PRC

Software Quality & Testing Guide

Version 6

July, 2008

Notice

The information contained in this document, as well as the software it describes, is subject to change without notice. SJ+ Systems Associates, Inc. makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. SJ+ Systems Associates, Inc. shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

The software described in this document is furnished under a license agreement or nondisclosure agreement. The software may be used or copied only in accordance with the terms of the agreement.

UniVerse, Unidata, SB+, and SBClient are trademarks of IBM Corporation in the United States, other countries, or both. Windows is a registered trademark of Microsoft, Inc. All other brand and product names are trademarks or registered trademarks of their respective owners.

Corporate Headquarters:

5047 Kensington Circle
Coral Springs, FL 33076
Phone: (954) 796-9868
FAX: (954) 796-9868 – call first
http://www.sjplus.com

Version: July, 2008

Table of Contents

Introduction

PRC® is a software development life cycle management application. In other words, it helps you manage or administer the development of your software from start to finish. This publication, the *Testing Guide*, explains the various concepts and features of **Testing** within PRC.

This guide assumes you already have knowledge concerning programming. If you need information or help concerning that subject, please refer to other sources of information.

Who Should Read This Book?

Focus and Audience

The focus of this manual is on the day-to-day mechanics of using PRC. The target audience is the ***programmers***, ***quality assurance technicians***, and ***project managers*** who will use PRC routinely to:

- Open projects
- Track changes to the software being developed
- Review changes, compare versions, and manage project content
- Close projects—mark them complete in preparation for rollout
- Perform rollouts—which may or may not involve programmers, depending on your organization's protocol

General Knowledge Required

This guide covers specific portions of PRC's overall functionality. This book occasionally refers to subjects covered in other PRC manuals. In some cases, base knowledge is assumed in the discussion of the subject. It is assumed that the reader has knowledge of: database administration, generally accepted programming conventions, terminology, software quality management, and some software configuration management.

If background information is needed on these subjects, please see your system administrator.

What is PRC?

At its center, PRC is a ***source control system***: its primary purpose is to keep track of changes to the source (programs and other components) of a software application. Beyond that, PRC encompasses all of the necessary peripheral management that supports its core purpose. PRC accomplishes all of the following:

- **Request Problem Reporting:** Front-line problem reporting, or the "help desk," is a system of logging requests made by the users or management of an organization. Requests can be cancelled or resolved without ever going further through the software development life-cycle, which means you can use the Request/Problem Reporting system to track other non-software requests with your organization.
- **Project Management:** The Project Management system supports traditional functions such as estimating hours, assigning priorities and resources, tracking time, following percentage of work completed, and providing automatic feedback to the requesting user as the project moves through its life-cycle. Projects can be generated from requests or created independently.
- **Source Control:** The system keeps track of the software, protects it from unauthorized changes, logs changes, and provides clear methodologies for deployment, archives, and audits.
- **Test Management:** The system provides a framework for establishing test plans, implementation plans, back-out plans, and the ability to grow multiple plans into full regression and integration test plans.
- **Deployment and Rollback Control:** An automated mechanism for delivering all components of a project from one realm to another—even across machines. It insures that everything attached to a project is archived and can be rolled back at any stage.
- **Security Management:** The system nominates files and realms that may be changed or may not be changed, the conditions they may be changed, and by whom.
- **Auditing & Reporting:** The system provides complete auditing for compliance reporting and research.

Terminology

A list of common terms used in PRC are:

- A ***request*** is a potential or upcoming project.
- A ***project*** is a "unit of work." A PRC project can be further broken down into *sub-projects*.

  Many organizations manage large projects and track work broken down into several tasks. In PRC, however, the smallest unit of work is a project. A sub-project is simply a "child" project to some other "parent" project.

  The basic tenet is that when a programmer is making a change, that change is being tracked against one single project or sub-project.

- A ***version*** is a collection of projects. A version can be further broken down into *sub-versions*, which are still simply a collection of projects.
- A ***realm*** is an environment—a specific region (that is, directory)—that is governed by a particular set of rules regarding what change can be initiated there and by whom; for example, DEV, TEST, and LIVE are realms.
- An ***item*** is a file or object of whatever is considered to be your source. PRC tracks changes made to source items.

PRC is a project-oriented (or "project-centric"') source control system. Everything is performed against projects. Many PRC entities or documents may be in use simultaneously, including: sub-projects, super-projects, versions, master versions, and customer requests. All of these other entities are related to a project. In PRC, the project is king.

There are dozens of different words that mean the same thing, depending on your organization; for example, military implementations use SCRs (Software Change Requests) to represent **tasks**. It is possible to "translate" PRC to match your corporate culture, whether it is the FDA, the DoD, or a small software company.

For further definitions of terms used within this document, please refer to the glossary at the back of this manual.

PRC & Other Source Control Systems

If you have ever used a source control system before, some portions or areas of PRC will be familiar, while other features or areas may be different and harder to grasp (at first).

Basing Work on Projects

The term "project-based" means something different in the PRC environment than it does in a Visual Basic environment. In Visual Basic, a "project" is designated—and controlled—by a file with the **.mak** extension. This MAK file is the controlling element and references all other components: the program, the screen form and other objects, methods, and so on. Each of those components is "versioned" independently, with the MAK file keeping track of which version is current and in use.

PRC is not structured in that fashion. A PRC project emulates a Visual Basic project in some ways: it is a controlling element that keeps track of what components (programs, processes, fields, screens, dialog boxes, error messages, and so forth) are included in this project.

PRC versions each component independently in archives. Program files contain all of the current, in-use programs, called by their real/original names; but, any program that has ever been modified by a PRC project is archived in its various past versions. Taking that approach can recreate any version or point in time (necessary for ISO 9003, IEEE, and DOD compliance) and ***still*** operate in the U2/Multivalue environment.

Checking Out Components

Years back, checking out components in a mainframe environment meant walking down the hall and requesting the tape. While you had the tape, it and all the components on that tape were completely removed from the library. No one else could edit your program because you had it physically in your hand.

Our database development environment makes that kind of total ownership unrealistic (if not impossible). Even knowing in advance what components you need to modify on behalf of a project is not relevant in today's programming environment.

If you feel your organization absolutely must have the ability to pre-check out components, PRC can support that in various ways: turning on Check-Out Mode in the user profile, creating "sandboxing" conventions and indicating them on the Preferences screen, or a semi-pre-checkout approach where each component that is edited is pulled from another account (for example, from the production library) at the time of edit.

What is more convenient in our programming environment is a first-come, first-served approach to checking out components. In other words, if you edit a component and no other project currently has it checked out, it is "belongs" to you:

- If you do not change or edit the component, it is released when you exit.
- If you ***do*** make a change, the component is checked out to your project. Others may still be able to edit the item, depending on their clearance level and the priorities of your project and theirs.

Checking out components, clearance levels, and priorities are explained in the *Installation & Configuration Guide* and must be defined by your management team.

Methodologies For Archiving

People who have used a source control system in the past often ask about archiving and the methodologies behind the process.

One archiving method used by popular "freebie" source control systems in UNIX (SCCS or RCS) is to archive the deltas between one version and another only. This method saves on disk space; but, it requires storing and intelligent parsing of a kind of pseudo-code. Archiving small deltas between two versions requires a pre-compiler to dig through old versions and becomes more and more time-consuming.

Other source control systems archive full working copies of all components every time they are changed. These copies devour a great deal of disk space and can become equally time-consuming to pre-compile.

PRC takes a middle-of-the-road approach. While a project is active, PRC stores a copy of a component every time it is changed or modified. The system rotates through a maximum number of copies, which is determined on the user's profile, each day (the default is three). If the same program on the same project is edited 15 times in one day, a full-copy archive of the program exists from the first time it was edited today, as well as the last three times.

When you close a project in PRC, all of the intermediary archives are discarded and the first copy from the first day is archived into a separate archive file. The copy as it is delivered is also archived, and the copy that is replaced during a rollout/unravel is also archived. In the long run, then, there are three full working copies of every component changed on a project, when the project is completed.

In addition, you have the option to archive the software you are delivering—and the software that it is replacing—when you perform a rollout.

PRC includes a split-screen editor that displays the differences between any two versions of any component. It also provides easy maneuverability to merge them or to revert them.

Where This Book Fits In

Projects go through several ***states*** or stages, from the moment a person thinks something should be added or changed to the moment it is included in the software product for general use. PRC assists you in tracking the projects as they go through their various states along the development path.

PRC assigns a corresponding ***status number*** to each state for easy tracking. PRC is flexible, allowing you to tailor the meaning of some states according to your needs. The meaning of other states, however, are "fixed" and cannot be changed.

In general, projects in PRC go through the following states:

1. The project is formally requested in order to fill a need or correct an issue.
2. The project is reviewed by the appropriate person and approved for development.
3. The project is started by a developer and becomes "active."
4. Development of the project is completed and it is waiting to be tested.
5. **Testing the project is started by a member of Quality Control or Quality Assurance.**
6. **The project passes testing and is waiting to be incorporated into the overall product.**
7. The project is delivered into the software system.

After the first four states, your states or stages may vary widely from this example. You can also skip the first two states, so that you can open projects quickly in a pre-activated state of **3**. The meaning of states **1–4** is hardwired in PRC.

Figure 1 illustrates the project states and state-changing processes within PRC:

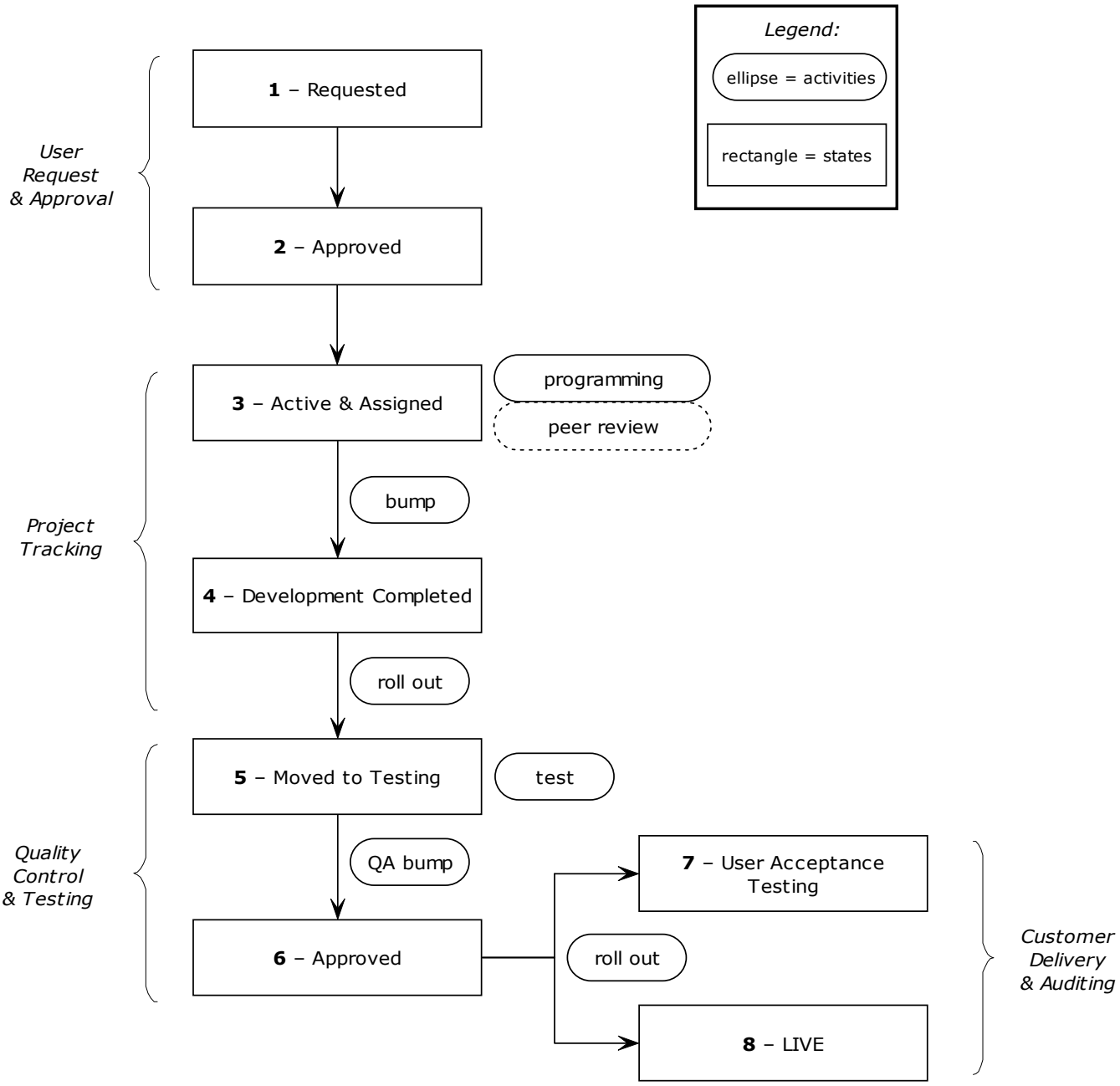

Figure 1: An basic, simplified example of PRC project states and the books that cover them

More PRC Documentation

Information about PRC is available in the following publications:

- *Installation & Configuration Guide*

  For initial and then occasional use by the system administrator.
- *Project Tracking Guide*

  Explains the options for tracking changes during the development process, as well as rolling out projects.
- *Quality Testing Guide*

  Explains options for automating and managing the quality testing process.
- *Software Knitting Guide*

  The definitive guide to merging software using the split screen view.
- *PRC Reports*

  Descriptions of available reports, as well as information for creating custom reports.
- PRC Tip Sheets (Tech Bulletins)

  Each one addresses particular functions and FAQs.
- *Customer Requests & Acceptance Manual*

  Explains how a non-development site uses PRC to make requests and accept the projects derived from those requests.
- *Customer Delivery & Auditing Manual*

  Explains how a non-development site uses PRC to review and accept delivered software changes.
- *PRC International*

  Explains options for managing localization and language translation.

Telnet Programs, Character Mode & GUI Mode

The most common telnet applications used by U2/Multivalue developers (in 2005) seem to be Wintegrate, Accuterm, and SBClient (for those using SB+).

Other telnet applications can certainly be used and in almost all cases some initial effort must be put into creating an interface between PRC and the mode/emulation that your particular telent application is using.

In the case of SBClient, you can use SBClient in either the traditional or "old school" character-based mode in which you navigate strictly using keys and function keys or the graphical user interface (GUI) mode using your mouse as well as the keys on your keyboard.

PRC was written in SB+ in character mode, so navigation is sleek and consistent throughout. All examples and screen captures are shown in character mode.

If you are using Wintegrate or Accuterm certain components can be presented in GUI mode (as of this writing) and there is a large developent focus on this and other client-based GUI front-ends. Check with SJ+ System Associates about what you are hoping to use for a front-end to your development environment.

Menu Navigation

In character mode (Figure 2), you can select a menu option by performing one of the following:

- Use the arrow keys to move to the desired selection, then press Enter or the spacebar
- Press a key that corresponds to one of the underlined letters, such as R for **Review Source Items**, Q for **BUMP/Quick Screen**, and so on

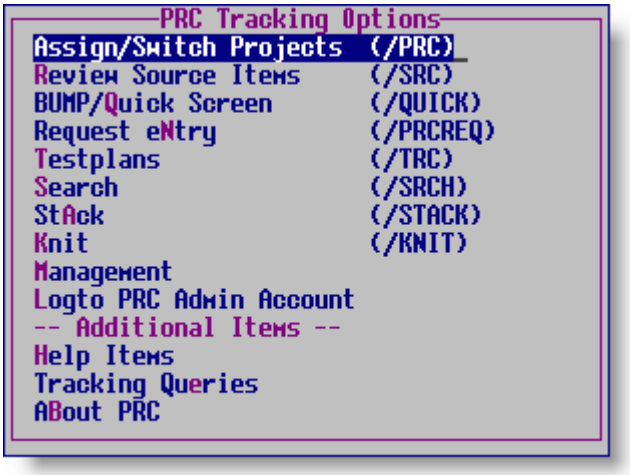

Figure 2: Tracking Options menu in character mode

In SB+ graphical user interface (GUI) mode (Figure 3), you can select a menu option by performing one of the following:

- Use your mouse to point to an option and left-click
- Press the Alt key, release it, and then press a key that corresponds to one of the underlined letters; for example, Alt Q for **BUMP/Quick Screen**, and so on

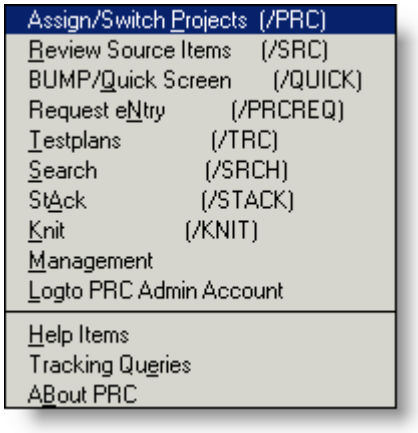

Figure 3: Tracking Options menu in GUI mode

- Just as in character mode, you can use the arrow keys in GUI mode to move to the desired selection, then press Enter or the spacebar

To go back to a previous menu level, use the Esc key.

If you are in GUI mode, some buttons that are labeled sF5 (or something similar). That is shorthand for you to press, in this example, Shift+F5 to display the requested information.

Standard Function Keys

From any screen in PRC, you access various features and subscreens by using function keys. The functions keys that pertain to a screen are listed along the bottom of that screen. Several functions are reserved and always perform the same purpose:

- F1 — Help, which displays a message describing the current field
- F2 — Save or Accept, which saves the values on the displayed record and updates the system
- F3 — Intuitive Help, which lets you select a value for a field from a list of existing values
- F4 — Delete, which removes the displayed record and does not save its values
- F5 — Drill-down, which PRC uses to let you view and modify further information about an item
- F10 — Action Bar, which displays an additional menu bar at the top of the screen (only available if it is listed at the bottom of the screen)

Screen & Field Navigation

Screen

To exit a screen without saving any input values you may have entered, use the Esc key.

Press the F2 key to both accept the current data in a screen and exit the screen. You can press F2 at any time, but the system will allow you to exit only if you have entered values in all mandatory fields.

If the system displays **INS** in the lower right corner, the screen is in full-text mode. Whenever it is in full-text mode, you can press Enter to add lines. You will have to press an additional F2 to close that particular field and move to the next one on the screen.

Field

To navigate through the fields of a screen, use ↓ or Tab keys to move forward from field to field, and ↑ and Alt+Tab keys to move backward to previous fields.

To clear a field and make its value null, use the backslash (****) character as the first character in the field. You can press the Delete key repeatedly to delete one character at a time.

To invoke other features and processes, use the slash (**/**) character. When you press the slash character, the system displays command line in which you type a command for the feature or process you wish to invoke. In other words, it is a two-step procedure. In this book, the text will not explain the command line every time and will abbreviate the two commands; for example: to call up TCL, type **/TCL**.

Text Conventions

The text formatting in this document uses the following conventions:

Table 1: Typographical Conventions

Font	Description
Bold San serif	The name of a PRC module or folder
Bold	The name of a menu, menu item, toolbar button, or icon
Regular	The name of a screen, dialog box, property sheet, or tab
Italic	The name of a control (field, button, or drop-down list) within a screen or sub-screen
Courier	A path name, system value, or in general, text displayed by the computer
Bold Courier	File names, file extensions, or in general, text to be entered by the user or system administrator
Enter, Tab, Esc	A key on the keyboard
Ctrl+S	A key combination where you press the keys simultaneously
Alt P	A key combination where you press the keys one at a time in the order presented

Chapter 1

Quality & Testing in PRC

Overview

Quality management (QM) begins at the beginning of any software project, and persists throughout the software development life-cycle. PRC aspires to provide cohesion and vision to QM throughout the life-cycle and to provide the tools to automate and support good QM processes.

There are two main aspects to test management:

- The ability to set up test plans that can be general or that can be for specific projects.

  These can be very simple or very elaborate. For more information on test plans themselves, see Chapter 3, "Test Plans."
- The ability to use the "QA bump" feature by attaching these test plans to projects and marking the steps as passed or failed.

  The QA bump—not to be confused with a jerky dance move invented by some misguided but well-meaning quality assurance engineers during the disco era—is a nice addition to your process even if the test plan itself is very simple.

  Using QA bump goes like this:

  a. You attach a test plan to a project.

  b. You use the test page to mark the test passed or failed (and enter specific information into the iteration details at that time).

  c. You press F7 (Bump).

     - If the step passed, the project is bumped (moved up a state to "tested," whatever that state is on your system).
     - If the step is failed, the QA bump will automatically open a sub-project for the step or steps that did not pass. The details about the failed steps are inserted as the description on the new sub-project. Optionally, the system sends an e-mail to the appropriate personnel automatically (via the status and/or notification.)

In this book, we will elaborate on what test plans can be. One PRC user was creative in instituting the use of one (and only one) test plan that had only one step: "Test it." This generic test plan was attached to all projects so that the steps above could be taken.

There are a number of steps to initially setting up the system to support Test Management. These are outlined in Chapter 2, "Quality & Testing Configuration."

To make your life-cycle complete and secure, it is recommended that you take some additional steps:

- Restrict manual changes of projects to states that indicate testing has occurred.

  Use the Status Code Maintenance screen. See "Project States" on page 25.
- Indicate notifications so that the requesting user or manager is notified when a project passes testing and the programmer can be notified when it fails.

  Again, use the Status Code Maintenance screen.
- Establish realm government such that a project can only be promoted/delivered to LIVE or the next Realm when it has arrived at the state at which the QA bump places it.

Features & Benefits

- Creating re-usable / hierarchical test plans
  - General Standards
  - Software type – standards
  - Screens, reports, fields, and so on
  - Application type – standards
  - Maintenance, batch processes, transaction processing, etc.
  - Application Specific
  - Order processing, Inventory change, Check processing, etc.
  - Project Specific
  - Requirements doc or Spec of a particular change request
- Attaching test scripts or screen shot paths
- Running iterations of test steps
  - Marking pass/fail, tracking iterations, storing remarks
- Bumping the project/test plan
  - **PASS —** All steps/all iterations: project moves to the next status with electronic user and date sign-off.
  - **FAIL —** Any iteration of any step: sub-project opened with the failure notes on it and programmer is optionally notified.
- Coverage (how much of the software is getting tested during testing cycles)
  - Source attachment
  - Source code can be attached to a particular test so that when that source item changes again PRC knows to pull in the past tests that have been associated with it.
  - If separate logins are used for testing sessions (as marked on the user profile), "run histories" of the software can be kept (an audit subroutine call must be placed in all programs).
- Increasing Automation

- History / repeatable tests against software components
  - Test plans that call test plans
  - Using test script programs and automatically noting results
  - Regression Testing — As test cases are built and attached to various top-level components, a complete regression test will gradually be built.

Chapter 2

Quality & Testing Configuration

Setting Up the QM Module

You have to configure several things to ensure your Quality Management (QM) module works properly and efficiently. You must:

- Indicate one or more realms as being testing (or QA) realms.
- Associate project states that can be tested within those realms.
- Create and define user "roles" and associate them to particular project states.

You must think through your quality management practices and workflow. A well-planned process that is set up properly produces an operation that runs very smoothly.

Roles

Role codes and status codes refer to each other. Role codes are explained in this section and status codes are explained in "Project States" on page 25.

The quickest way to explain a ***role*** is to describe what it is and what it is not:

- **What a role is —** It is a name for the person associated to that particular state on a particular project.
- **What a role is not —** It is not the title of a real job of a particular user.

To help explain a role further, consider the following example:

Your system has the following:

- A project status code **`6`** that has the description **`Tested`**
- A role code **`TESTER`** that is associated with status 6

When a person signs off on a project that has the status **`6`**, the person is acting in the role of **`TESTER`**; however, in real life, the person is a business user, or a programmer, or belongs to a different security group. At the same moment on a different project, the person is acting in the role of **`CODER`**, which is a role related to status **`3`**.

A role is a name for the sign-off slot associated to a particular status and refers to the individual person who signs off on a particular project at that state.

The role of an individual prescribes their duties and associates them with a particular life-cycle state. In the Role Code Maintenance screen, you define the various roles that anyone can play. An individual may be authorized to certain roles in certain states of certain types of projects, if desired. For more information, see "Type Codes," "Status Codes," and "User Profiles" in the *Administration Guide*.

> For example, Sally is a summer intern and belongs to the **INTERN** group. Permanently, her name is Sally; semi-permanently, she's an INTERN (until someone changes her group); but transiently, her role within the system is fluid.
>
> If she does the programming on one project and is the status **3** sign-off person, her role is **DEVELOPER** (which is defined in the system as a role associated to status **3**).
>
> On Tom's project, she performs a peer review for him. On that project, she functions in the role **PEER_REVIEWER** that was set up as a role associated to status **5**.
>
> So in this example: she's still Sally (always); she's still an INTERN (until that semi-permanent designation is changed for some reason); but while being Sally the intern, she can function in any role to which she is authorized on various projects.

When you set up roles for testing through role codes, perform the following steps:

1. Select **Files > Code Files > Roles Codes**.

   The system displays the Role Code Maintenance screen, similar to the one shown in Figure 2-1:

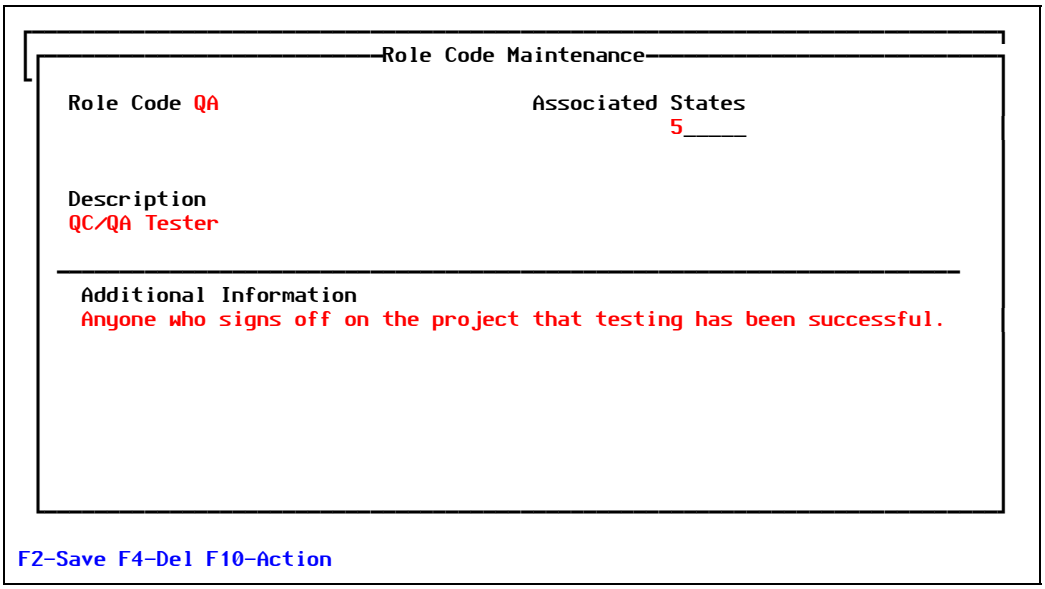

Figure 2-1: Role Code Maintenance screen

Explanations of the fields are as follows:

Role Code — The code or ID that represents activities an individual will perform in conjunction with different types of projects and project states.

	A role is similar to a job description or job title, in that how an individual relates to a project is implied. The role of an individual prescribes their duties associated to a particular life-cycle state. Press F3 to select from the list of valid entries.
States	The project state associated with this role. For example, the role PROGRAMMER is associated with the project state **Active & Assigned**. Press F3 to select from the list of valid entries.
Description	A concise but meaningful description of this role code.
Additional Information	Further descriptive information others should know about this role.

2. Fill in each field with the appropriate information.
3. Press F2 (Save) to save this role code.

Create roles that make sense in your organization. Common definitions include **TESTER** or **QA**. Some companies have different phases of testing and may have roles for **UNIT TESTER** and **SYSTEM TESTER** and **USER TESTER**.

For each role that you create, simply associate one or more states of project—meaning at what status should a project be associated to an individual in this role.

QA Preferences

After setting up the codes that define roles, you must indicate which realms are set aside for QA testing, at what state they can be tested, and to what status they will be "bumped" when testing is completed.

To set up system-wide QA preferences, perform the following:

1. Select **Setup > Preferences**.

   The system displays the PRC Preferences screen.
2. Press F6 (Addl Opts).

   The system displays the More Preferences screen.
3. Select **QA Options**.

   The system displays the QA Preferences subscreen, similar to the one shown in Figure 2-2:

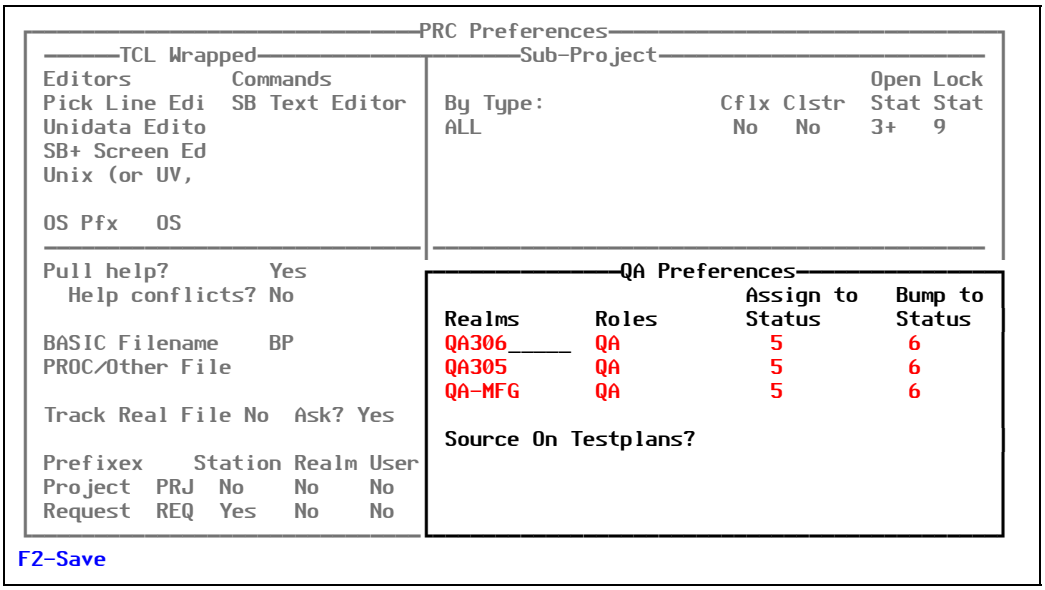

Figure 2-2: QA Preferences subscreen

Realms

The realms in which quality assurance (QA) will be performed.

Press F3 to select from the list of valid entries.

Roles

The role code that represents the QA activity in this realm. For more information, see "Roles" on page 21.

Press F3 to select from the list of valid entries.

Assign to Status

The status code that represents the state in which a project must be when it is tested in this realm. For more information, see "Status Codes" in the *Administration Guide*.

Press F3 to select from the list of valid entries.

Bump to Status

The status code that represents the state to which project will be bumped when QA is completed in this realm.

Press F3 to select from the list of valid entries.

Source On Testplans?

Yes turns on the feature where source items are stored against tests. Valid entries are:

- **Y** — Yes
- **N** — No

Press F3 to select from the list of valid entries.

4. When you have finished with the settings, press F2 (Save). The system returns you to the More Preferences screen.

Project States

The next step is to set up status codes to include QA testing.

Status codes are the most important of all of the codes in PRC. They define the development life-cycle.

PRC expects projects to move through numbered life-cycle states. The first four states are absolute in PRC. They are:

- **1** — Requested
- **2** — Reviewed/Approved
- **3** — Active/Assigned
- **4** — Development Complete

Beyond these four, you can define any number of states. A typical and simple implementation includes: **5** (Moved to `TEST`), and **6** (Moved to `LIVE`). You can add other numeric life-cycle states plus codes for things like cancelling, closing, placing on hold.

You can also use Alpha codes; however, no hierarchy or life-cycle is defined by Alpha codes. You would use them for closing out projects, cancelling projects, or perhaps placing projects on hold.

Two Alpha states are predefined with special behavhior.

Special codes defined in PRC are:

- **C** — (begin with "C") will ask, then check-in software still checked out to the project
- **X** — (begin with "X") will ask, then not only check-in the software but optionally replace it to its orinal state

To work with status codes, perform the following:

1. Select **Files > Code Files > Status Codes**.

   The system displays the Status Code Maintenance screen, similar to the one shown in Figure 2-3:

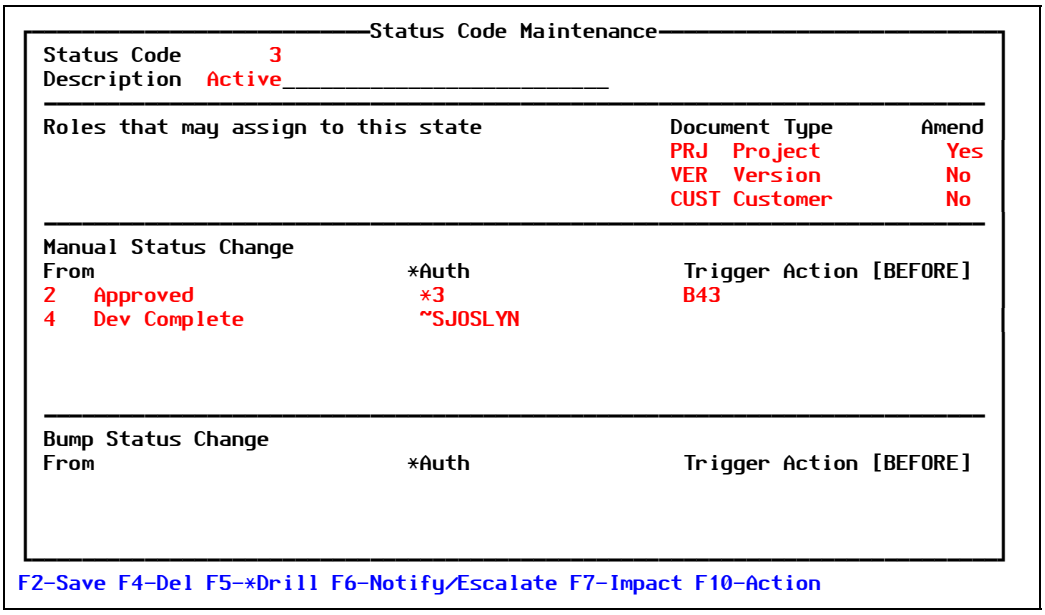

Figure 2-3: Status Code Maintenance screen

Explanations of the fields in the top portion of the screen are as follows:

Status Code A code or ID that represents meaningful "states" or "life-cycles" for your software versions, projects, and requests. Default, pre-defined values are:

- **1** — Requested
- **2** — Approved
- **3** — Active & Assigned
- **4** — Development Complete

You can define any number of numeric status codes after **1–4** that can mean anything you want them to mean. In addition, you can define any number of alphabetic status codes with any meaning. The examples that follow are provided to give you an idea of what you can do with your system.

- **5** — Moved to QC (Quality Control)
- **6** — Passed QC
- **7** — Moved to LIVE
- **A** — Accepted (User)
- **C** — Closed/Checked-In

  Special codes that begin with **C** will ask whether you wish to check in (release) the software, then check in software still checked out to the project.
- **O** — Open (Request)
- **R** — Resolved
- **T** — Tested

- **V** — Converted (Request to Project)
- **X** — Cancelled

  Special codes that begin with **X** will ask whether you wish to check in (release) the software, then not only check in the software but optionally replace it to its orinal state.

You can identify life-cycle states by project type.

Press F3 to select from the list of valid entries.

Description A concise but meaningful description of this status code.

Document Type The kind of item that uses this status. Valid entries are:

- **PRJ** — Project
- **REQ** — Request
- **VER** — Version
- Any combination of the above values

  For example, if a status code is to be used on all three types of documents, put all three in this field column.

This "document type" field, common to many of the PRC support code files, indicates the document type or types on which this status code may be used.

Press F3 to select from the list of valid entries.

Amend Indicates whether documents of this type can be amended when they are in this state. Valid entries are **Y** (Yes) and **N** (No).

Press F3 to select from the list of valid entries.

You can change the status of a project one of two ways by:

- Going into the PRC account, calling it up on the Project Master screen, and manually changing the status (which can be authorized for specific people, groups, or roles). Use the *Manual Status Change* area of the screen to set the parameters of this kind of status change.
- Bumping it from within the working account (**DEV** or **TEST**). Use the *Bump Status Change* area of the screen to set the parameters of this kind of status change.

Explanations of the fields in the *Manual Status Change* area of the screen are as follows:

From Indicates the previous status of the project, request, or version before it became this particular status.

Auth Establishes *who can change* a project to this state.

For example, if you are defining status **4**, you would typically enter here who can change it from **3** and who can change it from **5**.

Trigger Action [BEFORE] — The program or process from which a manual change from this state occurs.

A "bump" occurs when you press F7 from the Review Source screen or the Test Plan screen to move a project up one status (or in some specific cases down).

Explanations of the fields in the *Bump Status Change* area of the screen are as follows:

From — Indicates the previous status of the project, request, or version before it became this particular status.

Auth — Establishes *who can change* a project to this state.

For example, if you are defining status **4**, you would typically enter here who can change it from **3** and who can change it from **5**.

Trigger Action [BEFORE] — The program or process from which a bump to from this state code occurs.

2. Fill in each field with the appropriate information.
   - If you press F5 (*Drill), see "Drill Down" below.
   - If you press F6 (Notify/Escalate), see "Notify/Escalate" on page 30.
   - If you press F7 (Impact), see "Impact" on page 32.
3. Press F2 (Save) to save this status code.

Drill Down

From any field on the Status Code Maintenance screen (Figure 2-3 on page 26):

1. Press F5 (*Drill).

   The system displays the sub-screen in Figure 2-4:

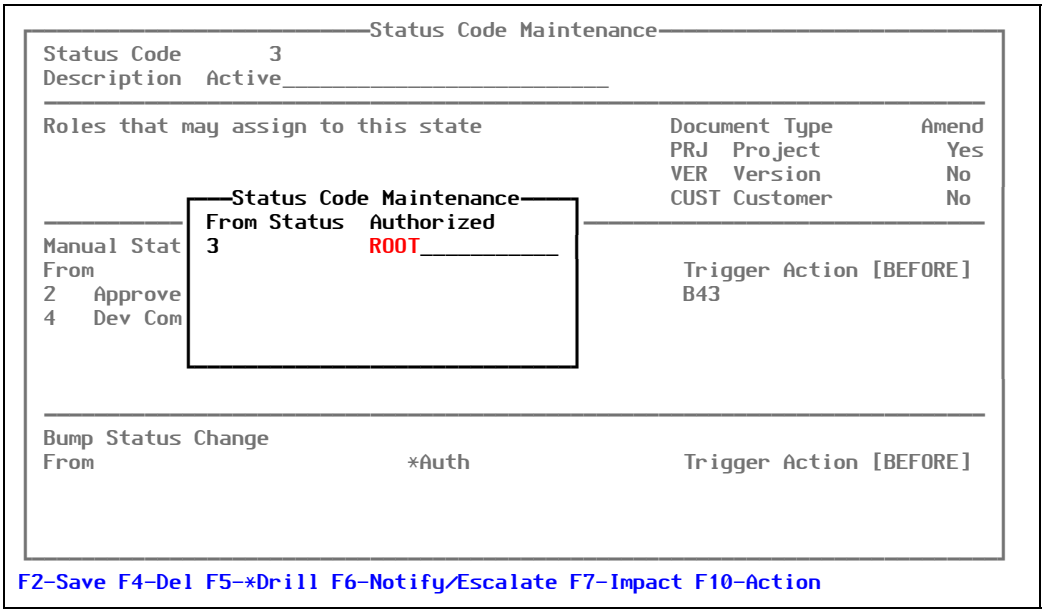

Figure 2-4: Status Code Maintenance drill-down sub-screen

Use this sub-screen to define one or more person (or group or role) to be authorized for this particular status change. When more than one entry is listed, the main page will display the asterisk (to remind that a drill-down is available) and a count.

The example in Figure 2-4 shows **ROOT** on the main page because that is the only authorized "entity" to change a project's status to **3**.

2. Press F2 (Save) to save the values on this sub-screen.

   The system returns you to the Status Code Maintenance screen (Figure 2-3 on page 26).

Notify/Escalate

From any field on the Status Code Maintenance screen (Figure 2-3 on page 26):

1. Press F6 (Notify/Escalate).

   The system displays the Status Emails screen, as shown in Figure 2-5:

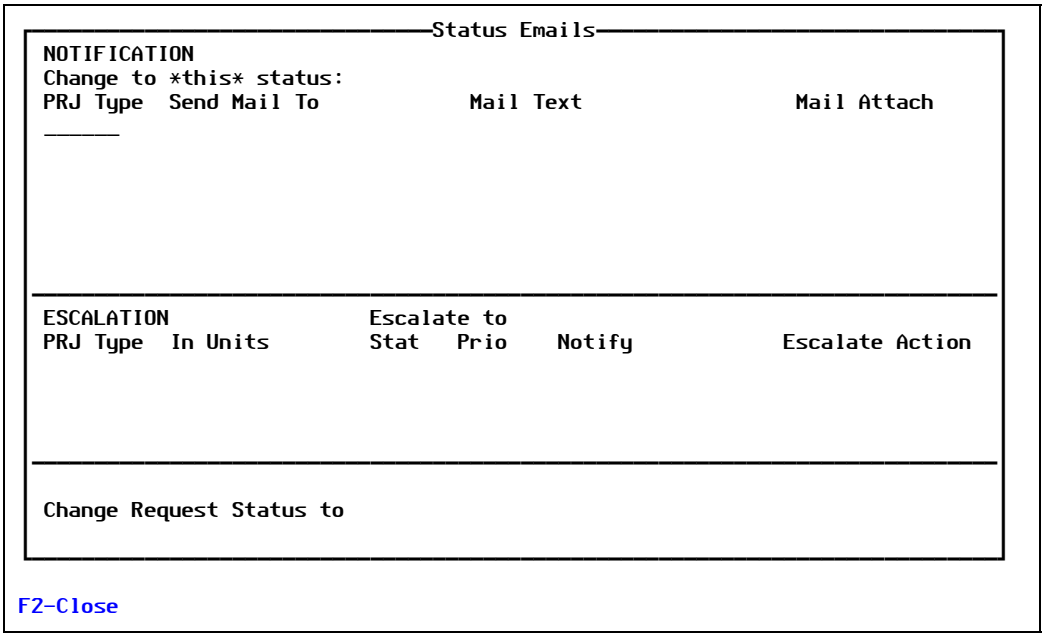

Figure 2-5: Status Emails screen

Use the *NOTIFICATION* area of the screen to define automatic e-mails that, based on the project type, notify a particular person, group, or role that an event has occured.

Field explanations for the *NOTIFICATION* area of the screen are as follows:

PRJ Type The type of project for which this definition applies.

Send Mail To The role to which this e-mail will be sent. If multiple users are assigned to that role, they will all receive the e-mail.

Mail Text The body of the e-mail that will be sent.

Mail Attach The path to a file that will be attached to the outgoing e-mail.

Use the *ESCALATION* area of the screen to indicate an escalated action must be take by a particular person or role.

Field explanations for the *ESCALATION* area of the screen are as follows:

PRJ Type The type of project for which this definition applies.

In Hours Answers the question, "How many hours will a project be in this state before it should initiate an 'escalation action?' "

Escalate to Stat, Prio — The status code and priority code to which the project should be escalated.

Notify — Indicates whom to notify via e-mail that this escalation has occured.

Escalate Action — The name of a program that should be executed when this escalation occurs.

Change Request Status to — Identifies the status code to which a request or event should be set when the PRC project reaches this status.

Press F3 to select from the list of valid entries.

2. Press F2 (Close) to save the values on this sub-screen.

   The system returns you to the Status Code Maintenance screen (Figure 2-3 on page 26).

Impact

From any field on the Status Code Maintenance screen (Figure 2-3 on page 26):

1. Press F7 (Impact).

   The system displays the Status Change Impact sub-screen, as shown in Figure 2-6:

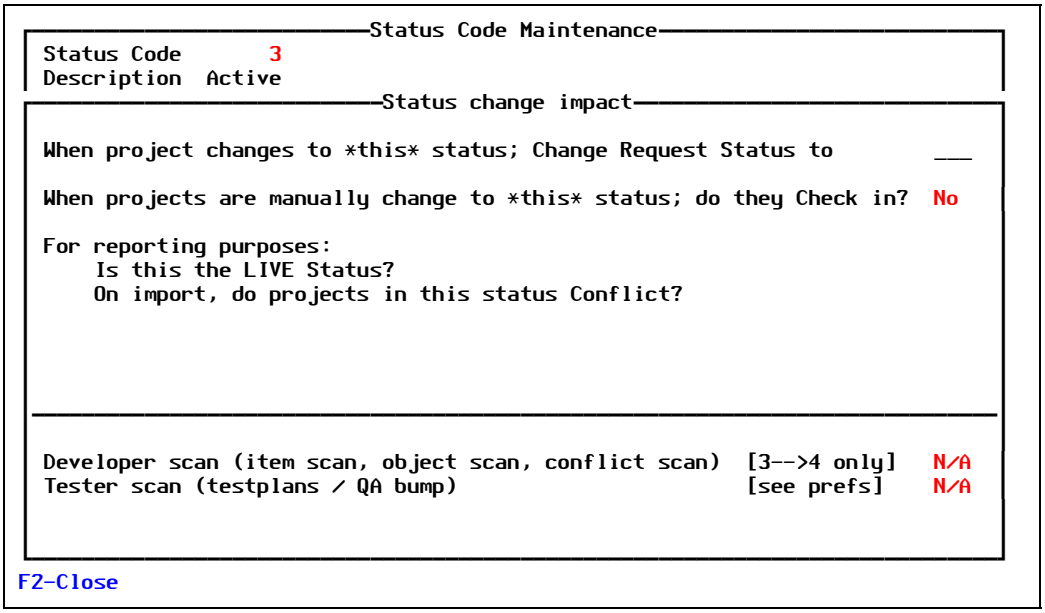

Figure 2-6: Status Change Impact sub-screen

   Use this sub-screen to dictate the impact of either a bump or a manual change.

2. Use the *Change Request Status to* field to indicate a status change to an associated request when:

   - The project arrives at this status
   - The Request system is in use
   - This project is linked to one or more requests

   Valid entries for the *Change Request Status to* field are any defined status level.

3. Use the *Do they Check in?* field to indicate that source items should be checked in when a project is manually changed to this current status.

   Valid entries are **Yes** and **No**.

4. Use the *Is this the LIVE Status?* field to indicate that this is the status at which a project will arrive when it has "gone live."

   This is for reporting purposes, so you can list projects, use the field `LIVE.DATE`, and PRC will find the correct status-associated date.

   Valid entries are **Yes** or **No**. Only one status should be **Yes**.

5. Use the *On import, do projects in this status Conflict?* field for customers with complex, multi-development platforms.

This allows programmers to import work that was performed off-site and control the conflict between which projects in which states.

Valid entries are **Yes** or **No**.

6. Press F2 (Close) to save the values on this sub-screen.

   The system returns you to the Status Code Maintenance screen (Figure 2-3 on page 26).

User Profiles

On the User Profile Screen (**Files > User Profiles**):

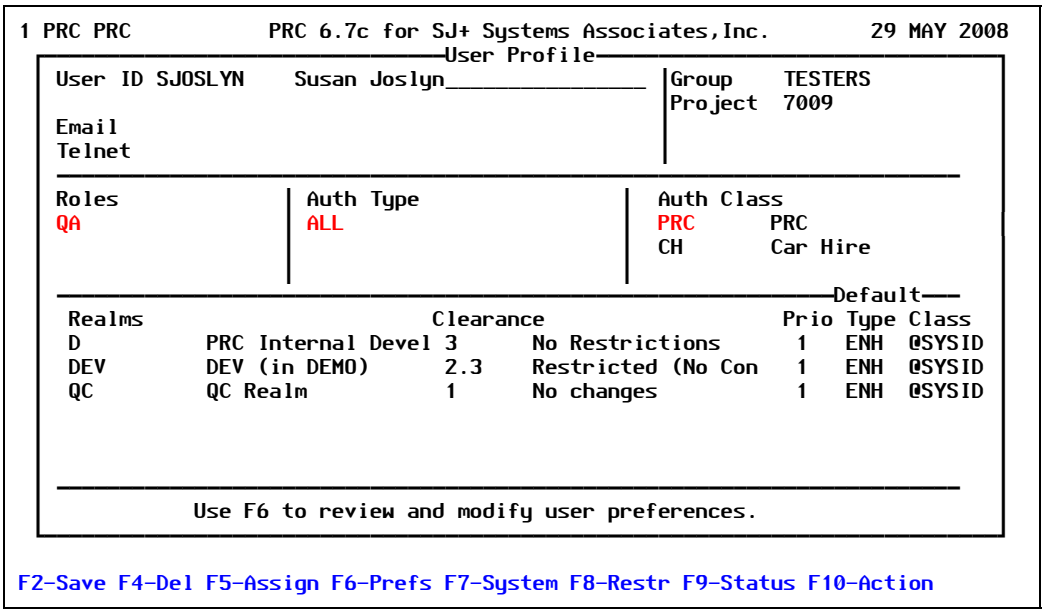

Figure 2-7: User Profile screen

For each authorized tester's profile, use this screen to indicate they are qualified for their individual testing roles.

Chapter 3

Test Plans

Types of Test Plans

You can create three different types of test plans:

- Project
- General
- Hierarchical

Individual Project

When you access the Test Plan page for the first time on any project, PRC displays a fresh test plan template. It automatically copies the project ID and makes it the key to the test plan ID. As you enter the test plan, it is specific to that project. You enter each step, its description, and the expected results.

General Re-Usable

If you want a test plan to be more general and re-usable, it should be entered on the Test Plan maintenance screen beforehand and given a meaningful name—not specific to one project, but one that can be stored and used again later. You can enter a name for the test plan ID rather than a project ID.

When you are working on a project and call up a general test plan, all of its steps will be pulled into the template for the current project. You can then modify them as desired without affecting the standard (general) test plan. You can save (store) this new test plan as well, using the current project number as the key.

Hierarchical

You also have the ability to use general test plans within other test plans, creating a hierarchical set of steps. You can enter each of those test plan IDs as individual steps of the overall test plan template for the project.

A test plan like this is "exploded" so that you can view all of the steps of the included test plans, giving you a complete listing.

As you create more test plans and grow in your use of them, you may find that you can use hierarchical plans to build a complete regression test.

Building Reusable Test Plans

There are as many ways to think about testing as there are testers in the world:

- If you have the "do it as we go along" personality, you can start by describing how you are going to test each project individually. After a while, you go back and review these project test plans to see if a more general testing pattern emerges.
- If you have the "plan it in advance" personality, you can consider some of the ideas used within PRC and within other testing frameworks.

Here are some specific suggestions and concepts to consider when you build general test plans:

- Think about using ***keywords*** to identify types of projects in words like `SCREEN`, `MAINT`, `UPDATE`, and `REPORT`.

  Create general test plans that go with these kinds of projects. A data entry screen, type SCREEN, for example, might have a test plan associated with it that outlines certain criteria, such as:

  ```
  KEY PROMPT ON ROW 1
  DATA ENTRY FIELDS LINE UP VERTICALLY
  ALL FIELDS HAVE FIRST LEVEL F1-HELP
  ```

- Create highly specific test plans that are associated with the major top-level processes.

  Consider naming them with one of your general keywords and a specific keyword; for example, `SCREEN` and `CUSTOMER`. In this case, the `SCREEN` general test plan is implied. Other examples are:

  ```
  SCREEN.CUSTOMER
  Standard key built
  Cannot delete if there is a balance
  Must have 9-digit zip code
  ```

  You can create action words that have specific test cases and test data, such as:

  ```
  SCREEN.CUSTOMER.ADD
  SCREEN.CUSTOMER.DELETE
  SCREEN.CUSTOMER.MODIFY
  ```

- Make general test plans for `DATE_FIELD`, or `TOTAL_FIELD` with expected results that describe software standards to be reviewed.

  You can use this approach instead of or along with the others listed above.

- Have everything—even the "little" things—on a checklist to ensure they are tested.

A lot of what is easy to codify and standardize is the easy stuff—but most people are surprised when they sit down and ask themselves about those kind of simple standards. Do we always have the key prompt on a data entry screen at the same location?

Having a test step for `FIELD_DATE`, `FIELD_NUMERIC`, and `FIELD_CALCULATED` can remind us to review some very simple basic functionality re-tests. This, in turn, brings our overall quality of deliverable way up.

If you think about it, most of the mistakes that get left in software that is delivered are the little things that you didn't consider testing.

- Whatever naming convention or overall design you decide upon, if you begin creating stored test scripts, name them in a consistent fashion.

Attaching Source Items to Test Plans

If you enter **`Yes`** in the *Source on Testplans* field on the Preferences screen (page 24), the test plan is automatically associated with the programs and other source items that you indicated on ***that*** project whenever you want to use that named test plan on ***another*** project. In other words, the next time you have a project with that screen or program on it, PRC can offer to "pull" in the same test plans you used previously.

When you call up a test plan template for a project for the first time, PRC automatically displays the screen for attaching source items. You can access the screen again by pressing the F6 (Source) key.

The "attach source" screen looks like the example in Figure 3-1:

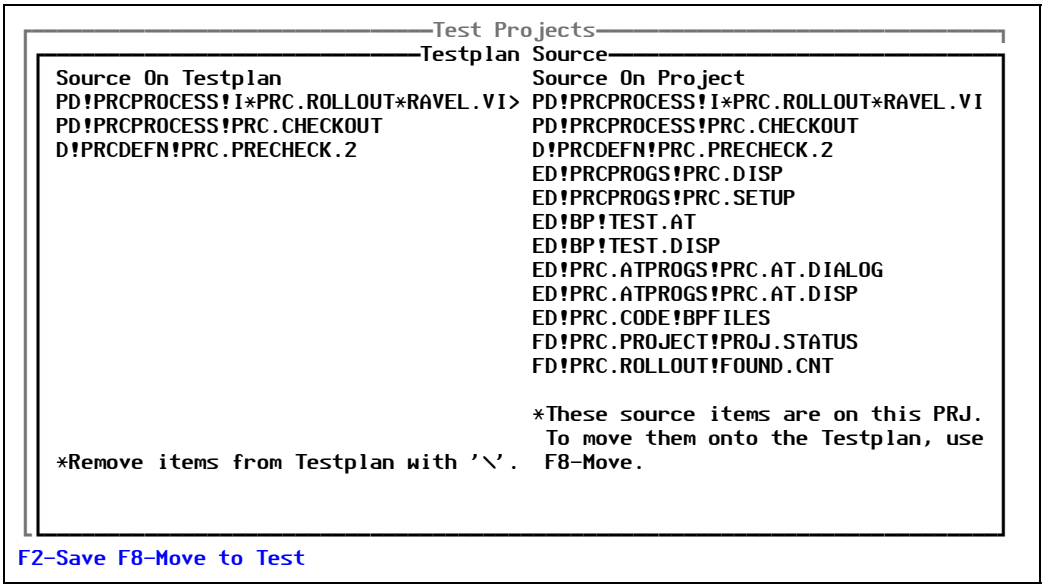

Figure 3-1: Test Plan Source sub-screen

All of the source items attached to the project are listed in the right-hand column (*Source On Project*). To attach those source items to the test—so that a permanent reference is made to that source item and this test—place your cursor over each item of interest and press F8. This will cause that source item to show up in the left-hand column (*Souce On Testplan*).

You can remove items from the reference to the test plan by using the standard backslash (****) navigation. It is the items in the ***left*** column that will be used to pull test plans forward in the future (for an explanation of "pulling" test plans, see "Pulling Stored Test Plans" on page 39).

Using Test Plans

You can attach, iterate, and review test plans from the following three locations:

- The Project source screen

  From the Review Source screen, press the F10 key and select **Q** (QA). The system displays the Test Plan screen.

  You can:

  - *Attach* a test plan to a particular project from this location.
  - *Iterate* a test plan against a project from this location.
  - *Review* a test plan from this location.

- The **/PRCMENU** command

  You can access test plans directly from the PRCMENU or by typing **/TRC**.

  You can:

  - *Attach* a test plan to a particular project from this location.
  - *Iterate* a test plan against a project from this location.
  - *Review* a test plan from this location.

- The Project Master Maintenance screen

  From the Project Master Maintenance screen, press the F7 (Testplan) key. The system displays the Testplan (project) sub-screen (see Figure 3-2).

  You can *review* a test plan from this location; in other words, you can see what test plans are attached to a project.

  However, you can neither change, add, nor delete test plans nor iterate testing. The following message displays at the bottom of the screen:

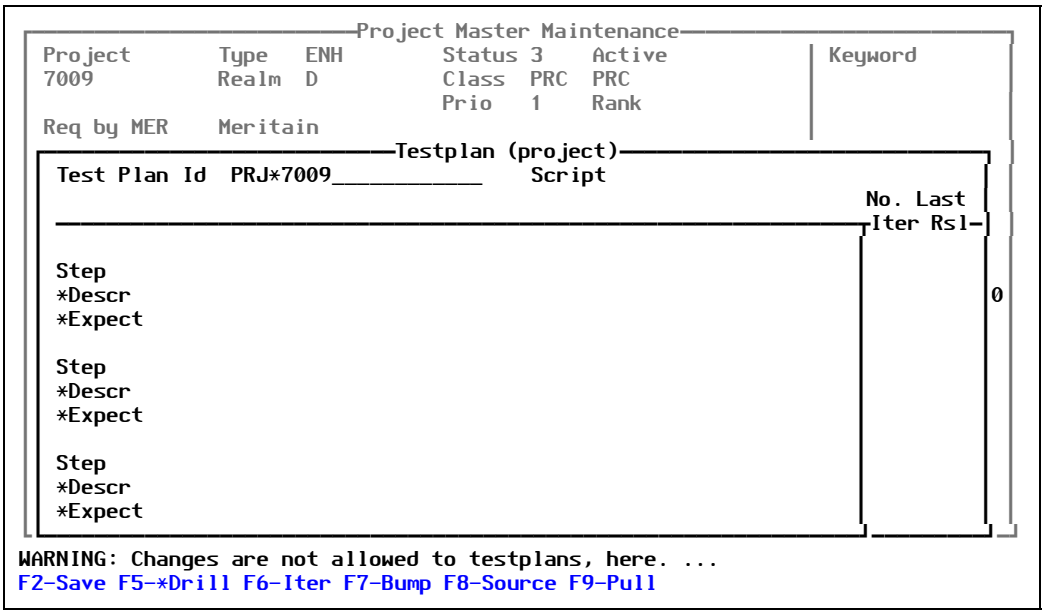

Figure 3-2: Test Plan (project) sub-screen

Pulling Stored Test Plans

You can pull stored test plans onto a new test plan automatically, based on the source items that are on the test plan and to which test plans those source items have previously been attached.

To pull stored test plans onto a new one, perform the following:

1. Use the "attach source" function—press the F6 (Source) key—to indicate which source items should be attached to this test plan.
2. Press F8 (Pull) to gather all previous test plans that have had each of those source items attached previously.

   Each test plan will be pulled on as a step of this test plan.

Working with Iterations

Put your cursor on a particular test step and press F5 (Iter). The system drills down into the specific iterations of testing against that step, then displays the Testplan (project) iterations sub-screen.

An example is shown in Figure 3-3:

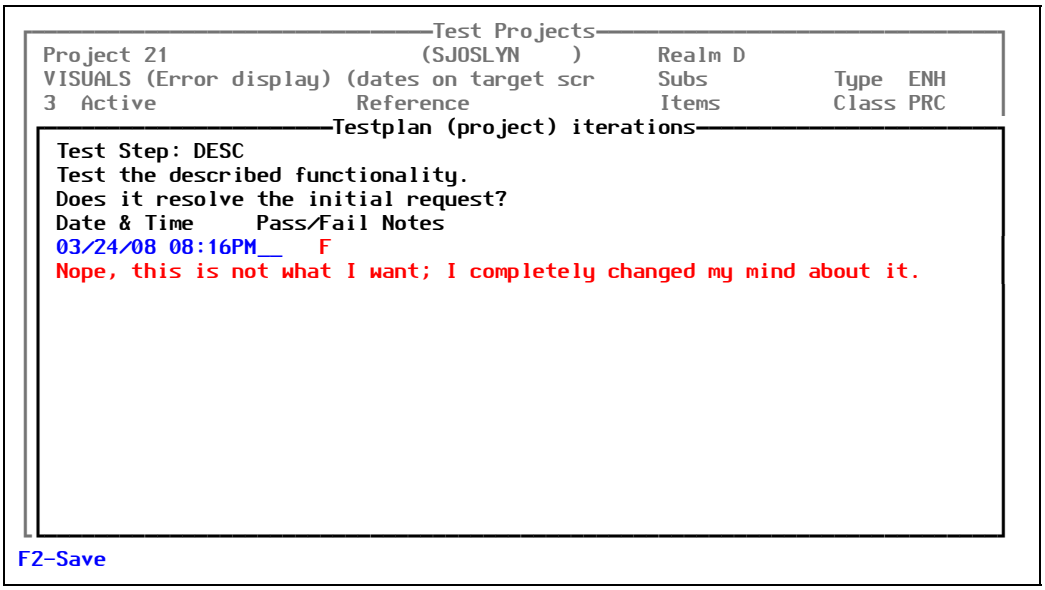

Figure 3-3: Test Plan (project) Iterations sub-screen

The step name and the description and expected results are displayed at the top of the Testplan (project) iterations sub-screen.

The time and date for this iteration of this test step are displayed on a line in which you can enter whether the step passed (**P**) or failed (**F**).

Below that is a line for notes about this iteration of this test step.

- These notes will create the description of created sub-projects by bumping when there are failed steps.
- Later, when all of the test steps have been executed (iterated), each step has a pass or fail marker on it.
- When you attemp to bump the project:
  - If all steps are passed (**P**), the project will bump.
  - If there are failed steps on the project, PRC offers to create sub-projects. If you create sub-projects, the text from this description line about what failed is inserted into the description of the created sub-project.

    In addition, you can optionally send an e-mail to the programmer to let them know some of the steps failed and a sub-project has been opened for their re-work.

Chapter 4

Quickstart

Set it up and give it a spin! This chapter contains a common (but not required) sample configuration.

Setup

1. Set up a role (see "Roles" on page 21). Name it something like **QA**.
2. On the Role Code Maintenance screen, place the role in status **5**.
3. Go to the Status Code Maintenance screen (Figure 2-3 on page 26) for status **5**.
4. Identify the **QA** role as "qualified."
5. Identify yourself on the user profile as being qualified for that role (QA).
6. Go to the QA Preferences subscreen (Figure 2-2 on page 24).
7. Set up your test or QA realm (whatever you call it) to allow users qualified for **QA** to:
   - Be assigned to projects in status **5**
   - Bump projects to status **6**
8. Build a test plan that simply states:

   ```
   Test this
   ```

   Other very simple examples of test steps are:

   ```
   Validate that new items can be added
   Validate that existing items may (or may not) be modified
   Validate that items may (or may not) be deleted
   ```

   Use whatever is appropriate to the particular software being tested.

Routine

1. Open a project.
2. Do some work on that project.
3. Bump the project to status **4** (Dev Complete).
4. Roll the project to the QA realm.

   This should be set to change the state of the project to **5**.

QA/Test Management Activities

1. Log in to the test/QA realm.
2. You can assign yourself to a project.

   In other words, you can log in to the account and use the **`/PRC`** window to actually assign yourself to that project. That way, whenever you call up the various screens, this project ID will default in. However, it is not necessary to be assigned to a project to execute test steps and store information about those iterations.

   As a member of QA, you are able to assign yourself to a project even though it is a higher state—and, as a programmer, you would not have been able to assign yourself. It is not necessary to assign yourself to a project in order to load test information about that project; but, it ***is*** a convenience because the current project number will default into various input prompts.
3. Using the **`/SRC`** **> F10 (Action) > QA or Test Plan** from the **`PRCMENU`** or use the short-cut **`/TRC`** to add a test plan to the project.

   At this point, you can use your pre-built test plan or you can enter the steps directly against this specific project.
4. Test the project according to your steps.

   From the test steps screen, you can press F5 (Drill) to drill down and indicate whether the project passed or failed a step.
5. Fail a step just to see what happens.
6. Using **`/SRC`** again, press F10 (QA) and choose "QA Bump."

   Because there is a failed step, this project will not bump; instead, the next sub-project will be opened with the original project's description, as well as a description of the failed step(s) in the "full description" window.

   If you have the status **3** set to e-mail the assigned programmer, the act of QA bumping a project and having it fail will open the project and e-mail the programmer, informing them that the project has gone back to them.
7. When the time comes to roll to live, PRC asks whether to roll just one or all sub-projects (including the original parent).

   You can select **`All`** to move the entire completed set of projects to live.

8. Once the project passes all the steps on the sub-projects, indicate on the parent project that they are passed and attempt the QA Bump again by selecting **/SRC > F10 (Action) > QA or Test Plan**.

   This time, the project (and its sub-projects) will l be moved to status **6**.

> **Note**
>
> If you use a client-side tool to memorize keystrokes or any other desktop utility to store test scripts, you can reference the path location of that script on each test plan and on each line of each test plan.