in a related application, you will be informed of this 'conflict' condition (see page 12).

6 More than one locked level can be defined (see examples on page 21).

You must define application unlocked levels consistently across a set of related applications. That is, if you have one test level for the base application, you must have one test level for each related application.

7 Object names must be identical across related application boundaries.

8 When you check out an object from more than one related application, be sure the applications are set up so that they do not overlay each other in the same library. In this example, the base application is set up to check out to the programmer's library; the related application is set up to check out to a second library. Where many related applications are defined, we recommend that you define development libraries for each one.

Additional rules

A **Base** application definition defines the entire application, exclusive of those objects that have been modified for any location.

A **Related** application consists of a subset of objects copied from the parent application and modified for use at one or more locations.

For each related application, the library list used to run the application will consist of the related application library(s) followed by the Base application library(s).[2]

Objects and source that exist in the base may exist in one or more Related application libraries in their modified form. Object and source names must match in base and related libraries.

A Source member and object may exist in a Related application library without being present in the Base application library. These apply only to that version of the application. If such an object exists in several related application libraries, they must be changed independently. In order for TURNOVER® for iSeries v100 to recognize objects as related, they must exist in the Base library(s).

Users who are authorized to the Base should also be authorized to each of the related applications. Users can be authorized to a related application without being authorized to the Base.

Note: A related application could represent a release, as it does in **Figure 1**, or it could represent a custom modification for a customer or location, a second language version of your application, or a combination of release, custom modification and language version.

[2] In some instances, base objects may be merged into a new release library so that fewer libraries need to be present in the library list.

Related applications and cross-referencing

TURNOVER® for iSeries v100's related application and cross-referencing features work closely together. If you check out a file from one application that is referenced by a program in another, whether it be defined as related or not, TURNOVER® for iSeries v100 will show you the relationship. Although you can use the related application feature without building a cross-reference database, TURNOVER® for iSeries v100 cannot predict the impact of your change. Therefore, we urge you to do both. If you are managing multiple releases, we recommend that you create a new cross-reference table for each new release of your application.

UNICOM Systems, Inc. Recommends

If the related application represents customer modifications and if you distribute changes to customers, then we recommend keeping the customer modifications in separate libraries from the base application on the customer's computer. If changes are installed into one, consolidated library on your customer's computer, it is essential that they be installed in the correct order to avoid overlaying customized objects with those from the base application. TURNOVER® for iSeries v100 will handle the installation correctly. Normally, all changes are run in the same Group Form job, so this is not a problem. However, if you were to distribute and apply changes manually, they could end up being installed in the wrong order.

Related application change scenarios (refer to Figure 1)

The examples that follow will help you better understand how related applications work and how these rules are applied:

A change is made to an (older) program in Release 1.

If it exists in MODS (in Release 2), you might have to make the same change there too.

You can check it out from Release 1 and Release 2. TURNOVER® for iSeries v100 prompts you to do so.

If you decided not to check the program out of MODS, TURNOVER® for iSeries v100 considers that a conflict and will not allow you to promote the object to BASE without verifying that you have resolved the conflict (see page 13).

If the program is already checked out of MODS, then a conflict exists that you will have to resolve. You will either have to make a similar change to the related object or to explain why you didn't (for example, it might have been completely rewritten for Release 2).

If the program doesn't exist in MODS, then you check it out, change it and promote it using just the Release 1 application.

A change is made to a physical file for Release 2. (Assuming you use TURNOVER® for iSeries v100's own cross-reference):

If the file does not exist in the MOD library, it will be checked out and copied from the BASE library and promoted back to the MOD library.

If the file exists in the MOD library, it will be checked out of the MOD library.

If logical files exist over the physical file, but are not changed, they will be created in the MOD library.

A program that uses the physical file and does not exist in the MOD library will be created there from the source in the BASE. (Think about it!)

Note: The cross-reference file is built in a table for Release 2 using all the application libraries, including those in the locked levels.

A change is made to a file for Release 1.

If the file has been modified and exists in Release 2, you'll be prompted to check it out. If you decline to do so, you need to state a reason in order to avoid a conflict.

If programs exist over the physical file in either Release 1 or 2, they'll be recompiled when you promote the file to the base.

Each application will use its own cross-reference table and will recompile the correct related objects for each release.

These are just a few examples to clarify the operation of the related application feature.

SETTING UP APPLICATION RELATIONSHIPS

You can interrelate application definitions so that when you change one application, TURNOVER® for iSeries v100 prompts you to check if a similar change is required for a related application. In application relationships, the base application is referred to as the *parent* application; a related application is referred to as a *child* application.

Before you can set up application relationships, you must first define each of your applications independently (see ***Chapter 1: Working with Application Definitions*** in the ***TURNOVER® for iSeries v100 User Guide***). It's important that each application be defined similarly. Here is a sample schematic:

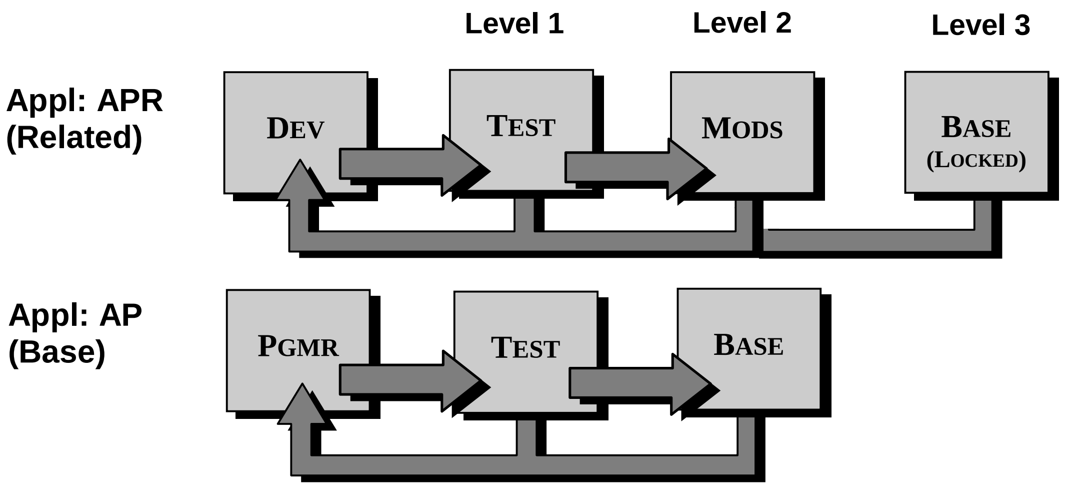
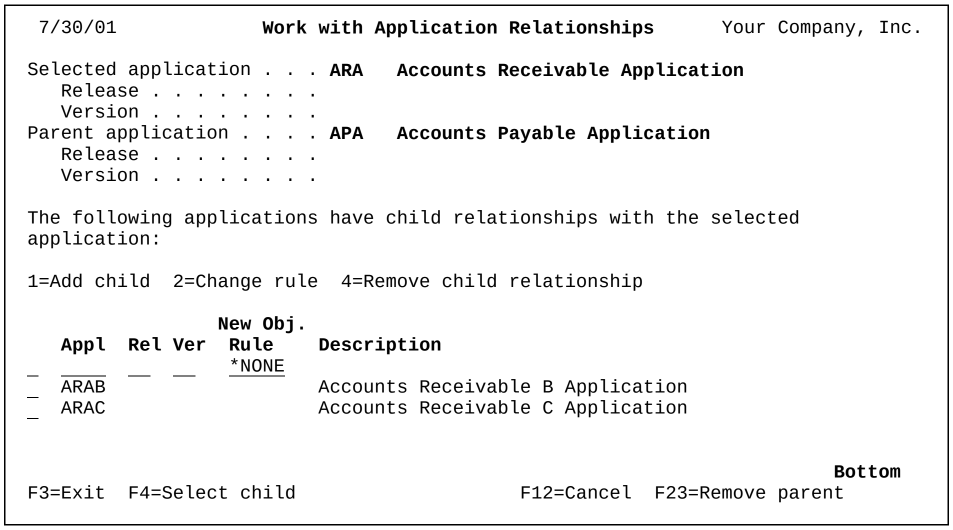


Figure 2

This is similar to the example in **Figure 1**, except that there are two unlocked levels, levels 1 and 2.

Once you have defined each application, you can begin working with application relationships. On the ***Work with Application Definitions*** panel, select an application definition with option **19** (***Application relations***). A panel like this appears:[3]



```
 7/30/01             Work with Application Relationships      Your Company, Inc.

Selected application . . . ARA   Accounts Receivable Application
   Release . . . . . . . .
   Version . . . . . . . .
Parent application . . . . APA   Accounts Payable Application
   Release . . . . . . . .
   Version . . . . . . . .

The following applications have child relationships with the selected
application:

1=Add child  2=Change rule  4=Remove child relationship

                 New Obj.
   Appl  Rel Ver  Rule    Description
_  ____  __  __   *NONE
_  ARAB                   Accounts Receivable B Application
_  ARAC                   Accounts Receivable C Application


                                                                       Bottom
F3=Exit  F4=Select child                   F12=Cancel  F23=Remove parent
```

This panel shows all existing application relationships for the application you selected. The names of the selected application (ARA) and its parent application (APA) appear at the top of this panel. If the selected application has any child relationships, the names of those applications are listed on the panel's bottom half (ARAB and ARAC). Options are available for adding and removing relationships between the selected application and its child applications, and for setting new object rules for these relationships.

Nesting related applications

As stated earlier, the base application definition is the *parent*. It defines the entire application, exclusive of those objects that have been modified for any location. A related application is the *child*. It consists of a subset of objects copied from the parent application and modified for use at one or more locations.

Note that an application related to one parent can be a parent itself. In other words, TURNOVER® for iSeries v100 supports 'nested' related applications. As an example (see **Figure 1**), you might have several customer modifications applications related to Release 1 and several others related to Release 2. In that case, the Release 2 application would have several other applications related to it.

In our Application Release example (**Figure 2**), it is likely that you would start a Release 3 someday. That would be related to Release 2. (For an illustration of nested application, see ***Advanced related applications***, beginning on page 19.)

[3] You can also display the ***Work with Application Relationships*** panel by selecting an application level with option **19** (***Application relations***) on the ***Work with Application Definition*** panel or by typing **1** next to ***Application relations*** on the ***Define the Application*** panel. For detailed information, see ***Chapter 1: Working with Application Definitions*** in the ***TURNOVER® for iSeries v100 User Guide***.

USING RELATED APPLICATIONS

There are two points in the change process where related applications come into the picture: when you check out source for an object and when you set up TURNOVER® for iSeries v100 forms to promote your changes. These features are supported on the ***Programmer's Worklist***; therefore, let's assume you are using that facility.

Working with multiple versions

The Programmer's Worklist, illustrated on page 11, allows you to work on all of the related objects across all of your related applications at one time. Referring to **Figure 3**, when you check out an item from the AP BASE library, TURNOVER® for iSeries v100 will search for the same object in the MODS library and APRTEST. It will also tell you if it is checked out for the APR application.

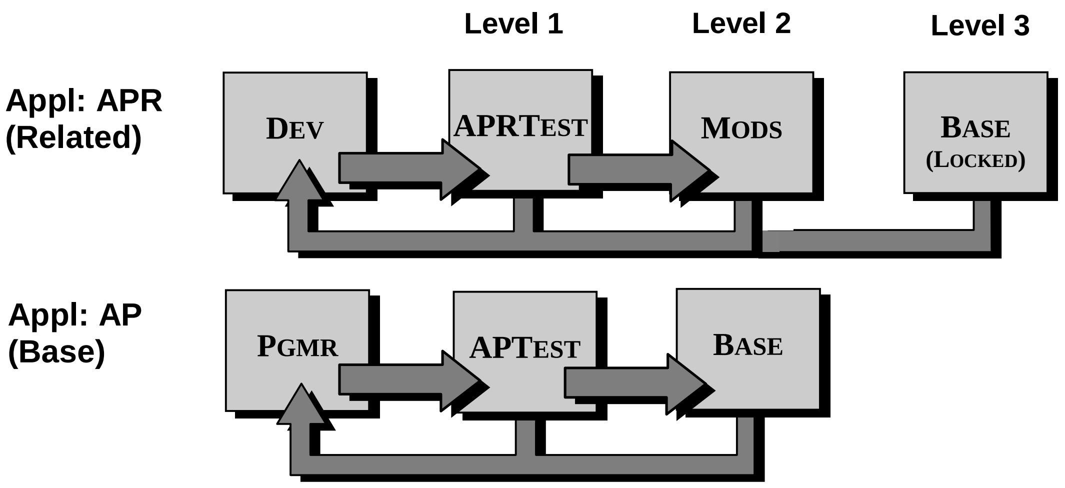


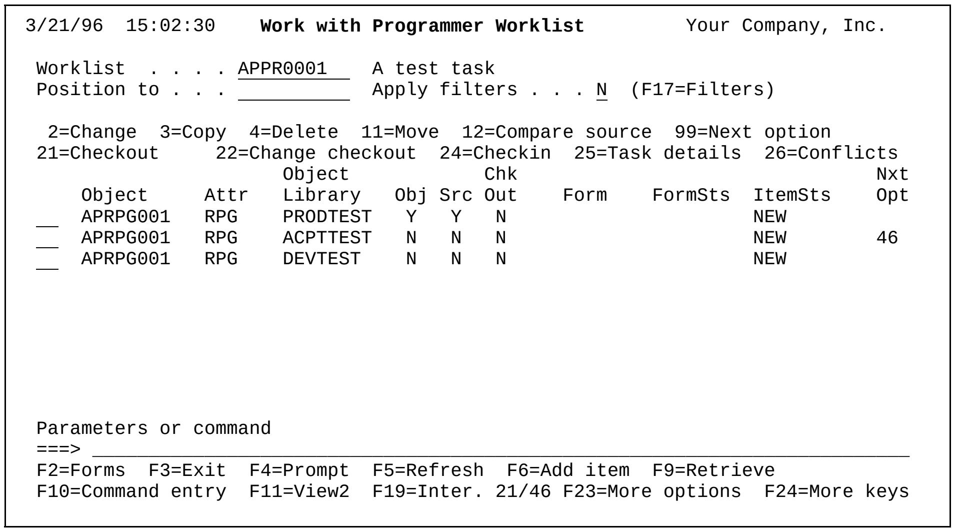
Figure 3

You can check it out for AP and APR. If it is already checked out for APR and has not been promoted to APRTEST, TURNOVER® for iSeries v100 writes a conflict record.

What happens if you ignore a related item and don't check it out? Again, a conflict record is written. You will have to resolve the conflict before you can promote the object to the Base application. (See ***Resolving version conflicts*** on page 13.)

All of this is managed from your ***Programmer's Worklist*** panel.

Here is a sample Programmer's Worklist panel:



```
3/21/96  15:02:30    Work with Programmer Worklist         Your Company, Inc.

 Worklist  . . . . APPR0001    A test task
 Position to . . . __________  Apply filters . . . N  (F17=Filters)

  2=Change  3=Copy  4=Delete  11=Move  12=Compare source  99=Next option
 21=Checkout     22=Change checkout  24=Checkin  25=Task details  26=Conflicts
                      Object            Chk                                Nxt
     Object     Attr   Library   Obj Src Out    Form    FormSts  ItemSts    Opt
 __  APRPG001   RPG    PRODTEST   Y   Y   N                      NEW
 __  APRPG001   RPG    ACPTTEST   N   N   N                      NEW        46
 __  APRPG001   RPG    DEVTEST    N   N   N                      NEW







 Parameters or command
 ===> _________________________________________________________________________
 F2=Forms  F3=Exit  F4=Prompt  F5=Refresh  F6=Add item  F9=Retrieve
 F10=Command entry  F11=View2  F19=Inter. 21/46 F23=More options  F24=More keys
```

Two options on the ***Work with Programmer Worklist*** panel are useful when you're working with related applications:

16=Related objects

Search for related objects in related applications.

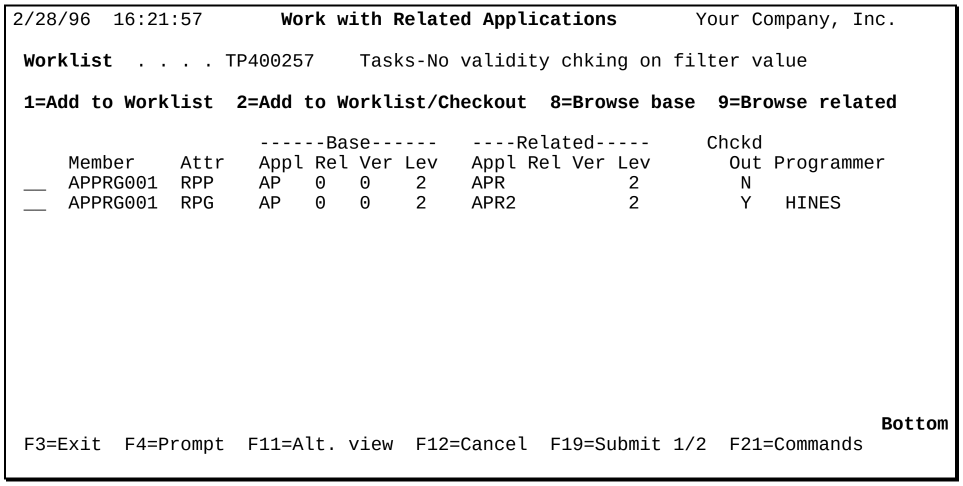
26=Conflicts

Show all conflicts for the object selected.

If there are one or more applications related to the application with which you are working, it is up to you to check for related objects (option **16**). However, if you forget to check, TURNOVER® for iSeries v100 will help you remember. How? TURNOVER® for iSeries v100 sets an indicator on when you add an item to your Worklist and sets it off when you select option **16**. If you forget, TURNOVER® for iSeries v100 will not let you add the item to a TURNOVER® for iSeries v100 form.[4] Cross-referencing (option **15**) works in a similar way.

[4] See **F18=Defaults** and the online Help for how this feature is controlled.

When you select option **16**, you'll either see a message informing you that ***"None of the selected objects have been modified using related applications"*** or you'll see the following panel:



```
2/28/96  16:21:57       Work with Related Applications       Your Company, Inc.

 Worklist  . . . . TP400257    Tasks-No validity chking on filter value

 1=Add to Worklist  2=Add to Worklist/Checkout  8=Browse base  9=Browse related

                     ------Base------   ----Related-----     Chckd
     Member    Attr   Appl Rel Ver Lev   Appl Rel Ver Lev       Out Programmer
 __  APPRG001  RPP    AP   0   0    2    APR           2         N
 __  APPRG001  RPG    AP   0   0    2    APR2          2         Y   HINES











                                                                              Bottom
 F3=Exit  F4=Prompt  F11=Alt. view  F12=Cancel  F19=Submit 1/2  F21=Commands
```

This panel shows you all of the versions of the object(s) you have previously modified within related applications (subsequent releases, foreign language versions, customer modifications, and so on.)

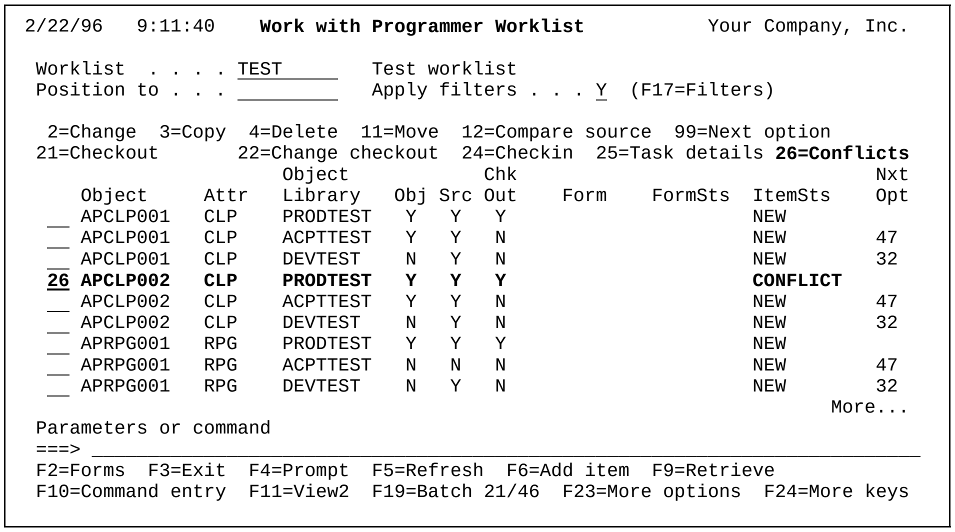
In the panel above, two versions exist, one for related application APR and one for APR2. (Notice that the second one is currently checked out.)

If the programmer checks out the item in the base application on the worklist, but decides not to check out the related object(s), ***TURNOVER® for iSeries v100 will write a <u>conflict record</u> for each of the related applications where the object has been changed***.

If the object in the related application is already checked out, the programmer who checked it out must resolve the conflict before creating a TURNOVER® for iSeries v100 form.

Resolving version conflicts

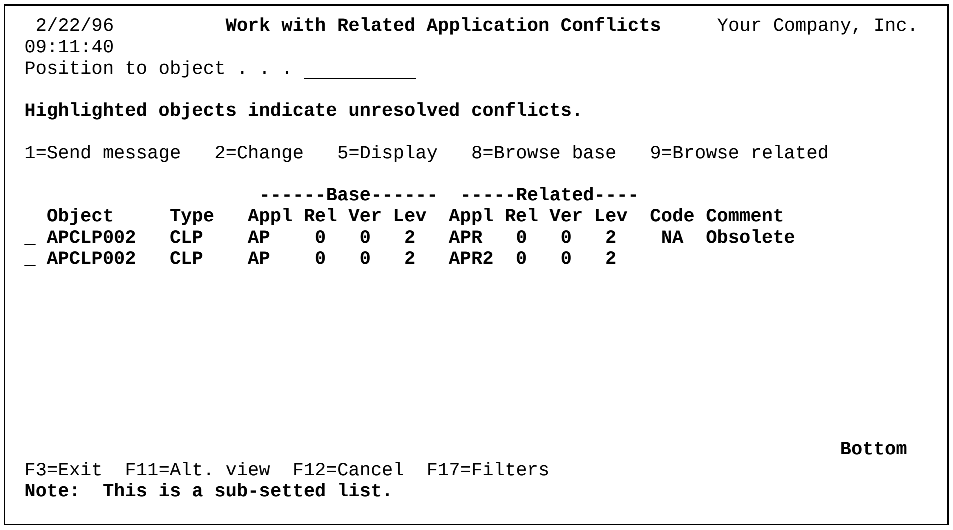
How do you know if a conflict exists? The ***Work with Programmer Worklist*** panel will display the word **CONFLICT** in the ***ItemSts*** field for the item in question. (This does not replace the item status you may have set; it overlays that status on this panel until the conflict condition is resolved.)



```
2/22/96   9:11:40    Work with Programmer Worklist          Your Company, Inc.

 Worklist  . . . . TEST         Test worklist
 Position to . . . _________    Apply filters . . . Y  (F17=Filters)

  2=Change  3=Copy  4=Delete  11=Move  12=Compare source  99=Next option
 21=Checkout       22=Change checkout  24=Checkin  25=Task details 26=Conflicts
                     Object            Chk                                 Nxt
     Object     Attr    Library   Obj Src Out    Form    FormSts  ItemSts    Opt
  __ APCLP001   CLP    PRODTEST   Y   Y   Y                       NEW
  __ APCLP001   CLP    ACPTTEST   Y   Y   N                       NEW        47
  __ APCLP001   CLP    DEVTEST    N   Y   N                       NEW        32
  26 APCLP002   CLP    PRODTEST   Y   Y   Y                       CONFLICT
  __ APCLP002   CLP    ACPTTEST   Y   Y   N                       NEW        47
  __ APCLP002   CLP    DEVTEST    N   Y   N                       NEW        32
  __ APRPG001   RPG    PRODTEST   Y   Y   Y                       NEW
  __ APRPG001   RPG    ACPTTEST   N   N   N                       NEW        47
  __ APRPG001   RPG    DEVTEST    N   Y   N                       NEW        32
                                                                         More...
 Parameters or command
 ===> __________________________________________________________________________
 F2=Forms  F3=Exit  F4=Prompt  F5=Refresh  F6=Add item  F9=Retrieve
 F10=Command entry  F11=View2  F19=Batch 21/46  F23=More options  F24=More keys
```

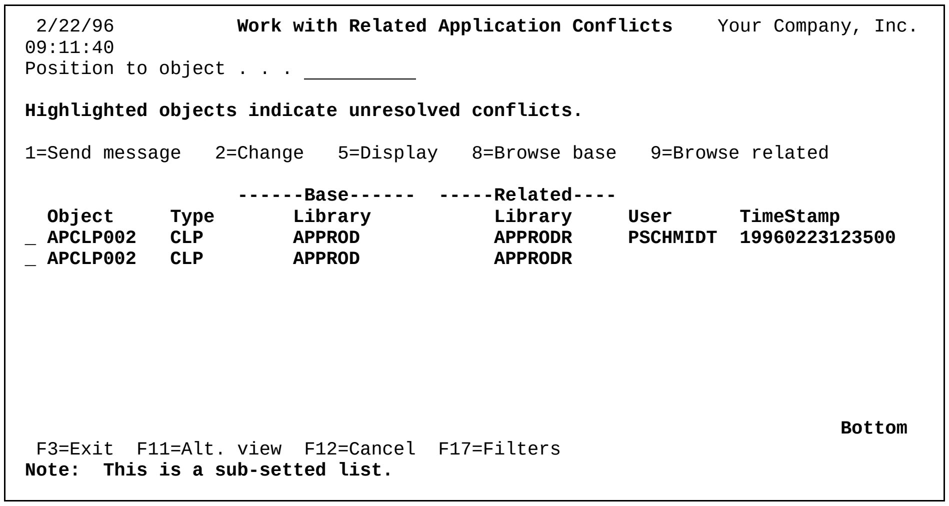
To resolve the conflict, you can either check out the related item or, where you have made the judgment that the related item does not need to be changed, you can update the conflict file. To do so, type **26=Conflicts** next to the item and press **Enter**.

You see the following ***Work with Related Application Conflicts*** panel.



```
 2/22/96         Work with Related Application Conflicts     Your Company, Inc.
09:11:40
Position to object . . . __________

Highlighted objects indicate unresolved conflicts.

1=Send message   2=Change   5=Display   8=Browse base   9=Browse related

                     ------Base------  -----Related----
  Object     Type   Appl Rel Ver Lev  Appl Rel Ver Lev  Code Comment
_ APCLP002   CLP    AP    0   0   2   APR   0   0   2    NA  Obsolete
_ APCLP002   CLP    AP    0   0   2   APR2  0   0   2







                                                                         Bottom
F3=Exit  F11=Alt. view  F12=Cancel  F17=Filters
Note:  This is a sub-setted list.
```

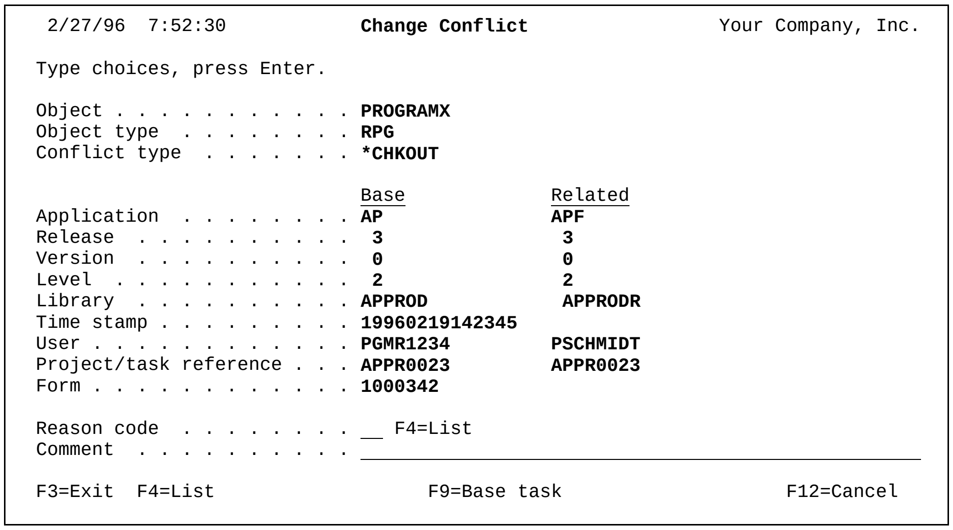
This panel shows you all of the resolved or unresolved entries in the conflicts file for this item. When selected from the worklist, it is subsetted for the item(s) selected. **F11** toggles between alternate views.



```
 2/22/96          Work with Related Application Conflicts    Your Company, Inc.
09:11:40
Position to object . . . __________

Highlighted objects indicate unresolved conflicts.

1=Send message   2=Change   5=Display   8=Browse base   9=Browse related

                   ------Base------  -----Related----
  Object     Type       Library          Library     User      TimeStamp
_ APCLP002   CLP        APPROD           APPRODR     PSCHMIDT  19960223123500
_ APCLP002   CLP        APPROD           APPRODR







                                                                         Bottom
 F3=Exit  F11=Alt. view  F12=Cancel  F17=Filters
Note:  This is a sub-setted list.
```

Select an item with option **1** to send a message to the user associated with the conflict, as in the case where the item is currently checked out to another programmer. Use options **8** and **9** to browse the base source member or the related source member, respectively.

To resolve a conflict for an item, type **2** next to the item. You'll see the ***Change Conflict*** panel.



```
 2/27/96  7:52:30           Change Conflict              Your Company, Inc.

Type choices, press Enter.

Object . . . . . . . . . . . PROGRAMX
Object type  . . . . . . . . RPG
Conflict type  . . . . . . . *CHKOUT

                             Base             Related
Application  . . . . . . . . AP               APF
Release  . . . . . . . . . .  3                3
Version  . . . . . . . . . .  0                0
Level  . . . . . . . . . . .  2                2
Library  . . . . . . . . . . APPROD            APPRODR
Time stamp . . . . . . . . . 19960219142345
User . . . . . . . . . . . . PGMR1234         PSCHMIDT
Project/task reference . . . APPR0023         APPR0023
Form . . . . . . . . . . . . 1000342

Reason code  . . . . . . . . __ F4=List
Comment  . . . . . . . . . . __________________________________________________

F3=Exit  F4=List                  F9=Base task                     F12=Cancel
```

You can resolve a conflict by entering a valid reason code.[5] Type **F4** to list the valid codes. The comment is required and will default to the description of the reason code file, but may be over-typed.

Press **F9** to see the project and task associated with the base changes.

TURNOVER® for iSeries v100 maintains an audit history of all conflict changes made to the Conflict transaction file. You can view and resolve these conflicts using the **TWRKCON** (**Work with Conflicts**) command on a TURNOVER® for iSeries v100 command line and then specifying filters for the list of conflicts you want to generate and review. Also, you can use another command, **TRPTCON** (**Report Conflicts**), at any time, to produce a report of related application conflicts that includes an audit history of conflict resolutions. You can type this command on a command line or run it from a user-written program.

For a list of parameters for both the **TWRKCON** and **TRPTCON** commands, see the TURNOVER® for iSeries v100 Supplement entitled ***TURNOVER® for iSeries v100 Commands*** (**#28**).

UNICOM Systems, Inc. Caution

You should not resolve a conflict if you have not actually resolved the conflict! In other words, if the related program needs to be changed, add it to your worklist, check it out and change it! When you check it out, TURNOVER® for iSeries v100 will automatically update the conflict file for you.

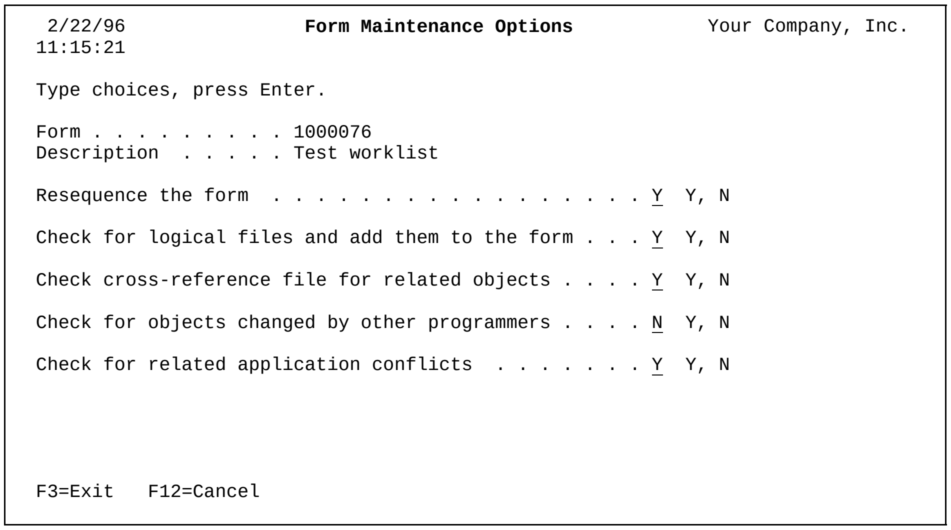
[5] Users with ***Defaults*** authority can maintain the ***Reason code*** file.

Conditions for which conflicts may exist

Condition	Action
You checked out an object from the base application but not from the related application(s).	Add object to your Worklist and check it out.
The item in the related application is checked out to someone else.	Use worklist option **26** to send a message to the other programmer. S/he will have to resolve the conflict manually before the item in the base can be promoted.
You don't have time to checkout and fix the related item at this time.	Resolve with an appropriate Reason code and then permanently resolve the conflict later.
The related item was checked out but then you realized that it did not need to be changed.	Check in the item. The conflict file will be updated to indicate that the item was checked in manually.
The object in the base application has been promoted but the one(s) in the related application(s) has not.	Do nothing. If the item is checked out, it will be resolved automatically when it is promoted.

Building a form with unresolved conflicts

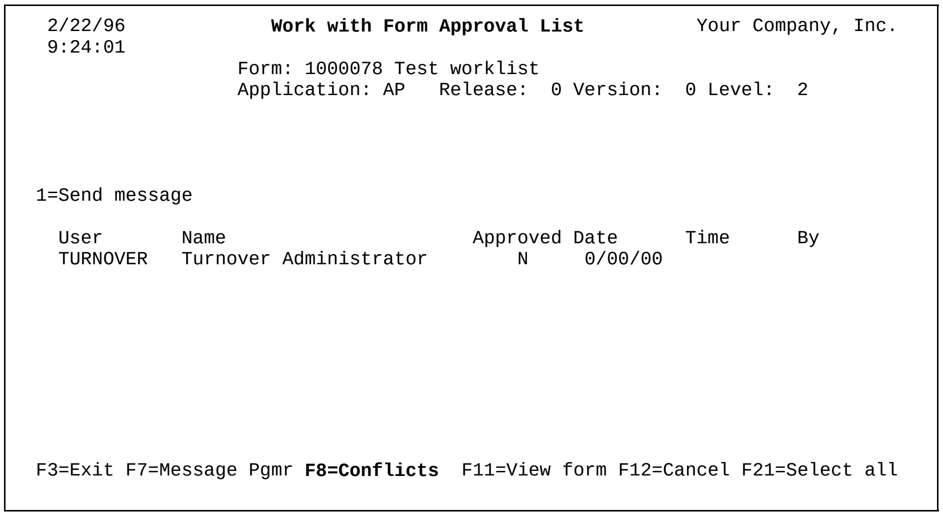
What happens when objects in the related application were created from the source in the base (as would be the case if you are maintaining different national language versions of objects from the same source and different language message files located in related application libraries)? Nothing special needs to be done and there is no conflict. If the ***Programmer Worklist*** session default, ***Related apps check automatic***, is set to **Y**, these objects are automatically included for recompile on TURNOVER® for iSeries v100 forms (See Programmer's Worklist, **F18=Defaults**.) If ***Related apps check automatic*** is set to **N**, you would see the following panel when you build the form from your Worklist (option **46**) or if you edit the form (option **42**). In either case, you would see the following panel:


```
 2/22/96               Form Maintenance Options           Your Company, Inc.
11:15:21

Type choices, press Enter.

Form . . . . . . . . . 1000076
Description  . . . . . Test worklist

Resequence the form  . . . . . . . . . . . . . . . . . Y  Y, N

Check for logical files and add them to the form . . . Y  Y, N

Check cross-reference file for related objects . . . . Y  Y, N

Check for objects changed by other programmers . . . . N  Y, N

Check for related application conflicts  . . . . . . . Y  Y, N




F3=Exit   F12=Cancel
```

When you press **Enter**, TURNOVER® for iSeries v100 displays the ***Work with Conflict*** panel (shown on page 14) if conflicts exist for items in applications related to those on the TURNOVER® for iSeries v100 form. If no conflicts exist, you'll see the message ***"No conflicts found for form nnnnnnn"***.

Approving forms with version conflicts

When you approve a form (option **48** on the ***Programmer's Worklist*** or option **10** on the ***Work with Forms*** panel) for which related applications exist, you'll see **F8=Conflicts** on the panel, as illustrated below:

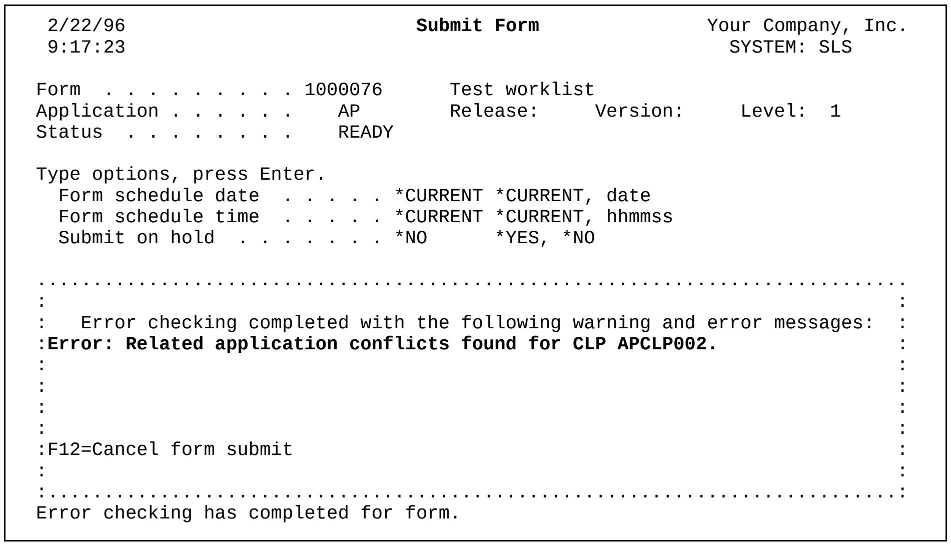


```
 2/22/96            Work with Form Approval List          Your Company, Inc.
 9:24:01
                 Form: 1000078 Test worklist
                 Application: AP   Release:  0 Version:  0 Level:  2



1=Send message

  User       Name                       Approved Date      Time      By
  TURNOVER   Turnover Administrator         N     0/00/00







F3=Exit F7=Message Pgmr F8=Conflicts  F11=View form F12=Cancel F21=Select all
```

If you press **F8**, you'll see the ***Work with Conflicts*** panel, filtered by the objects on the TURNOVER® for iSeries v100 form.

Submitting a form with unresolved conflicts

What happens if you try to submit a form with unresolved conflicts? You'll see the following panel:



```
 2/22/96                          Submit Form              Your Company, Inc.
 9:17:23                                                     SYSTEM: SLS

Form  . . . . . . . . . 1000076      Test worklist
Application . . . . . .    AP        Release:     Version:     Level:  1
Status  . . . . . . . .    READY

Type options, press Enter.
  Form schedule date  . . . . . *CURRENT *CURRENT, date
  Form schedule time  . . . . . *CURRENT *CURRENT, hhmmss
  Submit on hold  . . . . . . . *NO      *YES, *NO

..............................................................................
:                                                                            :
:   Error checking completed with the following warning and error messages:  :
:Error: Related application conflicts found for CLP APCLP002.                :
:                                                                            :
:                                                                            :
:                                                                            :
:                                                                            :
:F12=Cancel form submit                                                      :
:                                                                            :
:............................................................................:
Error checking has completed for form.
```

To resolve the conflict, cancel the submit request and resolve the conflict. Remember, conflicts are recorded only if an object of the same name as the one you changed was found in a related application, not for every object in the base application. If there are **Error** messages, the form will not be run.

Advanced related applications

SETUP examples

This section provides examples of the various ways in which related applications can be applied and recommendations for their use. As mentioned earlier, related applications can be used for:

- Managing customized versions of software
- Managing multiple releases of software
- Managing national language versions of software
- Various combinations of all of these requirements.

Managing customized versions of software

Perhaps you're a software vendor and need to maintain different customized changes for each of your customers; or you are the development manager responsible for maintaining customized changes for several different plant locations, using the same software. You might define and relate your applications like this:

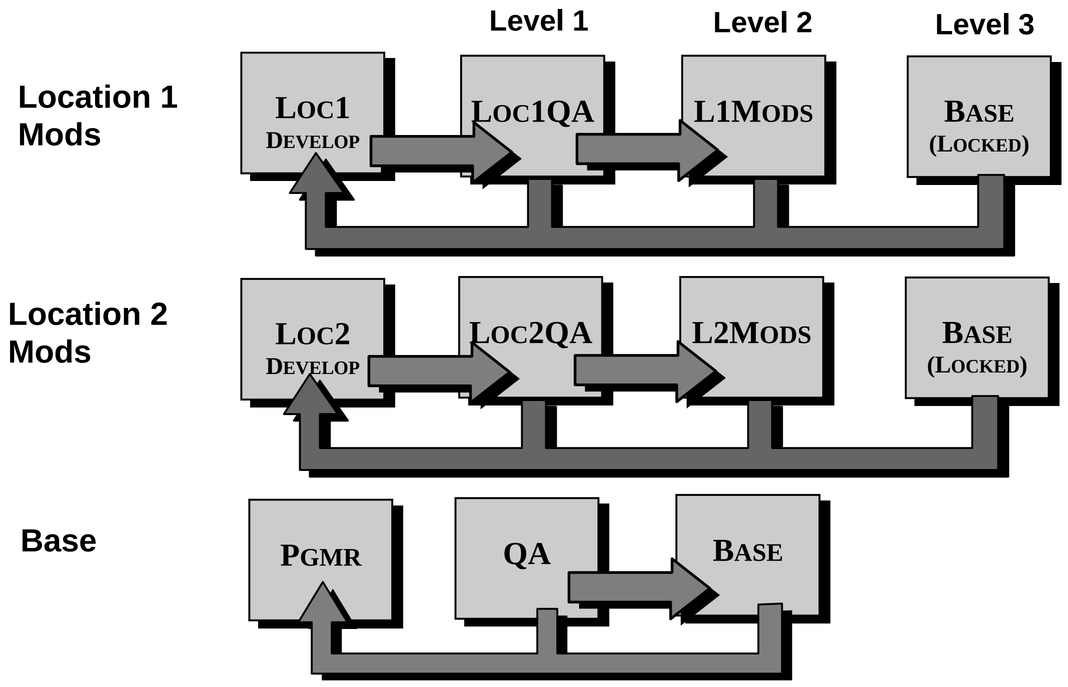


Figure 4

As with the examples described earlier in this document, when you check out an item for the Base application, you would need to know if any of the items you are checking out were customized for either location; in other words, have the objects been modified for either location 1 or 2? See page 4 for the rules that govern related applications.

Although we have only shown you two related applications in this example, there is really no limit to the number of related applications you can define. In fact, you may have several hundred sets of custom modifications. The only practical limitation is governed by disk space.

Managing multiple language versions of software

The most efficient method of creating different language versions of a software package is to put the national language text into message files and then translate the message file text. On the iSeries, display files can be defined in such a way as to allow the text to be loaded at run time. Of course, status and error messages are delivered to the user in the same way. Commands and printer files have embedded text strings, but these too can have externalized language text. However, they must be compiled over the correct national language message file. No matter how many language versions of software you have, you only need one copy of source. This is the way TURNOVER® for iSeries v100's different language versions are managed. Therefore, to maintain different language versions of your software which are, in every other respect, identical, you would probably want to set up your applications as illustrated here:

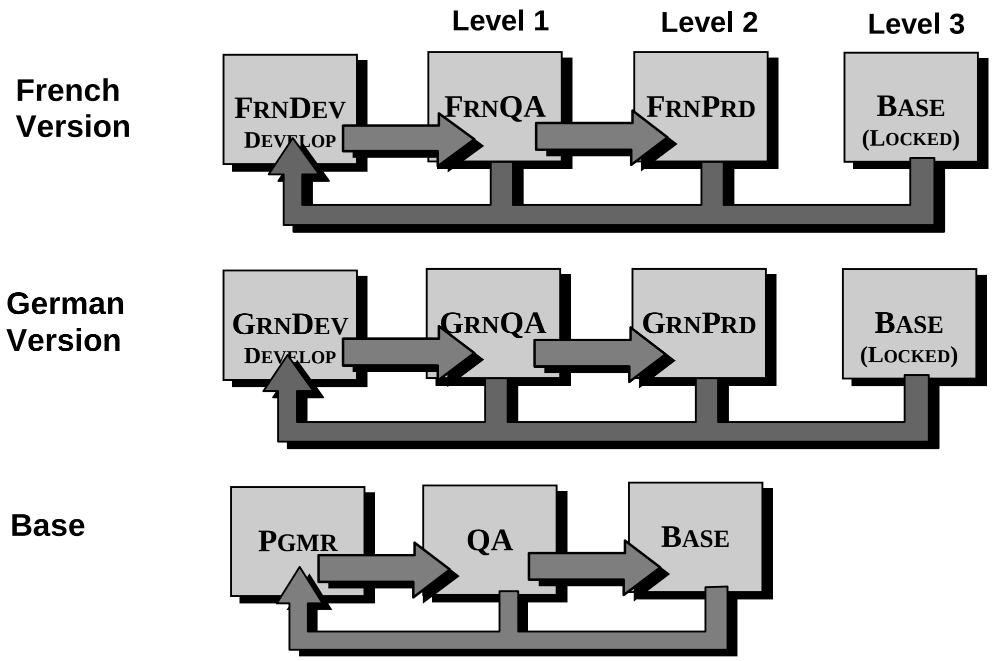


Figure 5

In these applications, the GRNPRD and FRNPRD libraries contain a message file and just those objects that need to be compiled over it, such as commands and printer files. Other than maintaining the translated versions of the message file, which TURNOVER® for iSeries v100 fully supports, no changes need to be made directly to these objects. Changes only need to be made to the source in the base application; in this example, no source exists in the language libraries. When you build a Base application form to promote any language-sensitive objects from, say, QA to BASE, TURNOVER® for iSeries v100 will create TURNOVER® for iSeries v100 forms for each of the related language applications. When these run, TURNOVER® for iSeries v100 will create the objects in the appropriate language library for each related application using the source from the base application compiled over the appropriate language message file. Of course, your application has to be clever enough to manage the substitution of the correct message file and language library when it is running. (See the ***TURNOVER® for iSeries v100 Application Planning Guide***.)

You will see how TURNOVER® for iSeries v100 manages multiple language versions of several releases on page 24.

Managing multiple releases of software

Although similar to the example we showed you on page 3, we've slipped in a few variations as food for thought. In this case, you're managing three releases and you need to support all of these releases, simultaneously.

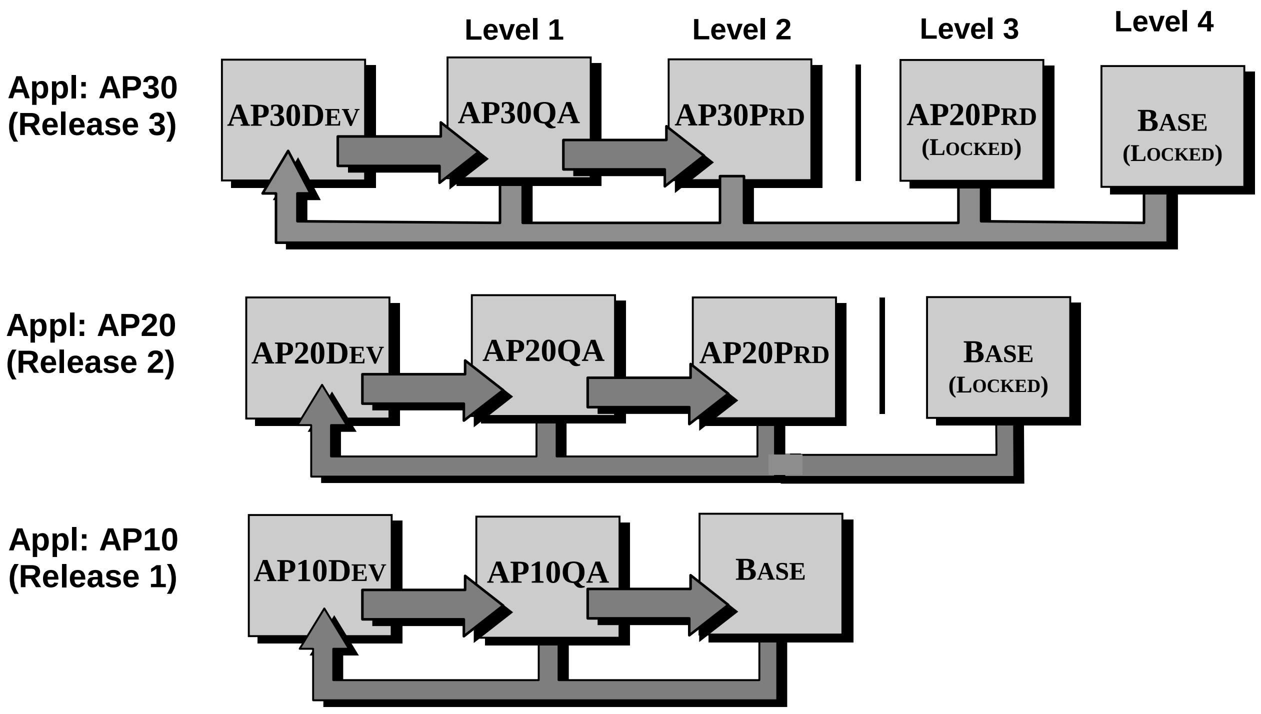


Figure 6

In this example, Release 1 is the Base for Release 2 and Release 2 is the base for Release 3! This is called a nested related application. If you detect an error (or one of your users did), it is likely that you would want to check out and fix all copies of the program affected, unless, of course, the program affected wasn't added to the application until a later release or had been rewritten along the way. So, the first rule for managing multiple releases is always to start by identifying the object(s) you need from the oldest release. Add these to your Programmer's worklist, and then select option **16** to check for related objects. If you need to fix these, add them to your Worklist. Then check them all out, using option **21**.

Once you have checked them out (into different libraries, of course), you need to change and test each version. Then promote them all back.

Now here are a few twists. What if Programmer A is modifying a program for Release 3 and you, Programmer B, need to check out the same object for Release 1, 2, and 3?

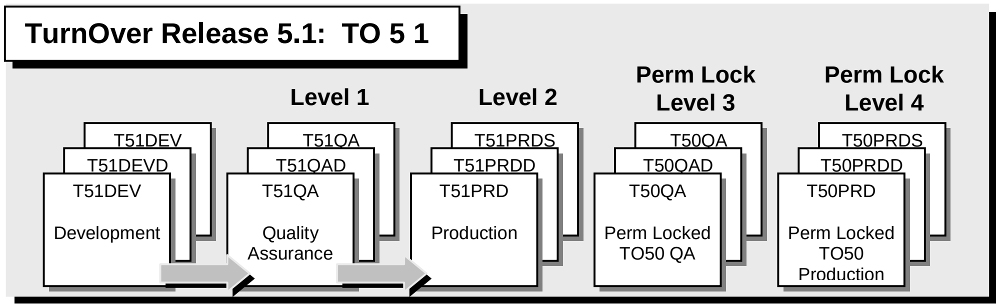
- You can wait until programmer A promotes the object back to production (AP30PRD in our diagram).
- You could go ahead and check out the object for Release 1 and 2 and put the third one on your Worklist, with the intention of changing it later.
- You could message programmer A and tell him to fix the Release 3 version of the object.
- You could forget it and not fix the Release 3 version of the object – no, sorry, TURNOVER® for iSeries v100 won't let you do this! **You must resolve the conflict before promoting the Release 1 and 2 versions**.

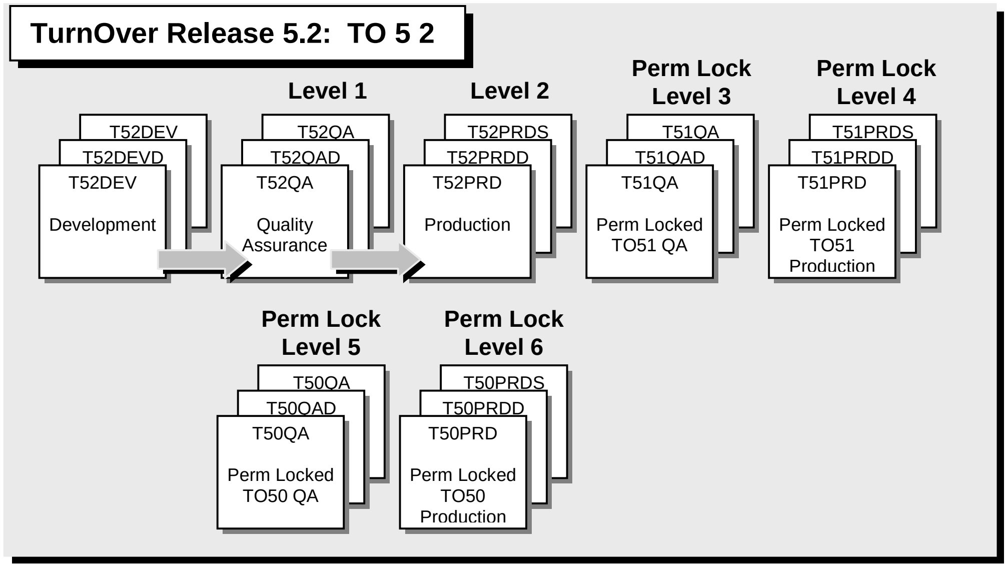
In the same scenario, what if Programmer A had already promoted the object to the test level (AP30QA). It is still checked out of AP30PRD, so a conflict record is written.

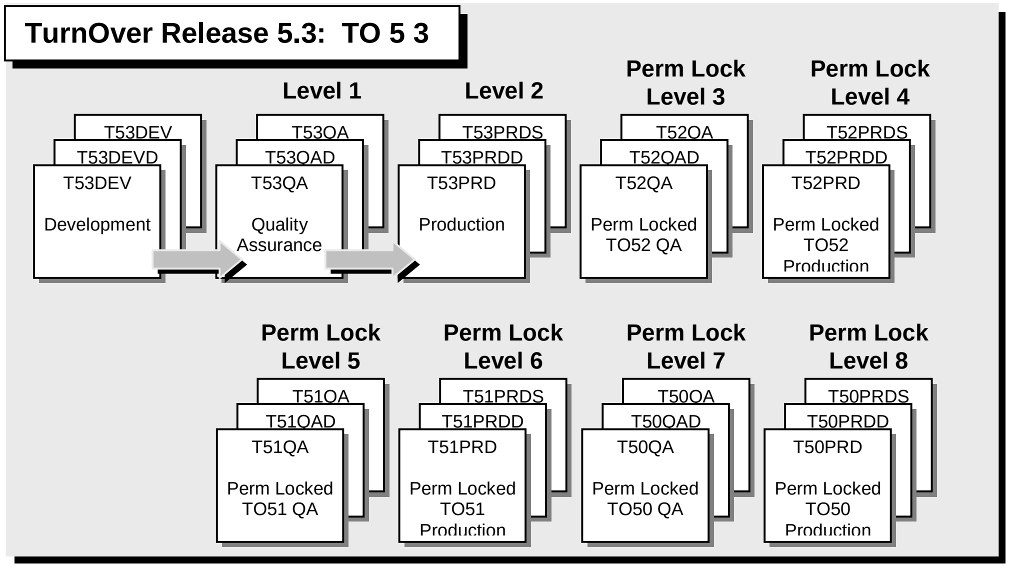
Combining multiple releases with language or versions (or both)

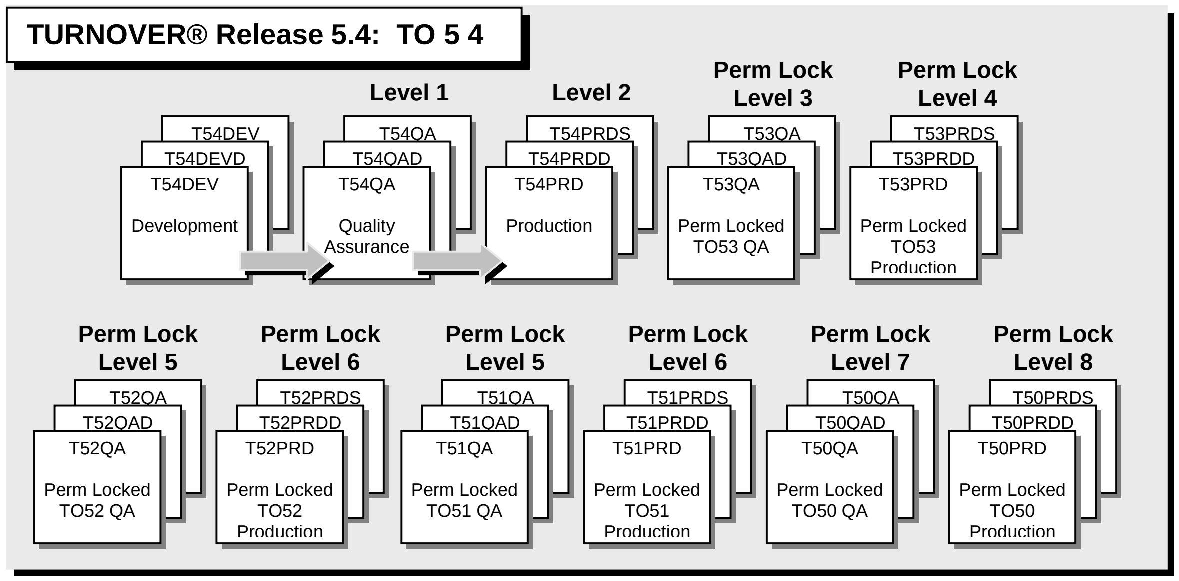
The examples on the next two pages are taken directly from our own setup, and reflect our experience managing multiple releases of the TURNOVER® for iSeries v100 product.

TURNOVER® for iSeries v100 application definition schematics (examples)








TURNOVER® Release 5.4: TO 5 4
Level 1
Level 2
Perm Lock Level 3
Perm Lock Level 4
T54DEV
T54DEVD
T54DEV Development
T54QA
T54QAD
T54QA Quality Assurance
T54PRDS
T54PRDD
T54PRD Production
T53QA
T53QAD
T53QA Perm Locked TO53 QA
T53PRDS
T53PRDD
T53PRD Perm Locked TO53 Production
Perm Lock Level 5
Perm Lock Level 6
Perm Lock Level 5
Perm Lock Level 6
Perm Lock Level 7
Perm Lock Level 8
T52QA
T52QAD
T52QA Perm Locked TO52 QA
T52PRDS
T52PRDD
T52PRD Perm Locked TO52 Production
T51QA
T51QAD
T51QA Perm Locked TO51 QA
T51PRDS
T51PRDD
T51PRD Perm Locked TO51 Production
T50QA
T50QAD
T50QA Perm Locked TO50 QA
T50PRDS
T50PRDD
T50PRD Perm Locked TO50 Production

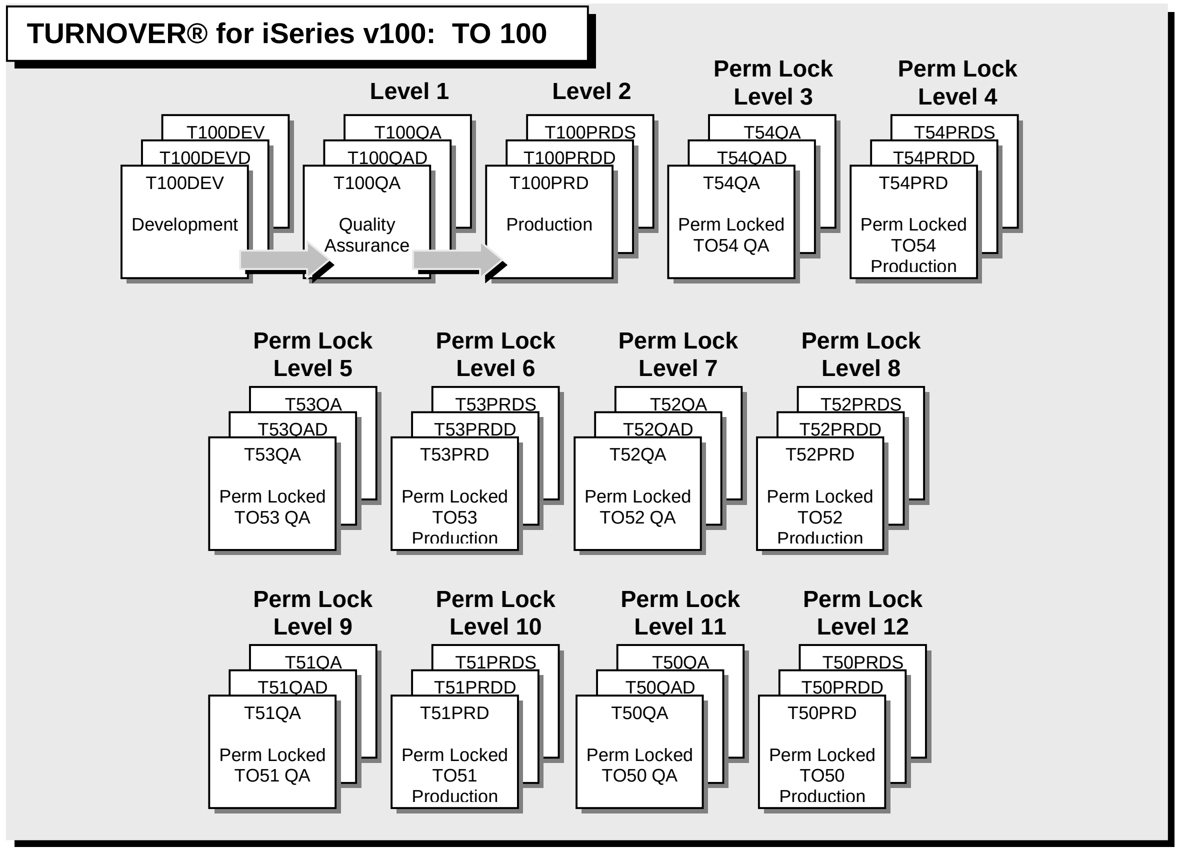

TURNOVER® for iSeries v100: TO 100
Level 1
Level 2
Perm Lock Level 3
Perm Lock Level 4
T100DEV
T100DEVD
T100DEV Development
T100QA
T100QAD
T100QA Quality Assurance
T100PRDS
T100PRDD
T100PRD Production
T54QA
T54QAD
T54QA Perm Locked TO54 QA
T54PRDS
T54PRDD
T54PRD Perm Locked TO54 Production
Perm Lock Level 5
Perm Lock Level 6
Perm Lock Level 7
Perm Lock Level 8
T53QA
T53QAD
T53QA Perm Locked TO53 QA
T53PRDS
T53PRDD
T53PRD Perm Locked TO53 Production
T52QA
T52QAD
T52QA Perm Locked TO52 QA
T52PRDS
T52PRDD
T52PRD Perm Locked TO52 Production
Perm Lock Level 9
Perm Lock Level 10
Perm Lock Level 11
Perm Lock Level 12
T51QA
T51QAD
T51QA Perm Locked TO51 QA
T51PRDS
T51PRDD
T51PRD Perm Locked TO51 Production
T50QA
T50QAD
T50QA Perm Locked TO50 QA
T50PRDS
T50PRDD
T50PRD Perm Locked TO50 Production

Figure 7

Notes about Figure 7:

The oldest supported version, Release 5.1 (previous page), is the base for Release 5.2; Release 5.2 is the base for 5.3; Release 5.3 is the base for 5.4 and so on.

At UNICOM Systems, Inc., all of the objects that exist in the base release also exist in subsequent releases (unless they were intentionally removed from the product). However, source code exists in subsequent releases only if the object was changed in that release. If you check out a source member for Release 5.1, TURNOVER® shows you the related source in subsequent releases only if it exists there (that is, if it was changed for both releases). If an object is changed for Release 5.1 but not in subsequent releases, then the object is automatically recompiled into subsequent Releases (5.2 and 5.3) from the source in Release 5.1.

If a new object is created in Release 5.1, TURNOVER® automatically creates the object in subsequent releases, depending on the new related object rule setting in the related application file (TAPPRELF). You can set the object rule for the child relationships within an application on the ***Work with Application Relationships*** panel. Also, you can use the **New Related Object** (**TNEWRELOBJ**) command to change the rule for any related application. See the command online Help for more information.

The possible settings for the new related object rule are as follows:

1= Do not create new objects in subsequent releases.

2= Create data objects in subsequent releases.

3= Create all objects in subsequent releases.

At UNICOM Systems, Inc., as with most companies managing multiple releases, this rule is set to 3.

To explore a particular setup in more depth – whether your setup, or ours – call our Technical Support line.

OBJECT VERSIONING

As described earlier, application versioning and object versioning are different, but similar. ***Application versioning*** means using TURNOVER® for iSeries v100's related application feature along with the native iSeries library management capabilities to manage a set of related objects. ***Object versioning*** is intended to:

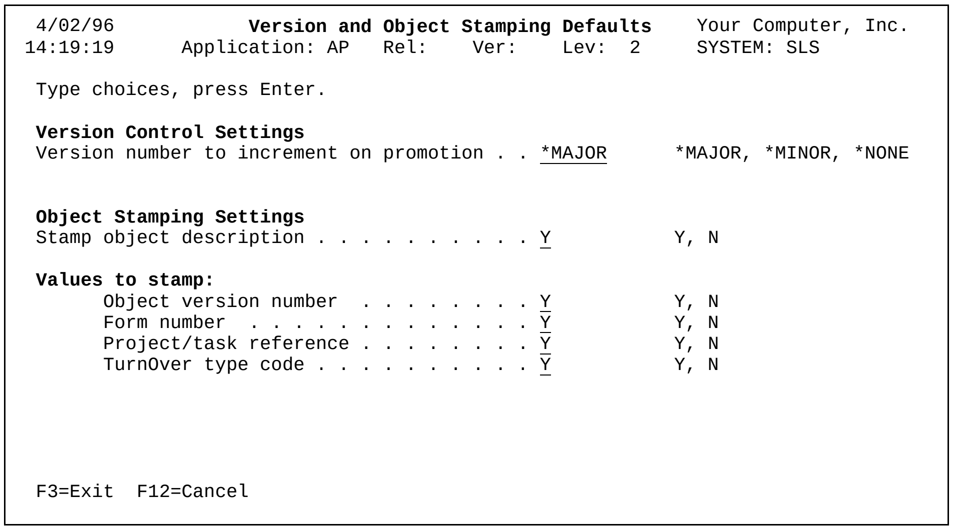
- Help you quickly identify the version of a specific object, by distinguishing it from its predecessors and successors;
- Quickly tell what version of an object a user is using,
- Tell exactly how an object was changed, who changed it, why it was changed, and so on, every time it was changed, throughout the object's life.

The iSeries is particularly well-suited for managing and auditing changes because of the information its operating system stores in its object description – information that is protected from unauthorized manipulation – and also because TURNOVER® for iSeries v100 augments this descriptive data with information of its own stored in fields provided for this purpose.

This section describes exactly how TURNOVER® for iSeries v100 tracks object versions, what information is stored in the object description, and how you can access this information.

TURNOVER® for iSeries v100 maintains a version number for all objects you maintain using TURNOVER® for iSeries v100. This information is tracked within TURNOVER® for iSeries v100's database, regardless of whether or not you decide to stamp the object's description.

You can use the following application definition panel to specify how TURNOVER® for iSeries v100 should control versioning and object stamping for a given application and level.



```
 4/02/96           Version and Object Stamping Defaults    Your Computer, Inc.
14:19:19      Application: AP   Rel:    Ver:    Lev:  2     SYSTEM: SLS

 Type choices, press Enter.

 Version Control Settings
 Version number to increment on promotion . . *MAJOR      *MAJOR, *MINOR, *NONE


 Object Stamping Settings
 Stamp object description . . . . . . . . . . Y           Y, N

 Values to stamp:
       Object version number  . . . . . . . . Y           Y, N
       Form number  . . . . . . . . . . . . . Y           Y, N
       Project/task reference . . . . . . . . Y           Y, N
       TurnOver type code . . . . . . . . . . Y           Y, N




 F3=Exit  F12=Cancel
```

What you see:

Version Control Settings

Every object that is changed by TURNOVER® for iSeries v100 will have a major and minor version number associated with it expressed in the form of "*xxx.yyy*", such as '17.2', where *xxx* is the major version number, and *yyy* is the minor version number. When an object is promoted for a given level, you can specify whether to increment the major or minor version number. Typically, you would only increment the major version number when the object is promoted to the highest unlocked level (production), and you would increment the minor version number when the object is promoted to lower levels. When the major version is incremented, the minor version is automatically reset to zero. You would then be able to quickly tell if the object is a test or production object.

Object Stamping Settings

If you set the **Stamp object description** parameter, and one or more parameters (**Object version number**, **Form number**, **Project/task reference**, and **TURNOVER® for iSeries v100 type code**) described below, to **Y**, TURNOVER® for iSeries v100 updates the object description with the corresponding information about the object. If you set the **Stamp object description** parameter to **N,** the following four parameters (**Object version number**, **Form number**, **Project/task reference**, and **TURNOVER® for iSeries v100 type code**) do not appear.

Note: If you plan to distribute objects for an application definition that uses **MO** (Move Object) or **CD** (Create Duplicate) as its promotion method, you should make sure that object stamping is turned off in the remote application definition.

Note: TURNOVER® for iSeries v100 information is mapped into object description fields wherever space is available – fields not necessarily descriptive of the data stored in them. TURNOVER® for iSeries v100's **TDSPOBJD** command, described on page 31, displays this information clearly.

Object version number

If set to **Y**, TURNOVER® for iSeries v100 stores the *xxx.yyy* version number of the object in the licensed program portion of the object description. To distinguish between changes made to the same object in different releases of the same application, the TURNOVER® Release and Version of the application used to promote the object would also be stored in this field. The application code associated with the object will be stored in the first four characters of the APAR ID field in the object description. This information, along with the release and version, is useful when you have multiple applications with the same release and version number, such as when your application needs to support multiple languages.

Form number

If set to **Y**, TURNOVER® for iSeries v100 stores the form number of the form that created the object in the PTF number field of the object description.

Project/task reference

If set to **Y**, TURNOVER® for iSeries v100 stores the first eight characters for the project/task reference of the last change to this object in the Object control level field of the object description. The last two characters are stored in the last two characters of the APAR field of the object description.

TURNOVER® for iSeries v100 type code

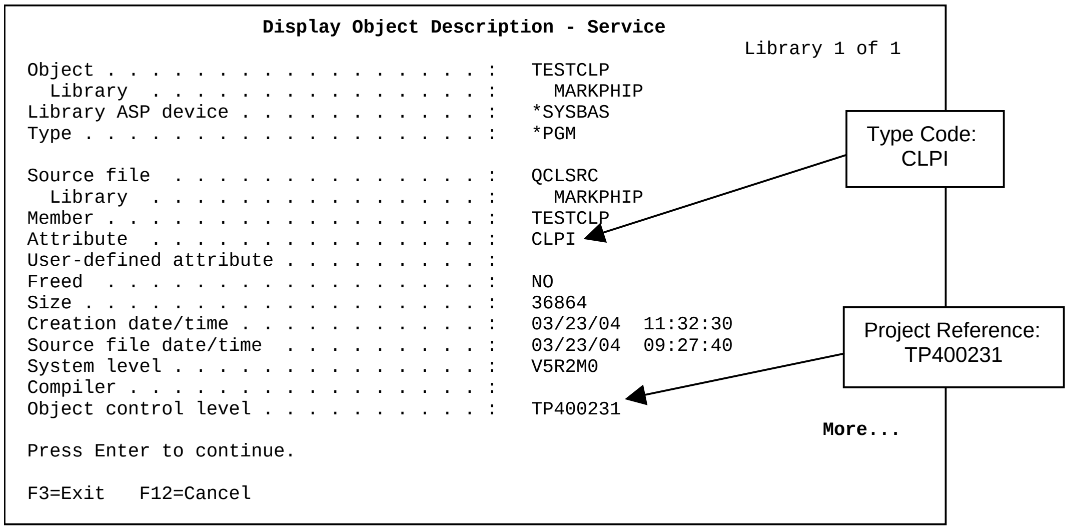
If set to **Y**, TURNOVER® for iSeries v100 stores the TURNOVER® for iSeries v100 type code used the last time the object was changed in the user-defined attribute field of the object description. TURNOVER® for iSeries v100 checkout programs retrieve this type code when checking out an object by retrieving it from this field in the object description, if available.

Excluding objects from stamping

In some cases, you might need to exclude certain objects from being stamped in the object description. The **TCODOMTF** (**Change Object Description Omit**) file, which is keyed by object/type/library, has a 1-character field for each of the values (object version number, form number, project/task reference, and type code) that you can stamp in the object description. You can exclude any or all of these values from stamping by adding them the **TCODOMTF** file and placing an **X** in the appropriate field. You can specify the library or use ***ALL** to exclude object stamping for all libraries.

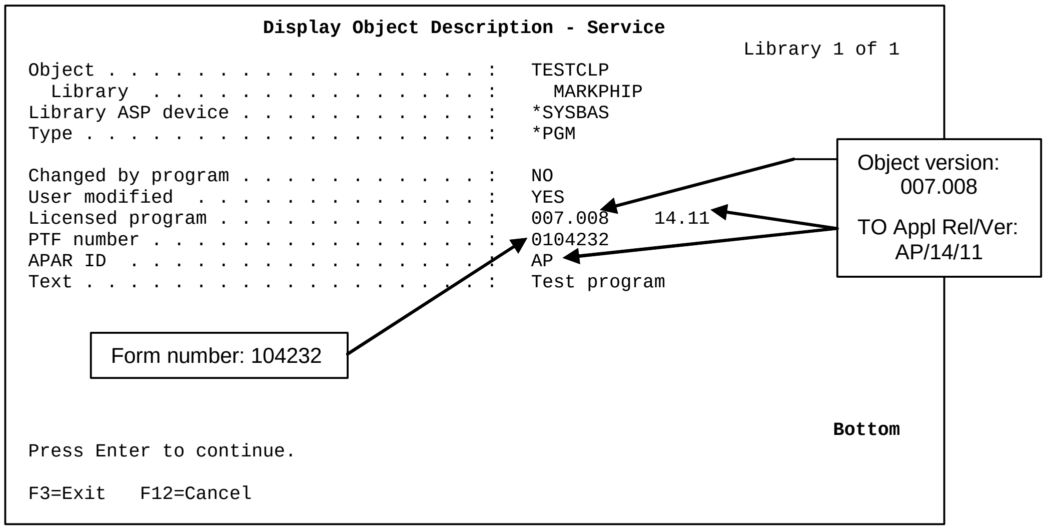
Sample display of object description

Here is what you see when you run the **DSPOBJD** command (***SERVICE**) for an object after it was promoted by TURNOVER® for iSeries v100. TURNOVER® for iSeries v100's **Display Object description TDSPOBJD** command, described on page 31, formats this information.



```
                      Display Object Description - Service
                                                            Library 1 of 1
Object . . . . . . . . . . . . . . . . . :   TESTCLP
  Library  . . . . . . . . . . . . . . . :     MARKPHIP
Library ASP device . . . . . . . . . . . :   *SYSBAS
Type . . . . . . . . . . . . . . . . . . :   *PGM

Source file  . . . . . . . . . . . . . . :   QCLSRC
  Library  . . . . . . . . . . . . . . . :     MARKPHIP
Member . . . . . . . . . . . . . . . . . :   TESTCLP
Attribute  . . . . . . . . . . . . . . . :   CLPI
User-defined attribute . . . . . . . . . :
Freed  . . . . . . . . . . . . . . . . . :   NO
Size . . . . . . . . . . . . . . . . . . :   36864
Creation date/time . . . . . . . . . . . :   03/23/04  11:32:30
Source file date/time  . . . . . . . . . :   03/23/04  09:27:40
System level . . . . . . . . . . . . . . :   V5R2M0
Compiler . . . . . . . . . . . . . . . . :
Object control level . . . . . . . . . . :   TP400231
                                                                   More...
Press Enter to continue.

F3=Exit   F12=Cancel
```

Page down…



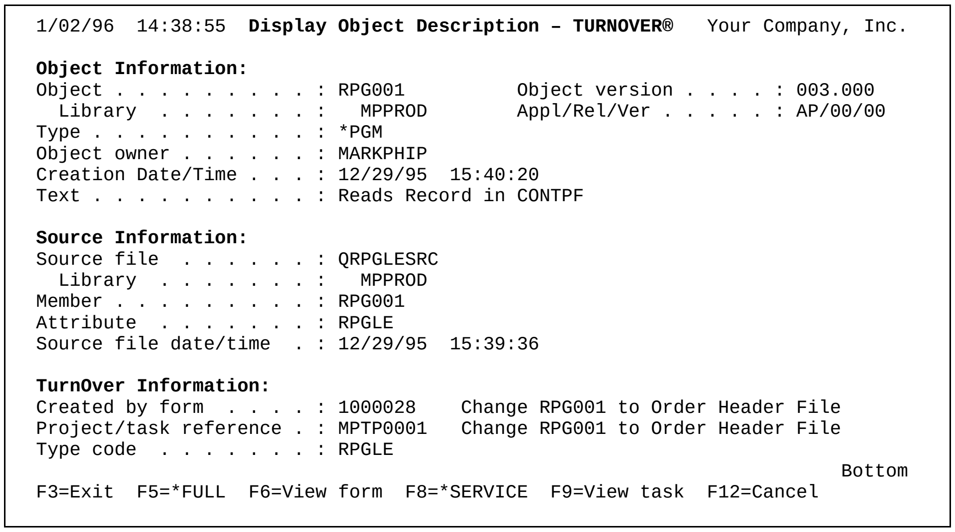
```
                      Display Object Description - Service
                                                            Library 1 of 1
Object . . . . . . . . . . . . . . . . . :   TESTCLP
  Library  . . . . . . . . . . . . . . . :     MARKPHIP
Library ASP device . . . . . . . . . . . :   *SYSBAS
Type . . . . . . . . . . . . . . . . . . :   *PGM

Changed by program . . . . . . . . . . . :   NO
User modified  . . . . . . . . . . . . . :   YES
Licensed program . . . . . . . . . . . . :   007.008    14.11
PTF number . . . . . . . . . . . . . . . :   0104232
APAR ID  . . . . . . . . . . . . . . . . :   AP
Text . . . . . . . . . . . . . . . . . . :   Test program




                                                                    Bottom
Press Enter to continue.

F3=Exit   F12=Cancel
```

Object version utilities

We have provided several utility programs to help you make greater use of the information stored in the object description, and to easily set up your initial object descriptions.

TURNOVER® for iSeries v100's Display Object Description (TDSPOBJD) command

TURNOVER® for iSeries v100's **Display object description** (**TDSPOBJD**) command can be duplicated into a library in the system part of your library list and used as an enhanced version of the **DSPOBJD** command, if you want. This command recognizes those fields in the object description that TURNOVER® for iSeries v100 updates. Here's an example of the command panel:



```
1/02/96  14:38:55  Display Object Description – TURNOVER®   Your Company, Inc.

Object Information:
Object . . . . . . . . . : RPG001         Object version . . . . : 003.000
  Library  . . . . . . . :   MPPROD       Appl/Rel/Ver . . . . . : AP/00/00
Type . . . . . . . . . . : *PGM
Object owner . . . . . . : MARKPHIP
Creation Date/Time . . . : 12/29/95  15:40:20
Text . . . . . . . . . . : Reads Record in CONTPF

Source Information:
Source file  . . . . . . : QRPGLESRC
  Library  . . . . . . . :   MPPROD
Member . . . . . . . . . : RPG001
Attribute  . . . . . . . : RPGLE
Source file date/time  . : 12/29/95  15:39:36

TurnOver Information:
Created by form  . . . . : 1000028    Change RPG001 to Order Header File
Project/task reference . : MPTP0001   Change RPG001 to Order Header File
Type code  . . . . . . . : RPGLE
                                                                     Bottom
F3=Exit  F5=*FULL  F6=View form  F8=*SERVICE  F9=View task  F12=Cancel
```

This panel provides the same information you would see if you were to use the iSeries display object description command, but it also lets you view TURNOVER® for iSeries v100's own audit information. Another nice feature is you can view the TURNOVER® for iSeries v100 form or project task record for an object by pressing **F6** and **F9**, respectively. This command also displays the object's version number in an easy-to-understand format. If you need to access any of the other information in the object description, there are function keys that run the OS/400 **DSPOBJD** command.

Viewing object version on Work with Object History

Option **11** (Line Details) on the ***Work with Object History*** panel displays the version numbers of the object you selected. Besides making it easy to track the version history for a given object, this is also useful for displaying the version number for non-OS/400 object types (which cannot be stamped as the native iSeries objects can), such as copybook code, MSG ID's, *DATA files, and so on.

TURNOVER® for iSeries v100's Set Application Object (TSETAPPOBJ) command

TURNOVER® for iSeries v100's **TSETAPPOBJ** (**Set Application Object**) command sets the initial version number for all existing objects in an application to a value you specify. TURNOVER® for iSeries v100 stores an object's version information in the licensed program portion of the object description. (To see how the object version information appears in the object description, see the ***Sample display of object description*** section on page 30.)

The **TSETAPPOBJ** command also has two other useful features:

If you have been using TURNOVER® for iSeries v100 prior to these new capabilities, **TSETAPPOBJ** will retrieve TURNOVER® for iSeries v100's object history database to determine the last form number, project reference and type code for an object, and update those values in the object description.

TSETAPPOBJ also provides an option to process just those objects for an application that have not already been processed. This can be useful if you add a new library to an existing application, and you just want to stamp the new objects.

TURNOVER® for iSeries v100's Set Object Version (TSETOBJVER) command

This command is functionally identical to the **TSETAPPOBJ** command, except it processes objects by name, library and type. This makes it useful for processing subsets of objects within an application. The **TSETAPPOBJ** command uses the **TSETOBJVER** command after first determining the libraries and objects within your application from the TURNOVER® for iSeries v100 database.

Setting object values during the initial build

The TURNOVER® for iSeries v100 initial build process sets the version number to 1.0 for all objects not already defined to TURNOVER® for iSeries v100, and updates the object description based on the application settings. If you currently use one of the user fields in the object description, be sure to define your application (see page 8) such that you avoid overlaying the field(s) you use.

Additional object version information

Logic used to determine version number

This section is informational; use it only if you have a question about the version number settings for a particular object.

1) Is Version field in line record filled in?

- Yes. Use this as base for incrementing version number based on application. This will only be the case in two scenarios:

  Distribution. This field will be filled in with the version number that was set on the development system. Presumably, you will choose to increment with ***NONE** on the remote system to keep the version numbers the same on all systems.

  Set by exit program. A user may choose to preset this value with an exit program which runs before the version is set, such as the pre-run or add line exit.

- No. Continue.

2) If Version Increment Value is *MINOR or *NONE

Does version record exist for the object in the from library?

- Yes. Use this as the base for incrementing the version number based on application. How do we determine the "from object"?

  Line record. If the fields on the line are filled in, they will be used.

  Application. If the fields are empty, we will get the values from the application type code.

- No. Find the next level of the application, starting with the target level, that the object is checked out from.

Does version record exist for the object in the checkout library?

- Yes. Use this as base for incrementing version number based on application.

- No. Starting at the level of the application where the checkout record was found, search up the application until you find a version record for the object, and use it as the base for incrementing the version number based on the application. If no record is found, then this is a new object and should be incremented from zero. This extra bit of logic is to find objects that were checked out from perm-locked levels and the object does not exist yet in the highest unlocked level against which the checkout record will have been written.

3) If Version Increment Value is *MAJOR

If the version increment value is ***MAJOR**, we must determine the highest current major version number in the application and use it as the value we increment. To do this, we start at the lowest level of the application and work our way up the entire application structure, recording the highest major version level we find along the way. Whatever this value is will become the base value for incrementing.

This logic is necessary to catch the situation where an object in production has been recompiled. This change would have caused the major version number to be incremented since the object was checked out. If we didn’t have this logic in the process, then the major version number could theoretically be reduced as a result of the promotion. This logic ensures that whenever the major version number is incremented, it is set to the highest value in the application.

Examples of object version numbering

These examples are intended only to give you an idea for the dynamics at work.

Example 1: (2-level App, Level 1 setting: *MINOR, Level 2 setting: *MAJOR)

Version 5.0 of object X is checked out from production. The object is changed and promoted to test. At the time of the promotion, the minor version number of this object is incremented in test and is now version 5.1. If the object were to be changed again before being promoted to production, the minor version number would continue to be incremented (5.2, 5.3 …). When the object is promoted back to production, the version would be changed to 6.0.

Example 2: (2-level App, Level 1 setting: *MINOR, Level 2 setting: *NONE)

You ran the **TSETAPPOBJ** command to set the major version number for all objects within the application to 5.0. Whenever an object is modified, you want to increment the minor version number until you decide to make a new release. At that time, you would run the same procedure to set the major version number to 6.0. You could reverse the ***NONE** and the ***MINOR** parameters if you only wanted to increment when we promote to production.

Example 3: (3-level App, Level 1 setting: *MINOR, Level 2 setting: *NONE, Level 3 setting: *MAJOR)

Similar to example 1, here we define multiple test levels. This example shows how you would keep the version number from incrementing at each promotion. Presumably, you would only want to change the version number when the object was actually changed, rather than when it was promoted. If you incremented the minor version number with every promotion, this would mislead someone into thinking the object was revised twice before being promoted, when, in fact, it was only revised once.

However, one thing to consider is that the files that the programs are compiled over could be different at each level, so the underlying object would, in fact, be different! In that case, you might want to increment the minor version number at each level. It's your choice!

What about source-only types?

TURNOVER® for iSeries v100 tracks the same version information for source-only types, such as copybook code, but no object stamping is done.

What about "special" object types?

TURNOVER® for iSeries v100 tracks version information for special types, such as ***MSG** IDs, but no object stamping is done.

DISTRIBUTING MULTIPLE VERSIONS

TURNOVER® for iSeries v100 will manage the distribution to several related applications so that all of the various objects and their modified versions arrive and are installed on the target computers correctly.

Forms for related applications are made dependent upon their base application forms. If they're run as a form group, they will be distributed and installed in the correct order.

If you distribute the forms manually, the forms are still dependent, but you would have to be sure that both forms run to completion on the target computer.

Note: Use extra care if you're distributing and promoting base and related application objects into the same target library. For instance, you would not want to install local modifications and base objects on top of the modifications you made.

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.